jamk

Evaluating the Suitability of the MERN Stack in the Development of Food Delivery Applications

Dilshad Sumer Mohammad

Master's thesis
December 2023
Masters in Full-Stack software development

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Dilshad, Sumer Mohammad

Evaluating the Suitability of the MERN Stack in the Development of Food delivery Applications

Jyväskylä: Jamk University of Applied Sciences, December 2023, 63 pages

Degree Programme in Full Stack Software Development. Master's thesis.

Permission for open access publication: Yes

Language of publication: English

Abstract

The various aspects of developing full-stack applications using the MERN stack, focusing on core concepts and the justification behind selecting this technology stack. The research investigated the fundamental idea behind MERN, finding how each component contributed to the creation of powerful and dynamic web applications. Analyzed the reasoning behind choosing the MERN stack, examining the unique strengths of MongoDB, Express.js, React.js, and Node.js and how they come together for efficient development. Author discussed the architecture of the application and its various development stages. Additionally, the interaction between technology and business, investigating the business model that aligned with food delivery applications, considering factors such as scalability, user experience, market dynamics and existing food delivery application analysis. Through a combination of theoretical analysis and practical implementation, the aimed was to provide a holistic understanding of full-stack application development, offering valuable insights for both technical experts and business opportunities in the rapidly evolving digital landscape.

The MERN technology stack is an efficient choice for creating a responsive and scalable application. It provides real-time features, an appealing user interface, and strong data security. In the business context, the research showed that having different ways to make money, partnering with restaurants, keeping users engaged, and staying ahead of the competition.

Keywords: Technology stack, Component, Development, Architecture, Scalability, User experience, Technical, Real-time, User interface

Contents

Figures

Tables

Acronyms

MERN	MongoDB, Express.js, React.js, Node.js
UX	User Experience
UI	User Interface
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
JWT	JSON Web Tokens
UCD	User Centered Design
DOM	Document Object Model
NPM	Node Package Manager
JSX	JavaScript XML
SEO	Search Engine Optimization
I/O	Input/ Output
AWS	Amazon Web Services

1 Introduction

In today's world, where technology is changing fast and how people like to eat is changing too, the food industry is going through a big change. The way people used to go to restaurants is now being mixed with digital options. This means people can order food online and have it delivered to them. This change has led to the rise of food delivery apps, which have changed how we order and enjoy our meals. Customers can enjoy many benefits by ordering food through mobile applications, including convenience, time savings, variety of purchasing options, offers from sellers, avoidance of travel, delivery at doorstep, quality guarantee (Reddy & Aradhya, 2020). Figure 01 showing the work flow of food delivery application.

Figure 01 Food delivery application work flow

People nowadays often using apps to order food, so it is important that these apps work smoothly and are easy to use. Developers who can do both the front and back parts of the technology are really important in making these apps successful. They make the screens look nice and easy to use, and they also make sure everything works right in the background. Food delivery apps are always trying to make things better for people. They want to make ordering food easier and faster, and they also want to use new technologies to improve. This thesis looks closely at how these apps are made, from the technology used to the ways they impact people's lives. It talks about different technologies like React, Angular and Vue.js that are used to make the app's screens.

According to the statista report in 2023, React.js and Node.js have been identified as top web frameworks by software developers all over the world. The survey showed that about 42.7 % of respondents said they already use Node.js while 40.6 % were using React. This was based on their responses to the questionnaire. According to the survey results, 42.7 % of respondents said Node.js was in use and 40.6 % indicated that React.js had been used. These results indicate a significant

dominance of these two frameworks in the developer community. The high usage rates of both technologies highlight their importance and relevance in the field of web development on a global scale (Vailshery, 2023).

In today's digitally oriented world, developing web applications can be a great business opportunity. The online restaurant food delivery industry is evolving significantly, introducing innovative methods to connect with end customers. Emerging delivery channels, including new approaches like robots and drones, are reshaping the landscape of delivery methods. Digital platforms, like apps, can be really successful by offering personalized services at good prices. Making user-friendly and creative web apps can solve problems for clients and also make money from subscriptions and updates. It is essential to stay at the leading edge of new technology and develop applications which are consistent with each sector's needs.

2 Research Design

2.1 Research Problem

In the area of food delivery applications, using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack presents an innovative approach. However, despite the growing trend in digitalization and online food ordering platforms, there is a need to examine and analyse specific issues in depth, efficiencies and potential improvements with MERN solutions specific to the field of food delivery. Understanding these challenges and exploring strategies for optimization and innovation within the MERN stack specifically for food delivery applications remains a significant research problem.

Key Components to Investigate:

- **Efficiency and Scalability:** Assess the efficiency and scalability of the MERN stack in handling large-scale data management, real-time order processing, and accommodating an expanding user base within the context of food delivery applications.
- **User Experience (UX):** Explore ways to enhance the user experience using MERN technologies, focusing on intuitive interfaces, efficient navigation, and personalized features for improved customer satisfaction.

- **Security and Privacy:** Investigate the robustness of the MERN stack in implementing stringent security measures to ensure safe transactions, protect user data, and maintain privacy in a food delivery app environment.
- **Integration and Performance:** Analyze the integration capabilities of the MERN stack with external services (payment gateways, maps) and assess its performance under varying load conditions to identify areas for optimization.
- **Competitive Analysis and Innovation:** Conduct a comparative study of existing MERN-based food delivery apps, identifying gaps and opportunities for innovation, new features, or enhancements that can provide a competitive edge.

2.2 Research Questions

The following research questions are expected to be answered by the current research in line with their research objectives:-

1. Why is the MERN stack consider suitable for developing food delivery web applications?
2. What are the key technological considerations and challenges in the development of a food delivery web application that aims to provide a seamless user experience and efficient order management?
3. How can user data be secured in a full stack web application?
4. What are the critical factors to consider when developing a food delivery web application for scalability?
5. What are the key features required for a full-stack food delivery web application?

This research aims to find logical answers to these questions.

3 Literature review

3.1 Introduction

Recently, the food delivery industry has gained immense popularity, especially in cities and urban regions. Technology has played a significant role in transforming how people order and receive food. Cities with a strong food scene and a tech-savvy population provide an ideal environment for the expansion of food delivery apps. To ensure success in this business, it is important to incorporate special features into the app. Instead of relying on a single app, it is recommended to develop apps that cater to the needs of customers, businesses, and delivery agents. By adopting this approach, the likelihood of attaining success in the fiercely competitive food delivery industry is significantly enhanced (Norris, 2020).

According to the research analysis the market for online delivery of foods in Europe is forecast to be worth USD 20.2 billion by 2026. Restaurants and fast-food businesses can grow their customer base affordably by using online food delivery services. At the same time, customers can easily order their preferred meals without the need for significant time or effort (Ray et al., 2019).In the coming years, Finland's food delivery market is set for significant growth. In 2023, the market is expected to generate revenues of around $1.5 billion and an annual growth rate of 8.74% for the period up to 2027. This growth trajectory would lead to a market volume of $2.11 billion by 2027. Notably, the Meal Delivery segment is poised for rapid revenue growth of 17.6% in 2024. Furthermore, by 2023, it is predicted that around 2.46 million people will be using meal delivery services in Finland. This means that about 33.2% of the population will be using these services. These numbers show that online food delivery is becoming more popular and there is a growing demand for it in Finland.

3.2 Evolution of Food Delivery Applications

Post 1990, rapid progress was witnessed in internet and wireless technology, and various tech-based systems were adopted by restaurants to enhance operational efficiency (Alimadadi et al., 2016). The European online food delivery market report states that, over the last five years, a significant increase has been registered in this market for food orders. The increasing number of mobile phone users, as well as the growth in internet penetration that makes online food delivery more accessible, is to be attributed to this increase. The process of placing food orders online in Europe has been substantially transformed due to the COVID-19 pandemic. Food delivery has become a global market worth more than $150 billion, having more than tripled since 2017 (Ahuja, Chandra,

& Peens, 2021). The revenue of the online food delivery market in Europe is projected to witness substantial growth, reaching an estimated 60 billion U.S. dollars from 2017 to 2027 (Beyrouthy, 2023). Figure 02 explaining the report online food delivery market revenue.

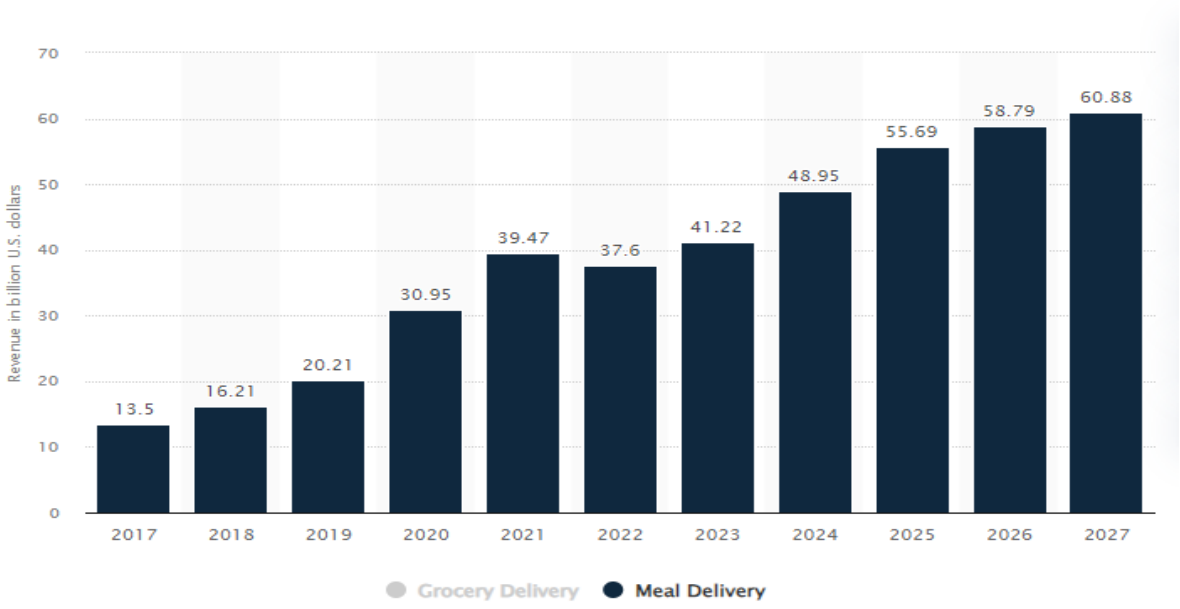

Figure 02 Online food delivery revenue development

Restaurants in Finland are increasingly utilizing food delivery platforms as a means to boost their revenue and expand their customer reach within the food retail industry. The trend of eating outside the home has become popular, leading to rapid changes in the business models of restaurants and cafes, focusing on meal delivery and takeaway. This shift has attracted interest from venture capital investors, as evidenced by Wolt, a food delivery start-up based in Helsinki, which raised €110 million and reached a valuation of approximately half a billion euros last year(Niemistö, 2020).

3.3 Existing Application Analysis Case Study Wolt

3.3.1 Introduction

In this case study, an analysis was carried out on the existing food delivery application, Wolt, situated in Helsinki, Finland. The examination encompassed various aspects of Wolt's operations, such as its user interface, user experience, delivery procedures, restaurant collaborations, payment methodologies, and customer support. By examining these elements, particularly in the context of Helsinki, the aim was to gain an understanding of the application's strengths, weaknesses, and opportunities for improvement, both in its functioning and on a global level. The insights gained from

this analysis played a crucial role in enhancing Wolt's services, allowing them to become more efficient and customer-oriented. Additionally, it served as a valuable resource to ensure the proper design, user-friendliness, and adaptability to users' needs and preferences when developing the food delivery application. By leveraging the knowledge obtained from the analysis, the app was tailored to provide a seamless and enjoyable food delivery experience, making it a competitive player in the market. These steps were taken when developing an application for food delivery and analysing current food delivery applications such as Wolt in Finland.

3.3.2 Market Research

Marketing is a way for companies or individuals to promote their products or services and encourage people to become loyal customers. It plays a vital role in introducing a company's products to customers and shaping the company's desired image. Finding and retaining new customers through knowing their needs and delivering a good customer experience is the main objective of marketing. Different strategies are used to achieve these objectives, such as personalized offerings and creating a satisfying customer journey.

An in depth analysis of the marketing strategies used by Wolt and its competitors in the food delivery sector is carried out in this research. The focus will be on understanding how these companies attract and retain customers through various means, such as promotional offers, loyalty programs, and referral systems. Furthermore, in this study, a specific marketing approach employed by Wolt was examined. Orders made by new users who had registered with the Wolt service for a period of 14 days after registration were exempt from delivery fees. (Wolt, 2023).

By conducting a comprehensive investigation of existing food delivery applications like Wolt in Finland, this research seeks to provide valuable insights for developing a successful and competitive food delivery app tailored to the Finnish market. The information collected will help make smart choices to improve how these apps work. This will lead to the food delivery business growing and staying successful in the long run.

3.3.3 In Finland, Online food delivery booking by brand from March 2023

A survey was carried out in 2023 to ascertain which eating delivery apps were used last year by consumers in Finland which is shown in figure 03. According to the survey, 64 % of the respondents used Foodora, while 61% used Wolt. The survey was done online with 844 consumers in Finland and is considered representative of the overall population (Kunst, 2023).

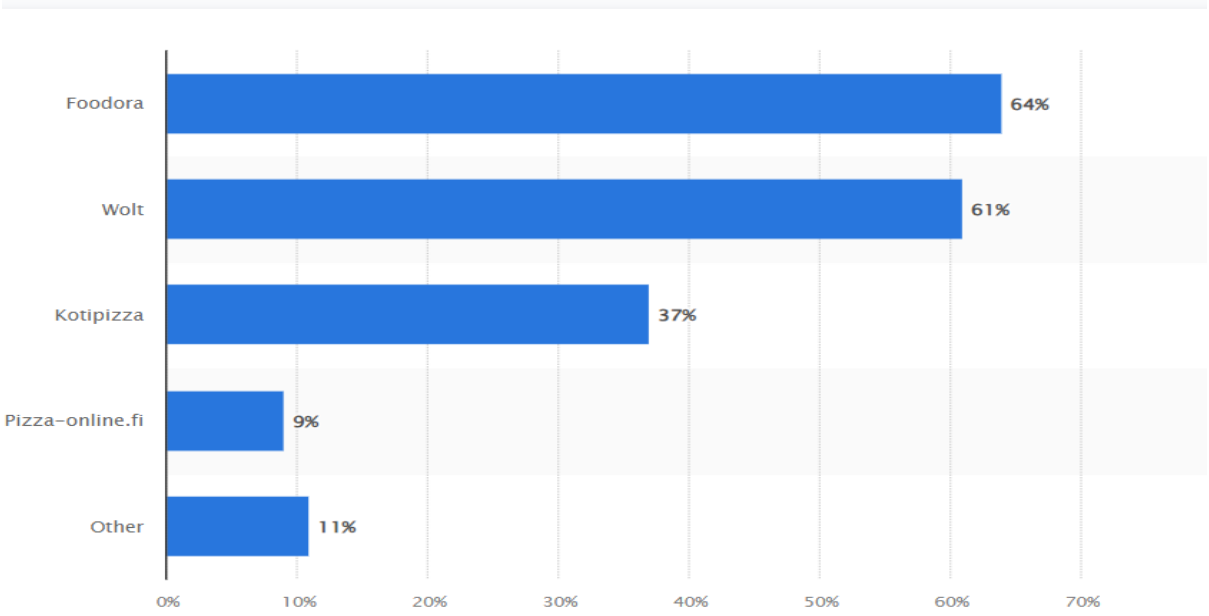

Figure 03 Online food delivery brand survey

3.3.4 Features and Functionality

Wolt Market is an online platform that connects customers with local grocery stores, providing a wide range of products for home delivery. The website allows customers to explore different categories, making it easy to use, search for items and add products to their virtual shopping carts. Wolt Market focuses on offering high-quality goods from trusted suppliers and retailers to ensure freshness and customer happiness.

One of Wolt market's standout features is its efficient delivery system. With the help of Wolt's extensive delivery network and advanced routing algorithms, the platform aims to provide fast and reliable delivery services. Wolt takes pride in delivering orders within 30 minutes and offers live tracking from the time of order to delivery. Customers can schedule delivery windows at their convenience, and Wolt Market strives to meet these timelines, ensuring even greater customer satisfaction (Wolt, 2023).

3.3.5 Partner Restaurants and Cuisine

Restaurants can join Wolt's platform without paying any fees, and if they wish to stop using the service, they can do so right away. Wolt also offers free promotion for new restaurants on its main webpage. Because of this, many restaurants are becoming interested in joining Wolt, and this is helping Wolt expand more quickly (Pärli, 2020).

In partnership with Wolt, we have introduced a brand new market for quick grocery deliveries in Finland. We have teamed up with K-Market, which has a wide network of local stores across the country, to offer fast deliveries at 40 locations. Our customers love the service, and the demand is increasing rapidly. The online grocery sales in Finland are expected to grow considerably in the coming years. As pioneers in this field, we are committed to continuously improving our online grocery services. Our collaboration with Wolt has been a success and is set to grow even more in the future (Akseli, 2022).

Head of Restaurant Operations, Pizzeria Reenrol, Antti Kuitunen, expressed that without Wolt, they would have had to come up with their own costly and labor-intensive shipping method. Fortunately, Wolt provided a solution that not only worked for them but also made life easier for their employees.

According to Head of Restaurant Operations, Sushibar and Wine, Kaisa Laulajainen, Wolt has been instrumental in attracting a whole new group of customers. The app's user-friendliness has made it incredibly easy to use. Working with Wolt has been a success for them, resulting in a significant increase in sales.

Can Mese, the owner of Topcapi for 27 years, shared that they had never provided home delivery before Wolt started operating in Hämeenlinna in the fall of 2019. Working with Wolt turned out to be a huge success as it brought in additional business and proved to be highly profitable for them.

3.3.6 Delivery Efficiency

Courier partners play a crucial role in delivering food to customers safely and on time. Wolt relies on these courier partners to handle all the delivery orders. These partners are not Wolt's employees; instead, they are independent parties who have a contract with Wolt, just like restaurant partners do. Wolt can only end the contract if there's a breach of the agreement, Wolt's courier partners enjoy the freedom to terminate their contract at their discretion. They have the flexibility to decide their working hours without any minimum requirements, allowing them to work according to their preferences. This flexible income option is particularly beneficial for various individuals, including students, as they can choose to work as much or as little as they desire and start and stop whenever needed (Pärli, 2020).

Working with Wolt gives couriers the freedom to decide when they work, so they can control how much money they make. They have a flexible schedule that allows them to earn a good income

without disrupting their daily life. Wolt also provides insurance that covers accidents while working. All these great benefits make working with Wolt a smart and practical choice for those who want to earn a living with flexibility (Interviewee, 2023).

3.3.7 Technology stack of Wolt app

Research the technology stack used by Wolt and other food delivery apps. This includes the programming languages, frameworks, and third-party integrations. Understanding the technology behind these apps can guide in development. Table 01 describe the WoltModalSheet features and functionality.

WoltModalSheet features	Description
Responsive design	Adapts seamlessly to various screen sizes, functioning as a dialog on larger screens and transforming into a bottom sheet on smaller screens for optimal user experience.
Multiple page functionality	Allows integration of multiple pages within the sheet, offering users a seamless and step-by-step flow without losing context.
Motion animation	Boasts captivating motion animations, including smooth page transitions and scrolling.
Scrollable content	Ensures easy viewing of all supplied information by enabling scrolling on each page within the window.
Customizable Appearance	Provides developers with customization options to match the modal's appearance with their application's branding and design.
State management	Seamlessly integrates popular state administration libraries into modal sheet pages for effective development and management.
Open source	An open-source library that encourages active developer contribution, bug fixes, and proposals for new features, fostering a collaborative and community-driven environment.

Table 01 WoltModalSheet features and description

3.4 Technological Frameworks for Full-stack web Applications

Selection the best technological frameworks depends on things like how skilled the developers are, how much the application needs to grow, and what exactly the food delivery app requires. The advantage of full stack development is the ability to select preferred components for the stack, based on the best tools for the job and in a fast-paced environment, all specific recommendations are capable of being updated quickly. (Singh et al., 2022) explain the term full stack can refer to a layer of web development that encompasses both the front-end and back-end components of an application. The table 02 provides a brief description of each framework and its primary use.

Front-end Framework	Description	Primary use
React.js	A JavaScript library for building user interfaces	Building interactive and dynamic user interfaces
Angular.js	A TypeScript-based web application framework	Developing robust, scalable, and dynamic apps
Vue.js	A progressive JavaScript framework for UI development	Building interactive and modular user interfaces
Back-end framework		
Node.js	A JavaScript runtime for server-side development	Building scalable and efficient server applications
Express.js	A minimal and flexible Node.js web application framework	Simplifying server-side development in Node.js
Django	A python web framework with advanced features	Fast development of secure and easily maintainable applications
Ruby on Rails	A web application framework written in Ruby	Streamlining the development of web applications
Database		
MongoDB	A NoSQL document database.	Storing and retrieving data in a flexible JSON-like format.
PostgreSQL	An open-source relational database.	Managing structured data in a scalable manner.
MySQL	An open-source relational database management system.	Handling relational databases efficiently.

Full-Stack		
MEAN Stack	MongoDB, Express.js, Angular, Node.js.	Full-stack JavaScript framework.
MERN Stack	MongoDB, Express.js, React, Node.js.	Full-stack JavaScript framework with React.
LAMP stack	Linux, Apache, MySQL, PHP/Python/Perl	Classic full-stack solution.

Table 02 technologies used in full-stack app development

Software developers and companies continuously refine their web development stacks, drawing from market trends and their own expertise, to enhance software development processes and deliver exceptional web applications. The MERN stack is a collection of various tools used to make modern web apps. It is like a complete framework for creating interactive and efficient applications using JavaScript. Node and Express connect the back-end of the application, MongoDB is used for storing data, and React builds the user interface that people see and use (Hoque, 2020). The MERN stack's conceptual blocks are illustrated in the following figure 04.

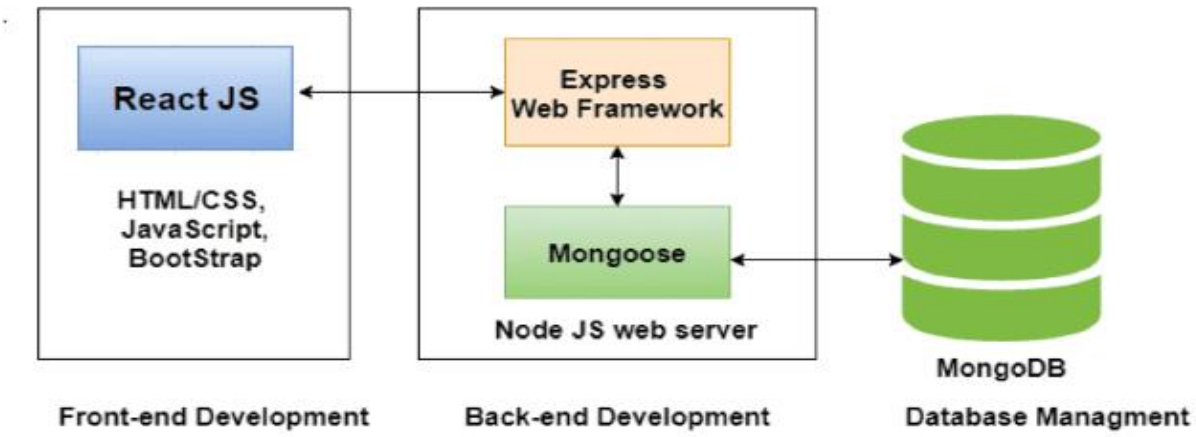

Figure 04 MERN stack development

3.5 System Architecture and Component Design

System architecture is the structure of a software system at its very highest level and organization, outlining how various components and modules interact and work together to achieve the desired functionality. It defines the overall layout, communication patterns, and dependencies of the system. In the context of a full stack food delivery web application, system architecture includes

both the frontend and backend components, as well as any external services or databases that are part of the application.

On the other hand, it devotes itself to detailed design of individual components or modules within a system. It involves outlining specific responsibilities, interfaces, and interactions for each component to ensure their independent development, testing, and maintenance capabilities. Component design considers factors such as code organization, modularity, reusability, and extensibility. In full-stack application development, component design is the process of separating an entire application from smaller, reconfigurable modules or components. They work together to create a comprehensive and efficient food delivery application.

Within the framework of a full-stack application, the following are some common components. Figure 05 showing the essential components of full-stack food delivery application.

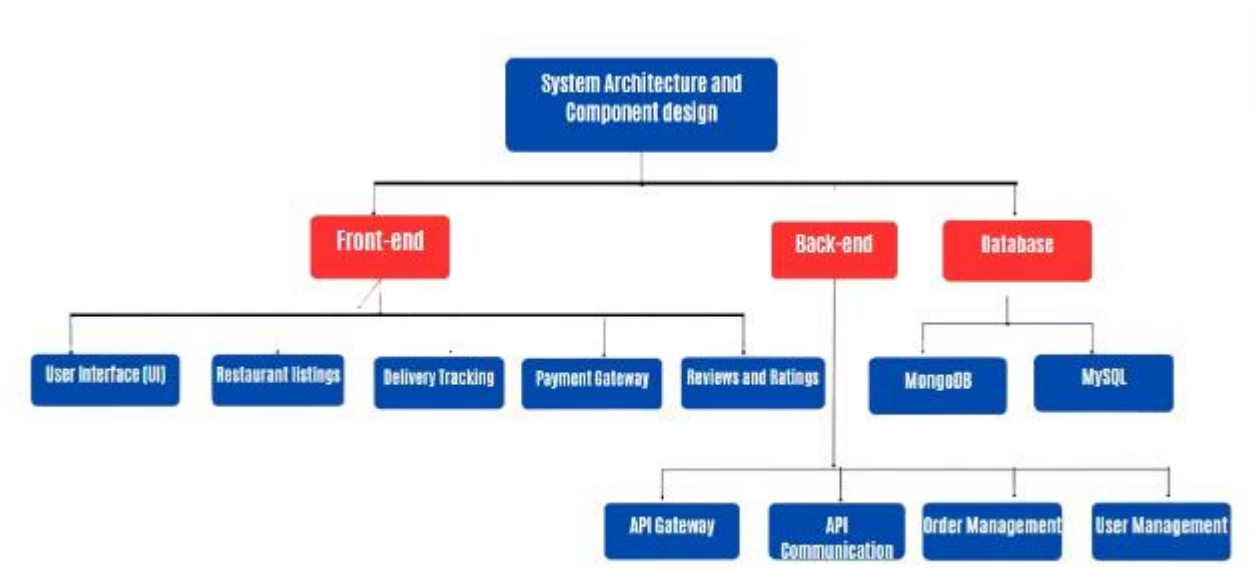

Figure 05 Components of Full-stack application

3.6 Front-end Components

3.6.1 User Interface (UI)

User Interface in a full-stack web application refers to the Visual and interaction components that enable users to interact with the application. This includes user interface design, layout and presentation of different components which make up a user interface such as menus, buttons, forms and other graphical elements. The UI is a critical aspect of a full stack web application as it directly

impacts the user experience and usability. It serves as the bridge between the user and the application's functionalities, allowing users to navigate through the application, input data, and receive feedback. User interaction with the system is easier and more satisfying for users thanks to a well-planned user interface, which results in higher efficiency. It does this by offering easy-to-understand and visually attractive designs.

3.6.2 Restaurant listings

The restaurant listings component, located on the front-end of a food delivery application, is a feature that directly interacts with users. Its main role is to present a list of restaurants that are currently available to the users. This component provides a user-friendly interface where customers can browse through various restaurant options and view essential details about each establishment. The restaurant listings component plays a pivotal role in helping users discover various dining options and make informed decisions while placing their food orders. It serves as the initial step in the user's journey towards selecting a restaurant and proceeding with the order placement process in the food delivery application.

3.6.3 Delivery Tracking

The user friendly tracking feature enables customers to monitor their orders progress in real time within the frontend of a web application. This feature begins its updates from the moment the order is confirmed and continues until the order is successfully delivered to the customer's doorstep. This component provides users with valuable information and updates, enhancing their overall delivery experience. The delivery tracking component plays an important role in keeping users informed and engaged throughout the delivery process, improving customer satisfaction and confidence in the food delivery application. By offering real-time tracking and status updates, this component enhances the overall user experience and builds trust between the app and its customers.

3.6.4 Payment Gateway

The payment gateway component in a food delivery application is a user-friendly feature that facilitates secure and smooth online payment processing. It allows users to make payments for their food orders using various payment methods, ensuring a hassle-free transaction experience. Support for more than one payment option such as credit card, debit card, mobile wallet and online banking to provide users with a wide choice of payment options is an important function of the Payment Gateway component. The component ensures the security of sensitive payment information

through encryption and verification, minimizing the risk of errors during transactions. It calculates the total amount to be paid based on the selected items, taxes, and delivery charges, providing users with accurate payment details.

3.6.5 Reviews and Ratings

The reviews and ratings component in a food delivery application allows users to share feedback and rate restaurants and their food orders. Users can write detailed reviews about their dining experiences, rate restaurants using a star system, and sort reviews based on various criteria. The component promotes transparency by displaying user profiles alongside their reviews and may allow users to edit or delete their feedback. Some applications enable restaurant responses to reviews, facilitating communication. This component enhances user decision-making, encourages restaurant improvement based on feedback, and fosters a sense of community. The system works together with the backend database in order to store and retrieve data concerning reviews, contributing to a higher level of user engagement and satisfaction.

3.7 Back-end Components

3.7.1 API Gateway

API (Application Programming Interface) communication is a crucial aspect of full stack web application development that enables different components of the application to interact and exchange data. The API enables seamless communication between the frontend client side component and the backend server components. In a full stack web application, the frontend communicates with the backend through APIs to request and retrieve data. The backend responds to these requests over HTTP or other protocols, commonly using REST (Representational State Transfer) principles or GraphQL.

3.7.2 API Communication

Provides a standardized and structured way for different parts of the application to exchange information. It allows the frontend to send requests for specific data or perform actions, such as submitting forms or updating information. The backend will process this request, retrieve or modify the data and send a response back to the frontend. API communication is essential for implementing features like user authentication, data retrieval, data modification, and integration with external services. It enables the frontend to interact with the backend services, retrieve data from databases, and perform business logic operations.

3.7.3 Order Management

The order management component in the application is a critical system responsible for handling all aspects of the order lifecycle behind the scenes. It manages and coordinates the flow of information related to customer orders, restaurants, and delivery personnel. Key functionalities include order processing, confirmation, delivery assignment, real-time tracking, status updates, payment processing, order cancellations and refunds, order history maintenance and delivery personnel communication. This component ensures smooth and efficient order processing, providing a seamless experience for customers and restaurants. Collaborating with the front-end order management system, it offers a comprehensive end-to-end solution for handling orders in the food delivery application.

3.7.4 User Management

The user management backend component in a food delivery application is essential for handling user accounts and profiles. It enables seamless user registration, authentication, and account management. Key functionalities include user registration, secure authentication, password recovery, account information management, and account security. This component also maintains user authentication states, manages user profiles, and may implement user roles and permissions. The user management backend component collaborates with the front-end system to offer a comprehensive user management solution and enhance user satisfaction with the food delivery platform.

3.8 Database Component

In a food delivery application, a database plays a crucial role as a structured collection of data, organized and managed for easy retrieval and manipulation. It acts as a central repository for various types of information related to the application. The database stores user data for managing accounts, authentication, and access control. It also holds details about restaurants, menus, and menu items, facilitating efficient restaurant and menu management. When customers place orders, the database records order details and aids in order processing and delivery tracking in real-time. The database forms the backbone of food delivery application, ensuring data reliability and efficient access, leading to a seamless and user-friendly experience for both customers and restaurant owners in managing orders and deliveries.

4 Methodology

4.1 Research Methods

Research methodology is a way of doing research that is organized and follows scientific steps. It helps gather, analyze, and understand data to answer research questions. Research methodology consists of a systematic and organized approach for the design, conduct or analysis of research. Research methodology outlines the framework for data collection, data analysis, and interpretation, while also addressing ethical considerations, constraints, and potential challenges that may arise during the research process (Dawson, 2009).

The selection of a research approach or data collection method should base on particular type of data (quantitative, qualitative, or mixed of both) that is necessary and feasible to acquire. Fundamentally, qualitative and quantitative approaches represent the two primary methods to gather data or conduct research (Naoum, 2007). The method chosen is dependent on research questions, available resources and study objectives. The choice of research method for a web application thesis depends on the specific research objectives, available resources, and constraints. There are three main types of interview method used in qualitative research, structured interviews, semi-structured interviews, and unstructured interviews (Islam & Aldaihani, 2022).

Mwita (2022) provides an overview of various interview types, in structured interview the researcher uses a set list of questions to ask respondent. Researcher do not allow to ask additional questions to get more information from the interviewee. In semi-structured interviews, researchers prepare a list of questions in advance to ask during the interview. These questions guide the conversation, but there is also flexibility to include additional questions based on the responses given by the interviewees. In unstructured interviews, researchers begin the interview session without any predefined questions. Questions are asked spontaneously based on the researcher's own choice. Considering the research questions and desired outcomes, this study used a qualitative approach.

4.2 Data Collection

When target was set, the data collection process started collecting information for the evaluating the MERN Stack in the development of Food Delivery applications. The qualitative research method was chosen as the foundation for studying the online food delivery application. This approach involves talking to people and getting their opinions through interviews. The interview is a process in which

researchers ask interview participants a variety of questions in an attempt to obtain information about a particular topic from the interviewer (Adhabi & Anozie, 2017). The interviews offer researchers valuable, detailed qualitative data that help in understanding the experiences of the participants, their descriptions of those experiences, and the meanings they give to those experiences (Rubin & Rubin, 2012). For this research, Interviews were conducted with IT professionals, UI/UX designers, front-end and back-end developers, and C-Suite professionals possessing analogous experience. This approach was suitable as it enabled the collection of comprehensive insights from professionals. Their experiences and perspectives provide valuable information to improve the online food delivery application and make it more user-friendly.

For a number of reasons, the interview method has great value in research:

- Interviews provide in-depth and detailed information, enabling researchers to thoroughly comprehend the viewpoints of participants.
- Depending on the context and response received, researchers can adjust their questions and approach leading to more detailed analysis.
- Researcher interviewed with IT professionals, UI/UX designers, and architects who had similar experiences, they got truthful and accurate answers.
- Interviews give participants the opportunity to share their experiences and viewpoints directly.

The interviews took place online via Team software and Google meet, engaging. Participants' preferences for anonymity were honored and respected throughout the process. In total, seven interviews were undertaken as part of this research effort. The interviews aimed to gather valuable insights and real-world data essential for shaping the research findings. The main goal of these interviews is to learn from different experts. Understanding the technical functioning of the app, like how it is built and how it works for users, as well as learning effective strategies for drawing in users and restaurants while maintaining competitiveness. The discussion revolved around project management, including the timeline and methodology would use. This information serves as a crucial foundation for developing food delivery application.

4.3 Interview Data Analysis and Results

A primary method employed to gather information for thesis involved conducting interviews with experts. Since several of these experts were located in different places and could not meet in person, we used online tools such as Team software and Google meet for interviews. Each chat session lasted between 30 minutes to an hour, the same types of questions were asked to everyone to ensure a straightforward comparison of their answers. It is interesting that these experts come from different parts of the world. Researcher was in Finland, but the interviewees were in Finland, India, and Pakistan. Interacting with individuals from diverse locations contributed to an enhanced understanding of various topics.

The professional background of the respondents and information about their respective organizations are detailed in Appendix 01. The selection of these individuals was based on their extensive expertise and prolonged experience within their respective domains. They displayed amazing professionalism and responded comprehensively to all inquiries. These interactions provided different perspectives and valuable insights. Conversations were conducted with a range of experts, including IT professionals, designers, developers, and high-level executives. The interview questions are outlined in appendix 02 for reference. Table 03, presented below interview details.

Number	Date	Time	Designation	Company
1	19.09.2023	30	UI/UX Designer	Indicsoft technologies
2	19.09.2023	30	Front-end Developer	Indicsoft technologies
3	19.09.2023	30	Back-end Developer	Indicsoft technologies
4	09.10.2023	30	Project Manager	BaseMark Oy
5	03.10.2023	35	Developer	Cloud Value Tree Oy
6	20.10.2023	35	Sales head	FreskFoods Oy
7	01.11.2023	40	CTO	CloudKraft

Table 03 Expert´s Interview details

During this research, interviews were conducted with seven professionals, each contributing a distinct set of skills and experiences.

A respondent, with 17 years of experience in the IT industry, offered valuable insights during the interview. With a robust background collaborating with multiple IT firms in Finland, this individual currently serves as a Project Manager. Their extensive experience in the IT sector made them an excellent candidate for the interview, providing valuable insights and knowledge to enhance my research.

Another respondent is a proficient UI/UX designer, able to create user interfaces that are simple and visually appealing. Their expertise in UI design and development significantly contributed to the research. The interview covered a range of topics, such as the food delivery application's scope, technology stack, its potential within the Finnish market, and the specifics of its development.

In the interview with an experienced UI/UX designer, the focus was on the design approach used to ensure the food delivery app's usability for both new and frequent users. In response, she highlighted her commitment to creating a simple and intuitive navigation system, so first-time users could quickly figure out how to order food. At the same time, emphasis was providing personalized features for frequent users. The discussion also covered methods used to improve the visual appeal of application. This statement provided detailed insight into the careful design choices made to create an elegant user interface. Furthermore, insights were provided on enhancements to the checkout and payment process. A one-click payment option for returning users was introduced to expedite and simplify the process, ensuring quick and convenient transactions. The discussion revealed the thoughtful design decisions that make the food delivery app user-friendly and visually appealing.

The analysis of the content from the UI/UX designer interview revealed the primary insights, which can be summarized as below:

- **UI/UX designer aimed for easy app usage:** Focused on creating simple navigation for new users and added personalized features for frequent users.
- **Emphasis on visual appeal:** Careful design choices were explained, ensuring a beautiful and attractive interface for the app.
- **Streamlined checkout process:** Introduced a quick one-click payment option for returning users, enhancing convenience during checkout.

In a conversation with an experienced front-end and back-end developers, we discussed how the application is structured and how it works. One important topic was about the architecture, specifically how the frontend and backend of the app communicate. The developer explained that the app follows a client-server architecture. The front-end, which is built with React, talks to the back-end through API calls. They highlighted the use of RESTful API endpoints, which act as bridges for data exchange between the front-end and back-end. This ensures that data flows smoothly between the two parts of the app. Back-end development is an important part of software creation that focuses on creating and maintaining the hidden parts of an application. It handles the server-side logic, database, and infrastructure, making sure everything runs smoothly behind the scenes while users interact with the visible parts of app. Backend developers create the rules and processes that control how the server manages requests from the app's frontend. Manage database and make sure data is organized, updated, and secure. Their responsibilities include designing the functionality of the app, managing the database efficiently, creating interfaces for communication, protecting sensitive data, configuring servers, testing the backend, and ensuring smooth collaboration. The discussion covered the methods for managing user authentication and authorization on the front-end. The developer referred to the use of JSON Web Tokens (JWT) for managing this aspect.

From interviews with front-end and back-end developers, here are the key points to summarize.

- Discussion on client-server architecture, the React based front-end communicates with the backend via a RESTful API. This enables smooth and continuous data transmission between both parts of the application.
- Importance of backend development in managing server logic, databases, and infrastructure for smooth operation while users interact with the visible app parts.
- Mention of using JSON Web Tokens (JWT) for managing user authentication and authorization on the frontend, enhancing security measures within the application.

Conversations with the sales head revealed strategies to attract both users and restaurants to the application, ensuring competitiveness in the food delivery market. Lastly, the project manager shared valuable insights into project timeline management and the methodologies and frameworks employed to ensure an efficient development process.

These interviews collectively provided a holistic understanding of the multifaceted aspects involved in the development and successful launch of a full-stack food delivery application.

5 Development Phase of Application

5.1 Front-end Development

5.1.1 UI/UX Design

User Interface (UI) and User Experience (UX) design, coupled with frontend development, form the cornerstone of modern web and application development. This interdisciplinary field encompasses the art of crafting visually appealing, intuitive, and user-centric digital interfaces while ensuring the seamless functionality of these interfaces on the web. Figure 06 describe the design interface of UI/UX design.

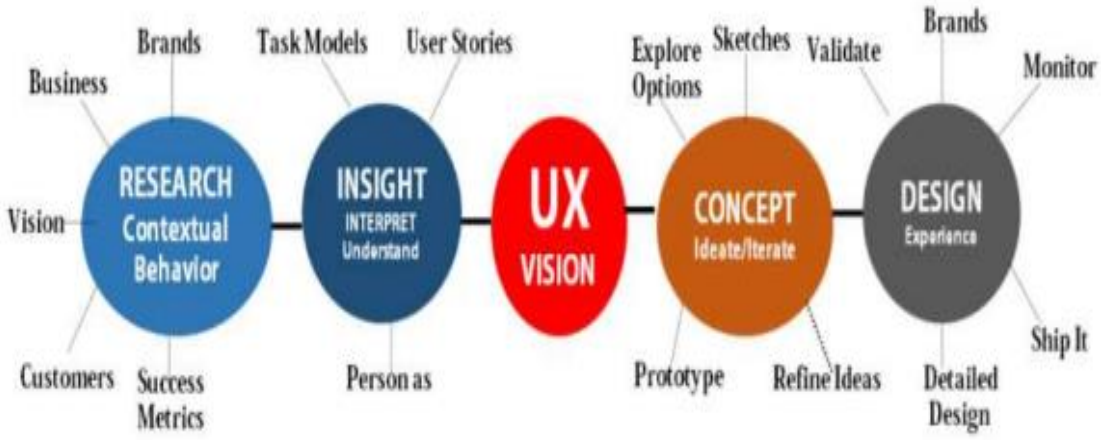

Figure 06 UI/UX design interface

A User Interface (UI) is the point of interaction between a user and a system which involves commands or methods for operating the system, inputting data, and accessing content (Joo, 2017). UI design revolves around creating the visual appearance and user experience of digital products, emphasizing visual aspects like layout, colors, typography, icons, and interactive features. UI designers aim to find a harmony between visual appeal and practicality, ensuring that users can effortlessly navigate and interact with the application.

User Experience (UX) Design is the practice of creating physical or digital products that are not only useful but also easy to use, offering a delightful interaction experience (Canziba, 2018). This involves various activities such as user research, information architecture, wire framing, prototyping, and usability testing. UX designers aim to understand user needs, behaviors, and challenges to create intuitive, efficient, and enjoyable experiences. Working closely with UI designers is a common

practice to guarantee a smooth alignment between the visual and experiential elements of the interface. Figure 07 smartly describe the difference between the UI/UX design.

Figure 07 Difference between a UI and UX design

The following key concepts in UI/UX design are given below:

- **User-Centered Design(UCD):** User centered design is an approach to design that focuses on user needs and preferences in the design process. It involves rigorous user research, the creation of user personas, and iterative design with user feedback. UCD ensures that designers understand users' motivations and challenges, resulting in user-friendly and efficient products. Accessibility and ethical considerations are also part of UCD, promoting the creation of enjoyable experiences by empathizing with users' perspectives.
- **Information Architecture:** Information architecture is the organization and structuring of information in digital environments for the purpose of enhancing usability and user experience. It functions as a blueprint, determining how data is categorized, labeled, and presented to users. Much like architectural plans for a building, IA establishes content hierarchy, navigation paths, and labeling systems to ensure that users can easily access and comprehend information. IA is vital for user-centered design, improving the efficiency and accessibility of digital systems while reducing user confusion. It encompasses principles of organization, search ability, and user-friendliness, contributing to the success of websites, applications, and other information-rich platforms.
- **Responsive Design**: Responsive web design is a method in web development ensuring a smooth and user-centric experience across different devices and screen sizes. It involve developing applications or websites to automatically adapt and reorganize their layout and

content according to the user's device, whether it is a desktop, tablet, or smartphone. The aim is to offer the best usability and readability, regardless of the screen size, using flexible grids, media queries, and fluid design elements. In multi-device world, responsive design is crucial as it enables businesses and developers to give a regular and enjoyable user experience across various platforms, ultimately enhancing accessibility and user satisfaction.

- **Interaction Design**: Interaction design is a branch of user experience design focused on creating intuitive and engaging user interactions with digital products and interfaces. It involves designing elements, behaviors, and functionalities that enable users to interact effectively and seamlessly with a product. Interaction design ensures that user actions, such as clicking buttons, filling forms, and navigating menus, are easy, efficient, and satisfying. It aims to optimize the user's experience by considering their needs, preferences, and the flow of their interactions within the digital environment.
- **Prototyping:** In the area of design and product development, prototypes are an essential technique that involves creating a simplified, preliminary version of a product, system, or interface. It serves as a visual and often interactive representation of the final product's key features and functionalities. Prototypes are used for various purposes, including testing ideas, validating concepts, and gathering feedback from stakeholders or users. They help designers, developers, and project teams visualize and refine their ideas, identify potential issues early in the design process, and make informed decisions. Prototyping can range from low-fidelity sketches or paper mock-ups to high-fidelity interactive digital models, depending on the project's needs and objectives.

5.1.2 React.js for Front-end Development

React.js commonly known as react, is an open-source JavaScript library used for building user interfaces (UI) or user interface components for web applications. Developed by Facebook, react allows developers to create reusable UI components that manage their own state and efficiently update and render components when data changes. React has become popular mainly due to its declarative and component-based nature. Web developers were taught for years to create HTML, JavaScript, and CSS separately. React recommends writing HTML and CSS in JavaScript (Chen, Thaduri, & Ballamudi, 2019). React is a popular choice for building single-page applications due to its effectiveness in managing complex and frequently changing data. Additionally, react can be used to support native mobile applications using react native. It enables the creation of native mobile

apps for iOS and android platforms using a single codebase. React native allows developers to become familiar with react syntax for building mobile applications. This approach streamlines development by reusing components across various platforms, enhancing efficiency in the development process. This framework provides the ability to develop high-quality mobile applications with a native look and feel, while leveraging the benefits of React's component-based architecture. Figure 08 showing number of downloads of popular front-end frameworks updated on npm trends website (2023).

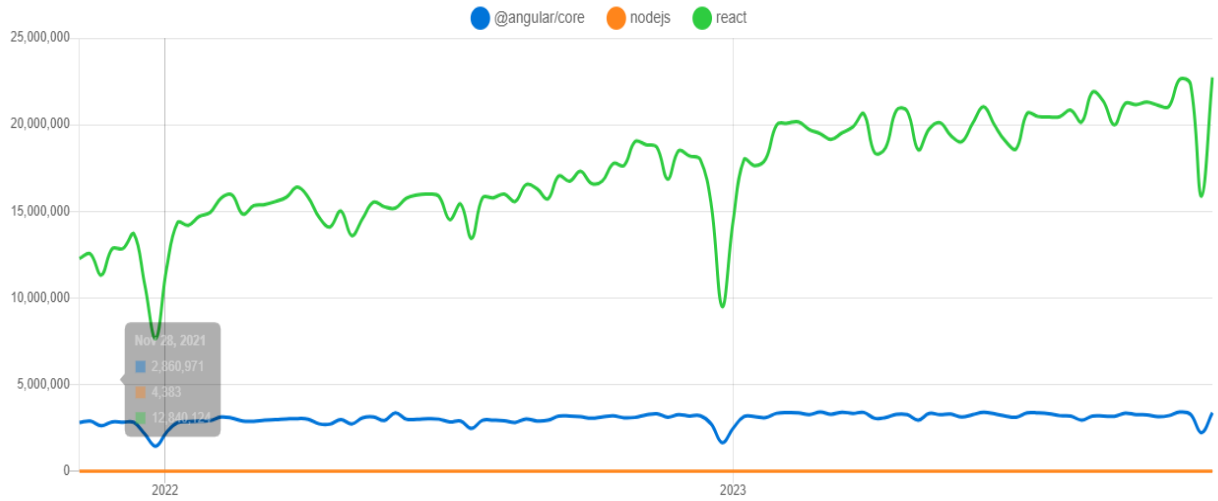

Figure 08 Popular front-end frameworks

5.1.3 React Components

Components serve as fundamental elements in react, functioning as independent and reusable pieces of code that form the foundation of React applications. They allows developers to break down the user interface into more manageable elements, allowing for individual consideration. Data can be conveyed to components through "props" objects, enabling dynamic display of elements in the Document Object Model (DOM). Unlike "props," which are external inputs, a "state" object is managed internally within each component. In React, two types of components exist, each serving distinct purposes in the development process.

1. Functional Components: JavaScript functions are essentially the functional components of react. A Javascript function must be created in react to build the functionality component, and this function may or may not receive data as parameters. The popularity of functional components in react development has grown because of their simplicity and ease of use. Figure 09 explain the syntax for functional component in react.

```
import React from "react";

const FunctionComponent = function(props) {
  return (
    <div>
      <p>This is function based component</p>
    </div>
  );
};

export default FunctionComponent;
```

Figure 09 Functional component

2. Class Components: In React, class components tend to be more complex than functional components. They operate independently and don't have an awareness of other components in the program. However, class components have the capability to collaborate with each other. They allow the transfer of data from one class component to another, can work with each other. Additionally, class components come with lifecycle methods that enable the development of more complex behaviors in the application. Figure 10 explain the syntax for class component in react.

```
import React, { Component } from "react";

class ClassComponent extends Component {
  render() {
    return <div>This is simple class based component</div>;
  }
}

export default ClassComponent;
```

Figure 10 Class components

5.1.4 Virtual DOM

The DOM stands for Document Object Model. It is a functional layout that shows all HTML elements on the web page. The DOM essentially represents the complete user interface (UI) of a web application. The DOM represents a data structure as a tree, each element in the UI is represented by a node. It allows developers to change content in JavaScript, which is very useful. Its structured format proves beneficial as it allows developers to choose specific targets, making the code much more manageable and easier to work with.

The react application uses a virtual DOM, which behaves like a functional replica of the actual DOM and basically it is an interactive representation of the DOM. Thus, for every object present in the original DOM, react creates a corresponding object in its virtual DOM. In some respects virtual DOM is similar to the real DOM, but it does not allow for direct modification of document layout. Since changing the DOM can take too long, it is quicker to change a virtual DOM that does not render anything on screen. It means that the virtual DOM is updated first rather than the actual DOM every time an application change occurs.

A virtual DOM is created when new data is added to an application which presented in the form of a tree structure. Each element within the application corresponds to a node within this tree. So, every time a change is made in an element's state the new virtual DOM tree gets created and compares it to previous one and making notes on that change. The best way to make these changes to the real DOM will be found, and only the updated elements will be displayed on the page again. Figure 11 showing difference between DOM and virtual DOM.

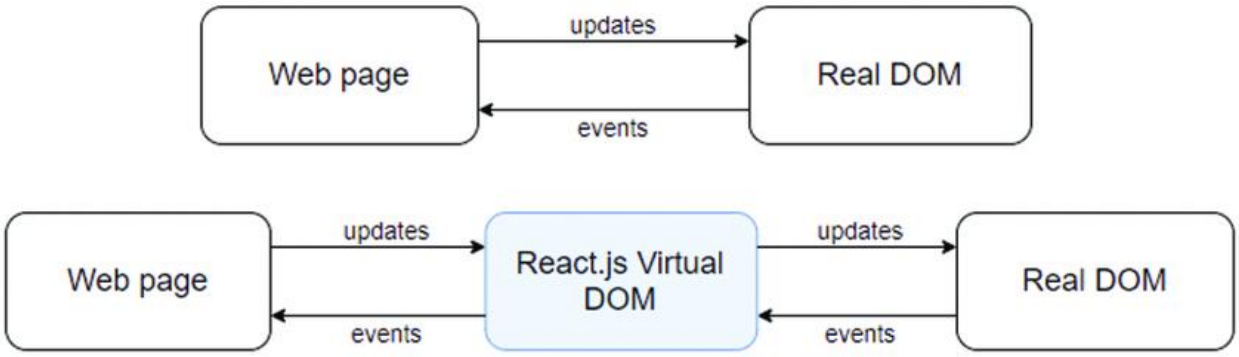

Figure 11 Real DOM and react virtual DOM

The idea behind the virtual DOM theory is that making changes directly to the real DOM is slow and not very efficient Instead of directly changing the real DOM, react create a virtual version of the DOM in the memory and updates this representation when changes made. After updating the virtual DOM, React compares it to the earlier version of the virtual DOM. It figures out the smallest set of changes required to update the real DOM. This process, called reconciliation, is faster and more effective than modifying the whole DOM whenever a change occurs.

5.1.5 Create-react-app

Create react app is a useful tool that helps to start new react application using node.js. This tool comes with useful features like speeding up development. It also provides tools for creating a final product. Among other useful features, it has the package manager, bundler and compiler. to use create-react-app, node and its package manager NPM need to be installed.

A compiler is needed to convert JSX to JavaScript in order that Node and web browsers can be able to work with each other, as the V8 JavaScript engine does not run syntax like JSX. Create-react-app includes a bundle that packages and minifies all application assets, such as node_modules and required JavaScript files called Webpack. Webpack is configured using the webpack.config.js file, which specifies the entry points for bundled files and creates bundled files. Additionally, the development server that hosts the application locally is also configured in the configuration file using the web packdevserver library. For additional functionality, the Webpack plugin can be used.

In the package.json configuration file figure 12, there is a startup script to start the webpack development server and listen for changes. By running the startup script, the application is hosted locally using the port specified in the webpack configuration and can be checked in the browser at http://localhost:port. Package.json also includes a build script to package and minimizes the application source files into a build directory ready to be hosted on the web server for production. The third script available is a test script that triggers the test runner to search for files with the .test.js extension.

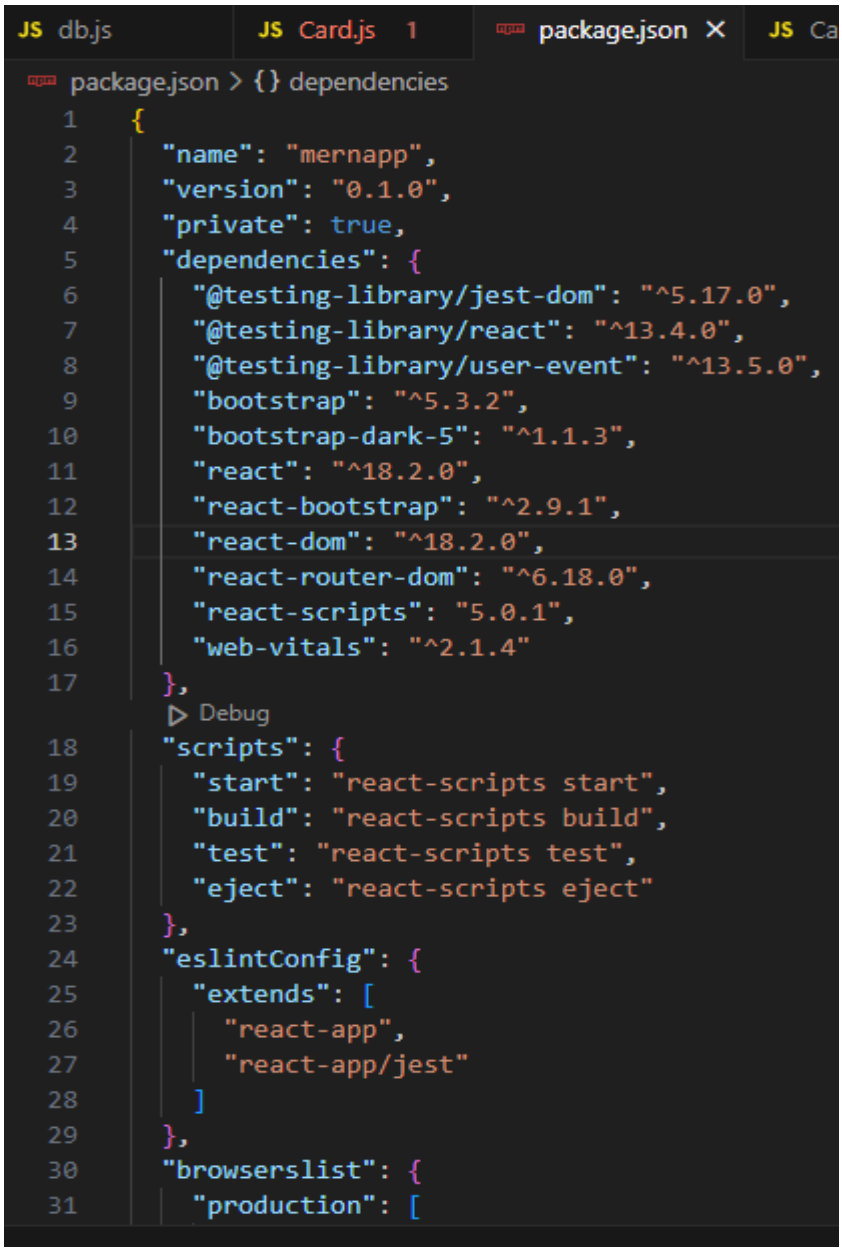

Figure 12 The Packaje.json configuration file

The following directories are present in the react application generated.

- **Node_modules:** Provides all required dependencies that are needed to develop react applications.
- **Public:** Directory that manages the app's static content, including index.html which contains the tag where the react app is displayed. Additionally, it contains a link to the app icon, useful metadata, The app's title and a reference to the manifest.json file situated within the public folder. The manifest.json file is a JSON file that contains information about how it behaves when installed on a mobile phone as a progressive web app.

- **Src:** In a react application, the "src" (source) directory is a fundamental part of the project structure. The source code, including CSS and JavaScript files, is stored in the directory. For example, within the "src" directory, main entry file, commonly named "index.js" or "App.js," which serves as the starting point react application can be find. Additionally, other components, utility functions, and stylesheets are often organized within subdirectories of "src". The "src" directory is essential for maintaining a clean and structured codebase in a react project.

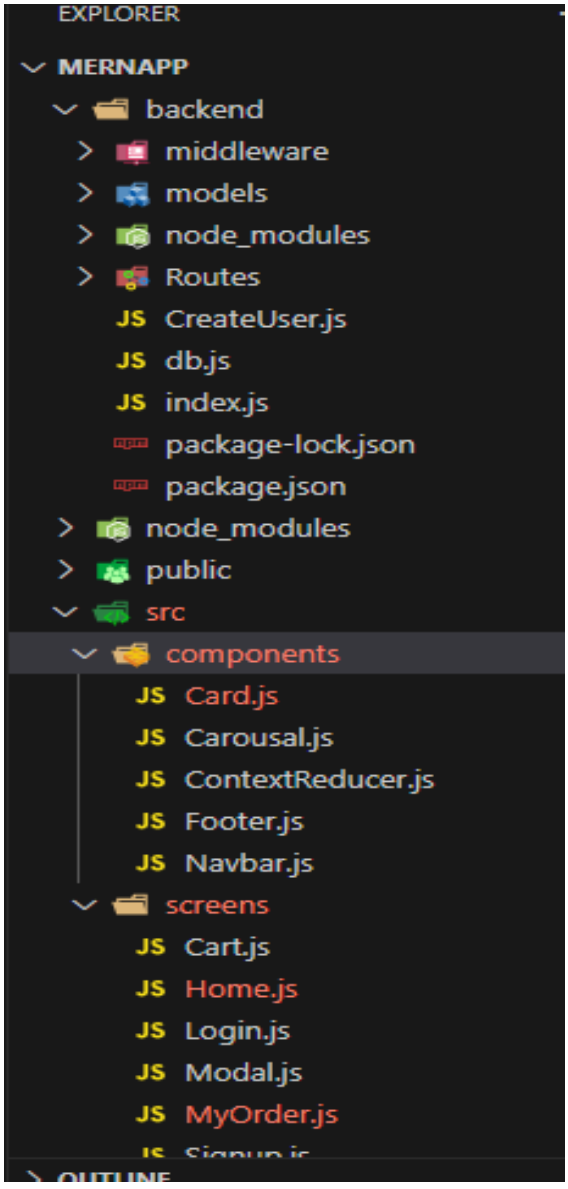

Figure 13 Directories structure of react app

5.1.6 API Integration and Data Fetching

API Integration and data fetching in MERN based full-stack application development involve the process of connecting the frontend with the backend to fetch and manipulate data from external sources or databases. It is fundamental aspects of full-stack application development, enabling applications to retrieve and manage data effectively from various sources. Figure 14 showing properly implementing these processes is essential for creating robust and responsive web applications.

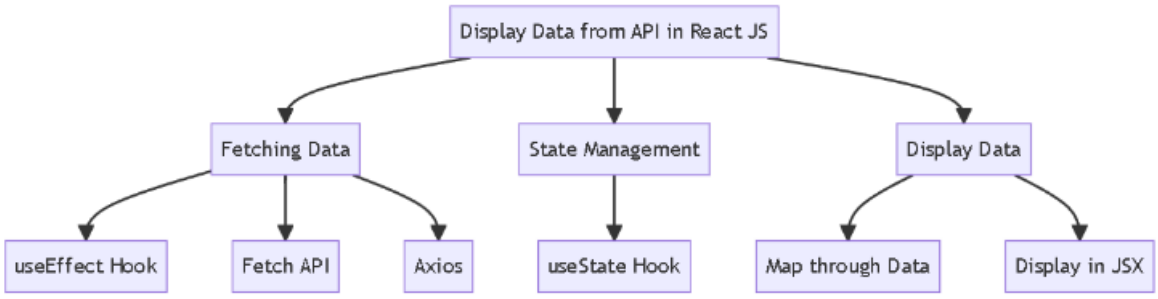

Figure 14 API work flow

Here's a simplified explanation of how API process works:

- **Application Programming Interface:** An API acts as a connection that enables communication between two different software applications. In web development, APIs are frequently used for obtaining data from external sources like databases or online services like payment gateways or social media platforms in web development.
- **Frontend Integration:** In a MERN stack, the frontend is typically built using React. To fetch data, JavaScript libraries like Axios can be used or the built-in fetch function. These tools send HTTP requests to the backend.
- **Backend API:** The backend, developed with node.js and express.js, serves as the intermediary between frontend and the data source. It contains API endpoints that receive requests from the frontend, process them, and return data as responses.
- **API Routes:** In Express, routes that handle specific types of requests (e.g., GET, POST, PUT, DELETE). Each route is associated with a controller function that manages the request and response logic.

- **Fetching Data**: When the frontend needs data, it sends an HTTP request to the appropriate API route on the backend. For example, a GET request could be used to fetch a list of products from a database.
- **Data Processing:** The backend controller function processes the request, which may involve querying a database or integrating with external APIs. Once the data is retrieved, it is formatted into a JSON response.
- **Sending Response:** The backend sends the JSON response back to the frontend, where it can be processed and displayed to the user.
- **Displaying Data:** In React, the fetched data can be stored in component state or props and then rendered in the user interface.
- **Updating Data:** APIs also handle data updates, such as creating, updating, or deleting records. These actions involve sending POST, PUT, or DELETE requests to the appropriate API endpoints.

5.1.7 Challenges, Impact and Solution in Front-end Development

Challenges in frontend development	Impact	Solution
Inconsistent browser compatibility	Renders differently in various browsers, affecting user experience.	Implement cross-browser testing and use CSS frameworks for consistency.
State management complexity	Difficulty in managing and synchronizing state across components.	Adopt state management libraries like Redux for centralized state control.
Performance bottlenecks	Slow rendering times impacting user interactions.	Optimize code, use lazy loading, and implement performance monitoring tools
Security vulnerabilities	Prone to cross-site scripting (XSS) and other security risks.	Employ secure coding practices, input validation, and utilize security libraries.
Scalability challenges	Difficulty in scaling the application as user traffic grows.	Implement code splitting, optimize components, and consider server-side rendering.
Learning curve for new developers	Steeper learning curve for beginners in the MERN stack	Provide comprehensive documentation, tutorials, and mentorship for new developers.

Limited SEO optimization	Challenges in optimizing for search engines due to client-side rendering.	Implement server-side rendering (SSR) or static site generation for improved SEO.
Dependency management	Version conflicts and challenges in managing multiple dependencies.	Use package managers like npm, and regularly update and audit dependencies.

Table 04 Challenges of Frontend, impact and solution

Addressing table 05 challenges in UI/UX design and frontend development is essential for creating web applications that not only function flawlessly but also offer an exceptional user experience. By proactively identifying and mitigating these challenges, developers and designers can ensure that their applications are efficient, user-friendly, and accessible to a diverse audience.

5.2 Back-end Development

The backend development also known as server-side development, plays a crucial role in development robust and functional web applications. In this phase, developers focus on creating the behind-the-scenes components of a website, which are responsible for data management, logic, and overall functionality. It involves working with programming languages, databases, and server technologies to ensure that the application runs smoothly and efficiently. Backend developers handle tasks like data storage, user authentication, and server management, this will allow the user interface and server to interact smoothly.

If an app is slow, crashes a lot, or keeps giving errors to users, it's probably due to issues in the backend. There are many tools and frameworks used in backend development. For instance, Node.js, MongoDB, and Express.js are all part of backend development. The backend takes care of both the server and the database.

5.2.1 Node.js for Back-end Development

Choosing the right backend technology is a crucial decision in the development of web applications. This choice directly impacts an application's performance, scalability, and functionality. When deciding on the right backend technology, developers must consider factors like the project's specific requirements, the expected user load, and scalability needs. Node.js is a JavaScript environment based on Chrome's V8 JavaScript engine. Node.js was initially developed by Ryan Dahl in 2009 for server-side applications. Figure 15 illustrates and explains the features of node.

Figure 15 Node.js features

Node.js enables the development of scalable, rapid, and lightweight applications. While commonly used on the server-side, it can also serve as a client-side platform for constructing JavaScript web applications. Node.js is built on an event-driven architecture, allowing it to handle asynchronous and non-blocking input/output (I/O) operations. Its distinct non-blocking I/O model eliminates the need for waiting to fulfill requests. This capability allows for the development of lightweight and scalable real-time web applications, proficiently managing various requests (Haque, 2019).

Node.js operates as a single process, as illustrated in Figure 16, avoiding the need to create new threads for each request. It performs I/O operations in two ways: blocking, which halts the process until data is received, and non-blocking, where the process continues while awaiting data. This non-blocking approach enables Node.js to efficiently handle multiple requests and simultaneous connections on a single server. Node.js provides its modules for various tasks like creating HTTP servers, reading local files, and managing child processes with access to the operating system. The Node.js modules alone may not offer the required functionality that other modules need to be added using npm. They should be selected on the basis of their popularity, maintenance status, and the credibility of the publisher.

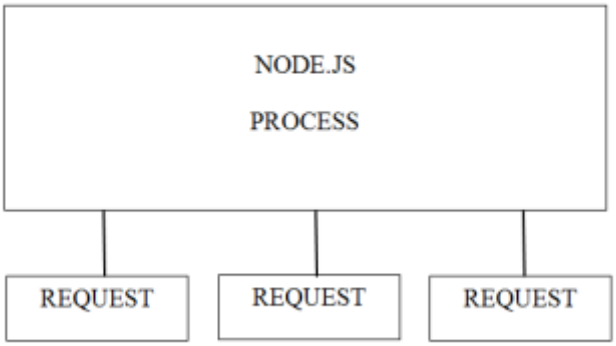

Figure 16 Single thread model of Node.js

5.2.2 Node Modules

In Node.js, a module refers to a reusable, self-contained unit of code that can be used to perform a specific tasks or provide certain functionalities. Node.js allows to create and use modules to structure code and promote code reusability. These modules can include variables, functions, classes, and other code elements which related to a particular task or feature. The module is defined as a node.js feature that is structured into JavaScript files and may be reused across the application.

In Node.js, some common types of modules are as follows:

- **Core modules:** These are built-in modules that come with Node.js. Examples include fs (file system), http (HTTP server and client), path (file and directory paths), and more. Core modules are able to be used in node.js applications and do not require installation of these modules separately.
- **Local modules:** These are modules created by developers for their specific application. Local modules are stored in separate JavaScript files and can be used within the same application by importing them using the require function.
- **Third-party modules:** These are modules created by other developers and shared through the Node Package Manager (npm). They can be install using npm and then include them in project by requiring them.

5.2.3 Node Package Manager (NPM)

Node Package Manager is a important tool for Node.js, enabling developers to manage and install code packages and handling dependencies within JavaScript applications. Node.js is a cross-platform JavaScript runtime environment extensively utilized for various projects and solutions, including serving as an HTTP server. NPM provides command-line tools to assist in the installation of various packages and manage their dependencies. Thousands of modules and packages that are created by developers using NPM can be used by other developers. This makes it faster to start applications because it is not necessary create all the features from the beginning. Modules installed from NPM can be utilized in both the backend and frontend. Using npm, it is very easy to find packages for specific needs, download them, install them, and manage the installed packages. The npm establishes a structure for Node.js packages that closely follows common JavaScript package functionality. NPM utilizes the package.json file, which is supported by Node.js, and extends it by additional fields to enhance its functionality (Herron, 2016).

In short, npm is:

- An online storage space for sharing open-source Node.js projects.
- A command-line tool that help in communication with the repository, facilitating the installation of packages and the supervision of package versions and dependencies.

5.2.4 Key features of Node.js

Node.js offers key features that make it an excellent choice for software architects

- **Asynchronous and event driven**: Each API within the Node.js library functions asynchronously implying that they do not block the server. Simply, the Node.js server does not pause while waiting for an API to transmit data. Rather, it will return to the API call and use a notification event mechanism of Node.js events for receiving responses from an earlier API call.
- **Single language:** JavaScript is used as the single programming language throughout the entire Node.js stack, both on the server (backend) and in the browser (frontend). This simplicity makes it easier for developers to work on both parts of a web application using the same language.
- **Large ecosystem:** Node.js contains a wide variety of Open Source libraries and packages available for developers, which significantly accelerates the development process

- **Fast Execution:** Node.js is a very fast code execution library based on Google Chrome's V8 JavaScript engine, this feature makes it an outstanding option for developing responsive and high-performance applications. Unlike other applications, Node.js applications never store data in buffers, they output data directly in chunks.

5.3 Express.js

Express.js is a node framework for developing web applications and mobile apps that offers a wide range of features. It is designed for the creation of a single page, multipage and hybrid web application. Express.js is a web framework built upon the core components of Node.js HTTP module and connect components. A set of tools for web applications is provided by the framework, handling HTTP requests and responses, managing routes, and employing middleware to building and launch extensive, enterprise-ready applications. It additionally provides a command-line interface tool known as node package manager. The express.js is a compact framework that uses the web server features of node.js to simplify its APIs and provide valuable new features. (Hahn, 2016). Figure 17 shows client hitting server if application used only Node.js without express.js. Figure 18 illustrates how a request goes through the app when use express.js.

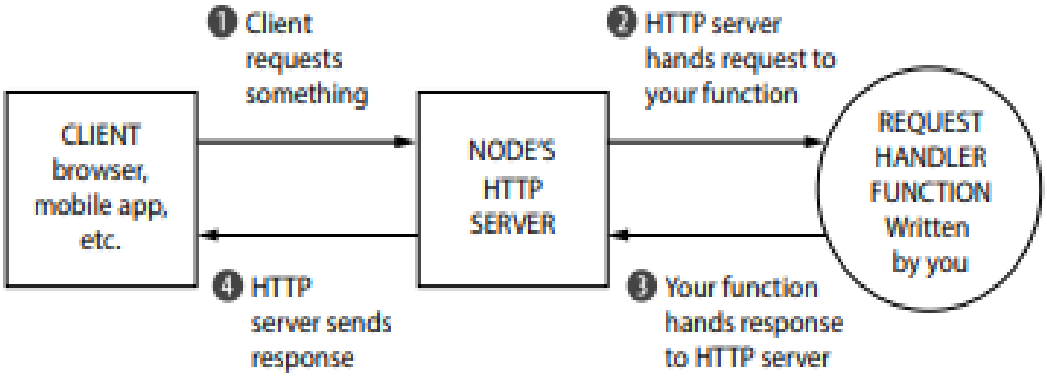

Figure 17 Journey of a request within a node.js application

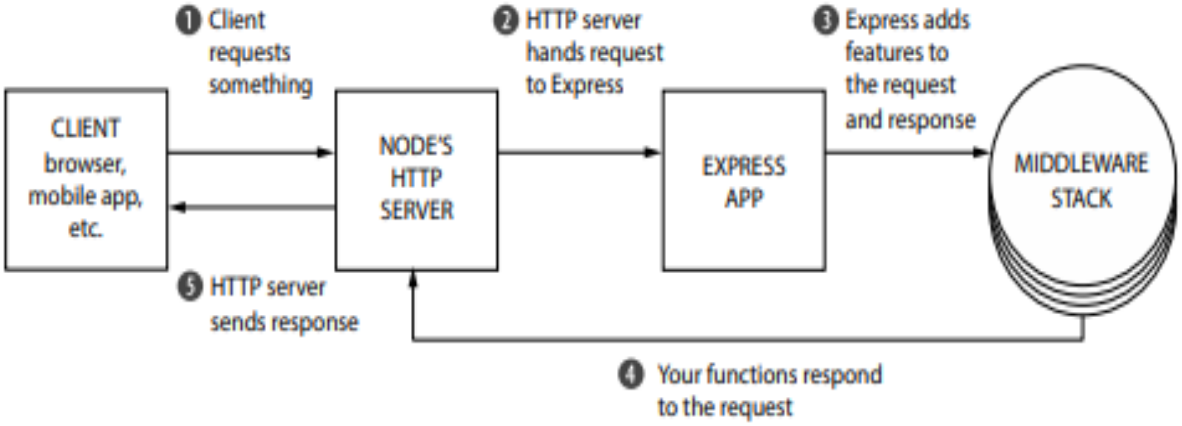

Figure 18 the flow of a request through Express

Key features of Exprees.js are outlined below:

5.3.1 Middleware

Middleware is a feature of Express that manages requests and responses. In a Node server, it usually listens for incoming requests and handles them one by one in a single process. Middleware functions are like helpers that break down this big task into smaller steps. They can do various things like changing requests and responses, and then move the process to the next middleware function to keep working on it. This helps in organizing and simplifying how the server handles requests.

Fundamental structure of middleware functions is shown in the figure 19. These functions requires three critical parameters the requests, response and next function in application's request-response sequence. In some cases, middleware functions might use two objects and one function. Middleware functions enhance the request or response objects and run code that can impact the application in many ways.

```
app.get(path, (req, res, next) => {}, (req, res) => {})
```

Figure 19 Middleware function request

5.3.2 Routing

Routing in web applications is essential because it decides how the application reacts when users request something. It is not just about the web address and method it can also look at things like headers, queries, and request content. In an application, when a user sends a request to a particular endpoint, the routing system of the application decides which handler function should be called to manage the request. This handler function can perform various tasks, like interacting with a database, processing data, or sending a response to the client. Additionally, routing can define middleware functions, which are executed before the primary handler function. Middleware functions perform tasks like authentication, logging, or data validation, and they can make changes to the request or response as needed. For example, it can use the app.get (path, callback) function to deal with incoming GET requests. There are other methods like post (), put (), and delete () for various operations. The "path" is the URL, and the "callback" is the function that manages the request.

Figure 20 illustrates how a route functions works in an application, specifically designed for a GET request. In this instance, "app" represents an instance of Express. The code "app.get" specifies the HTTP request method, which is GET in this case. "/home" denotes the route, which is the particular URL path. Lastly, "(req, res) => {}" represents the handler function. When it receives a matching request for this route, it triggers this function to handle the request and generate a response.

```
const express = require('express');
const app = express();

// GET method route
app.get('/home', (req, res) => {
  res.send('OK');
});
```

Figure 20 Route functions handle GET requests

5.3.3 Sub applications

When application gets larger, it is helpful to split it into different folders and files as shown in figure 21. Express provides a useful feature called sub applications or routers. These are like mini-apps within the app. Separate parts of the app, such as an administration panel, can be created by creating sub applications. It will keep code organized and manageable.

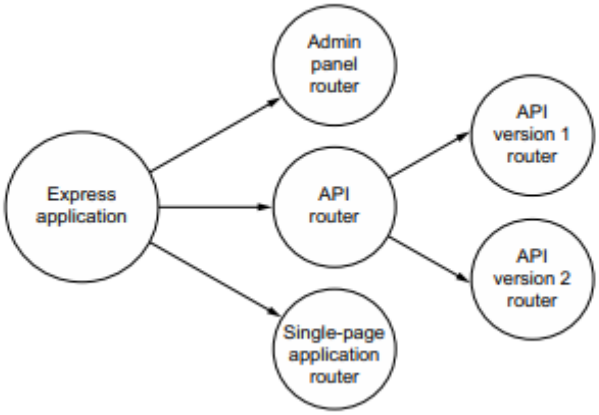

Figure 21 Routers creation

5.4 User Authentication and Security

User authentication and authorization are crucial aspects of web application development. They play a vital role in ensuring the security and privacy of user data while managing access to the resources within the application. User authentication is the process of verifying that the user's credentials are valid in order to confirm their identity. This process can involve different methods like passwords, biometrics (like fingerprints or facial recognition), or using two-factor to confirm identity. The main aim is to make sure that only the right people can use the app

User authorization is all about deciding whether to allow or refuse access to certain features or data within the app. To protect sensitive information and to limit unauthorized activity, this involves creating a number of roles, granting permissions or implementing access controls.

5.4.1 JSON Web Token

A JSON Web Token (JWT) serves as a secure method for transmitting information between two parties, It is commonly used in app development for authentication and authorization purposes. JWT is a concise, self-contained approach to represent information. It is easy for machines to read and generate, and it can be sent as part of a URL or in an HTTP header. It is like a digital passport that can be used to verify who you are and what you are allowed to do on a website or app. After a user signs in with their credentials, the server generates a JSON web token signed using a secret key and containing unique user details. (Haque & Haque, 2018). Figure 22 describe JWT authorization work flow.

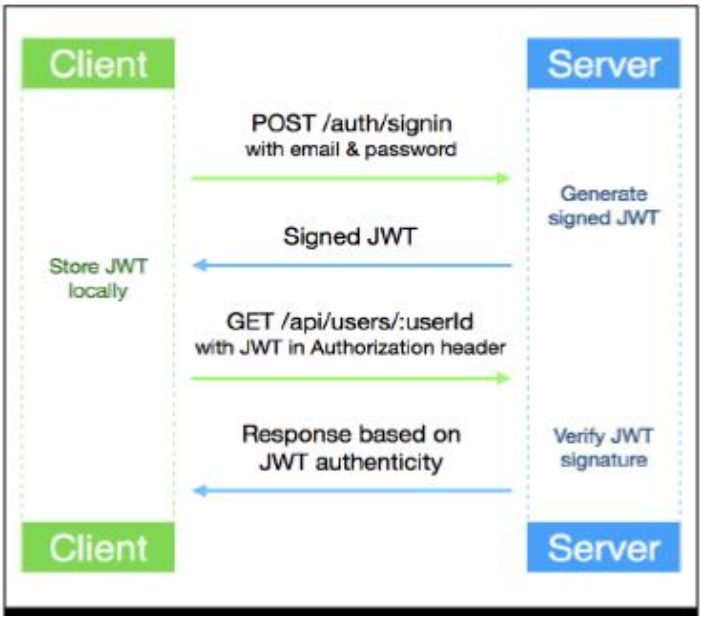

Figure 22 JWT authorization flow

User authentication and authorization are important for several key reasons:

- **Increased Safety:** by verifying the user's identity and restricting access to information, authentication and authorization mechanisms are effectively preventing unauthorized access to data as well as reducing the risk of data breach and malicious activities.
- **Access Control:** These mechanisms allow users to precisely control their user permissions, which ensures they only have access to the resources that are needed for their role. This also helps to protect sensitive data, and ensures the integrity of an application.
- **Personalization and User Experience:** The user authentication mechanism allows applications to configure content, settings and preferences according to the profiles of each user. This is also beneficial for users and contributes to their engagement.

5.5 MongoDB

MongoDB stands as a management system for databases categorized under the NoSQL database. NoSQL databases are used instead of relying on traditional relational database. They are great for handling large, distributed datasets. MongoDB is a technology that makes it possible to organize, save and access document based information. It can handle different data types. It is commonly used for big data applications and other tasks that involve data that does not work well in a strict relational model. MongoDB is a document-oriented, nonrelational database that can be used to

distribute and store big binary files like videos and images. Data is stored as a binary representation of a document called a BSON object (Eyada, Saber,Genidy & Amer, 2020).

MongoDB Atlas is a cloud-based database solution designed for modern applications, and it is accessible worldwide. It provides top-notch automation and follows best practices for deploying and managing MongoDB databases on popular cloud platforms like AWS, Google Cloud, and Azure. Utilizing MongoDB Atlas enables the easy setup and maintenance of fully managed MongoDB databases in the cloud (mongoDB, 2023).

MongoDB operates in two primary layers the application layer and the data layer.

1. Application Layer: This layer, also known as the final abstraction layer, is composed of two main parts:

- Front end: This is a user interface in which users are interacting with the database. Users have the flexibility to access the database either through mobile applications or directly using web interfaces.
- Back end: The back end refers to the server portion of the application. It handles logical operations and communicates with MongoDB using drivers and queries to manage data.

2. Data Layer: The data layer directly consists of the MongoDB server. When queries are initiated from the back end of the application layer, MongoDB receives these queries and forwards them to the storage engine. The storage engine is responsible for reading or writing data in memory.

5.5.1 Features of MongoDB

MongoDB is a document-oriented NoSQL database platform that offers many important features (mongodb.com, 2023).

- The document model, with its flexible schema is a method of organizing and storing data to meet specific application requirements.
- Replication to make the data more accessible and stable.
- Handle more data and traffic by splitting it into smaller pieces and distributing them across multiple servers or nodes.
- Ad-hoc queries on-the-fly and real-time analysis to quickly find and retrieve the specific data.

5.5.2 MongoDB vs MySQL

Table 05 shows the Head-to-head comparison of these two popular databases

Criteria	MongoDB	MySQL
Database Type	NoSQL	Relational
Flexibility and Scalability	Offers flexibility and scalability. Good for dynamic and evolving data structures.	Well-suited for well-defined and stable data schemas. Ideal for complex transactions and extensive data querying.
Application Suitability	Ideal for applications with unbound or semi-unstructured data. Good for continuous updates and high scalability.	Suitable for applications requiring strong data consistency, complex transactions, and extensive data querying capabilities.
Data Modeling	Enables flexibility in document-based data modeling	Follows a structured data model.
Real-Time Communication	Ideal for applications needing real-time communication (requires additional tools like Socket.io).	Well-suited for real-time communication between clients and servers, facilitating features like order tracking and chat functionality.
Data Representation	JSON documents	Tables and rows
Schema	No need to define	Must define columns and tables
Priority	Cloud-friendly	High data security

Table 05 Comparison of MongoDB and MySQL databases

5.6 Mongoose

Mongoose is a query language that may be used to read, write, update, or delete data between server and database. Mongoose is a comprehensive object document mapping (ODM) library designed for Node.js and MongoDB (Mardan, 2018). Mongoose uses schemas to organize data, which makes the database more clear and understandable. It also confirms that the data being entered into the database is accurate. Mongoose gives all the features of MongoDB, like tools for searching for data and adding special rules to the data. Mongoose provides all of MongoDB's features, including tools for query creation and logic in the data. Figure 23 explaining the functionality of mongoose.

The disadvantage of Mongoose is that it can slow down some queries because it has to process a lot of code. However, using this kind of tool is very important in modern software development, especially for big businesses. The primary advantage of Mongoose is its ability to simplify and clarify interaction between application code and the database by abstracting away the complexities. In this setup, the application code communicates solely with objects and their methods, making the process simpler and easier to understand.

```
const mongoose = require('mongoose')

const { Schema } = mongoose;

const OrderSchema = new Schema({
    email: {
        type: String,
        required: true,
        unique: true
    },
    order_data: {
        type: Array,
        required: true,
    },

});

module.exports = mongoose.model('order', OrderSchema)
```

Figure 23 Mongoose functionality

5.7 Data Retrieval

The data retrieval process involves accessing and extracting data from a storage system, typically a database or data repository, for the purpose of using, displaying, or analyzing that data. Data retrieval is crucial in various applications, including web development, where user interfaces and web applications fetch data from databases and display it to users.

Key aspect of data retrieval include:

- **API endpoints:** The backend will expose API endpoints to retrieve data from the database. These endpoints are URLs that the frontend can request data from. For example, to fetch a list of food items from a food delivery application, there might be an API endpoint like /api/products.
- **Front end:** The frontend, developed using React, can make HTTP requests to these API endpoints. Data may be retrieved using a variety of methods, such as GET requests.

- **Display:** The frontend will display the data to users once it has been retrieved. This might involve rendering lists of items, showing user profiles, or presenting any information stored in the database.

6 Project Implementation

6.1 Evaluation of a MERN Based Food Delivery Application

The study aimed to develop and evaluate a food delivery application using the MERN stack. The application is designed for individuals who wish to place food orders from restaurants and have them delivered to their homes. The goal of developing a food delivery app is to establish a system that is user-friendly and efficient, allowing people to easily order food from various restaurants. This should make their dining experience better and generating revenue for both the app providers and participating restaurants. The secondary goal was to identify backend and frontend technologies that enhance the efficiency of full-stack development. This involved assessing the range of features offered by modern frontend and backend technologies and evaluating the significance of these features in the context of technology assessment.

A MERN-based food delivery application operates by combining various technologies to provide a seamless experience for users. Full-stack application depends on MongoDB for the database, utilizes Express.js as the server framework and use react for the front-end along with Node.js for server-side scripting. React offers a powerful and interactive user interface, while Node.js provides a robust server-side environment for efficient scripting. Users access the application through a React-based client-side interface, where they can browse restaurants, select food items, and place orders. The client-side communicates with the server using API endpoints, enabling it to request information and interact with the back-end. Meanwhile, the server, driven by Node.js and Express.js, manages user requests, business logic, payment processes, and retrieves data from the MongoDB database. Additionally, real-time features, feedback and ratings, and interfaces for both customers and restaurants contribute to a comprehensive food delivery application. This application's architecture ensures a smooth experience, from order placement to food delivery and feedback submission. The MERN stack makes it easy to do things quickly and can be used in many different types of web applications.

6.2 Environment Setup

Setting up an appropriate development environment is a crucial step in any software development project. The right tools and environment can have a huge impact on efficiency, productivity, and overall project success.

In this project, visual studio code acted as the editor, providing various features that enhance productivity and simplify code maintenance. Visual studio code apart is its vast ecosystem of tools and extensions that can greatly enhance productivity and efficiency for developers (Dwyer, 2023). Visual studio code allows to write code in many different programming languages without having to switch to different editors.

Visual studio code provides comprehensive language support, including Python, Java, C++, JavaScript, and more. Additionally, various third-party tools and packages were employed during the development process. These tools added extra functionalities and simplified the development tasks. Figure 24 presenting visual studio code unique features.

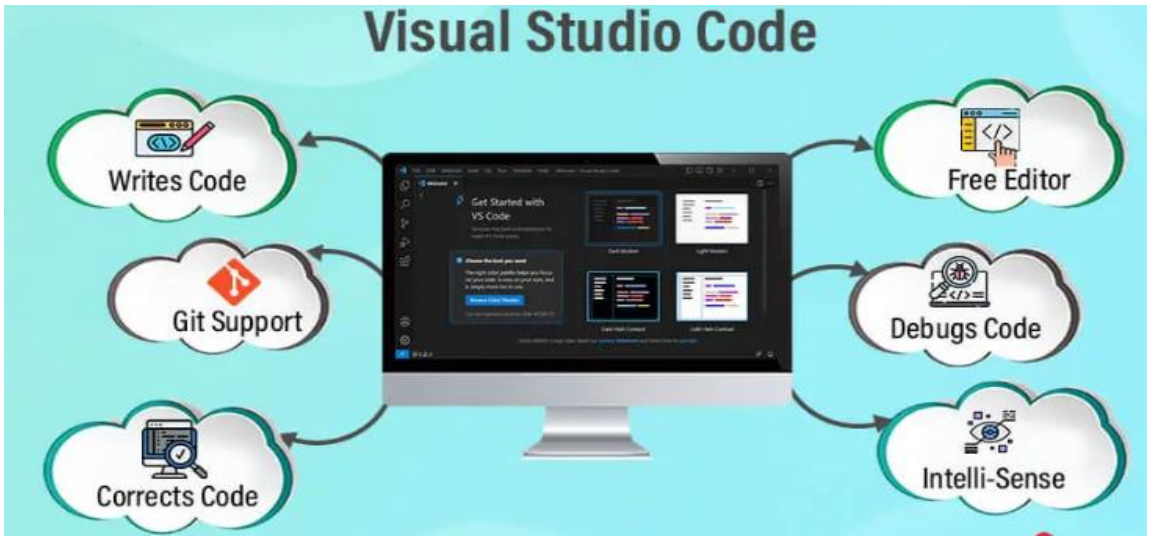

Figure 24 Features of visual studio code

6.3 Back-end Implementation

This section covers the installation and configuration of different packages and tools. NPM (Node Package Manager) was utilized to manage and install dependencies for this project.

NPM is not just any package manager it stands as a world's largest software registry for JavaScript. There are three essential elements the website, CLI, and the registry. The website functions as a hub where developers can explore packages and manage both public and private packages. The NPM

CLI is the tool that developers use to interact with NPM. It allows to run commands for manage packages, initiate projects, and execute scripts defined in package.json file. NPM Registry is the central repository where JavaScript packages are hosted and available to the others. The registry is used by developers to retrieve and integrate existing packages into their applications and to use them to meet their specific needs.

For configuring the runtime environment to run server-side JavaScript in this project, Node.js version 20.09.0 was installed from the official website. Once Node.js installation was complete, the next step included initiating the project through the "npm init" command in the root directory of the project. This command created a "package.json" file, which serves as the project's configuration file. This file empowers developers to define and manage various dependencies crucial for tasks like building and execution the application.

Once Node.js is installed, the next step involves integrating server-side logic into the application with Express.js. This is done by utilizing the command "npm install express" in the root directory of the project, Version 4.18.2 of Express was chosen and incorporated into the project. This command ensured that Express.js was set up and ready for use in the application. Figure 25 showing backend package.jsn file.

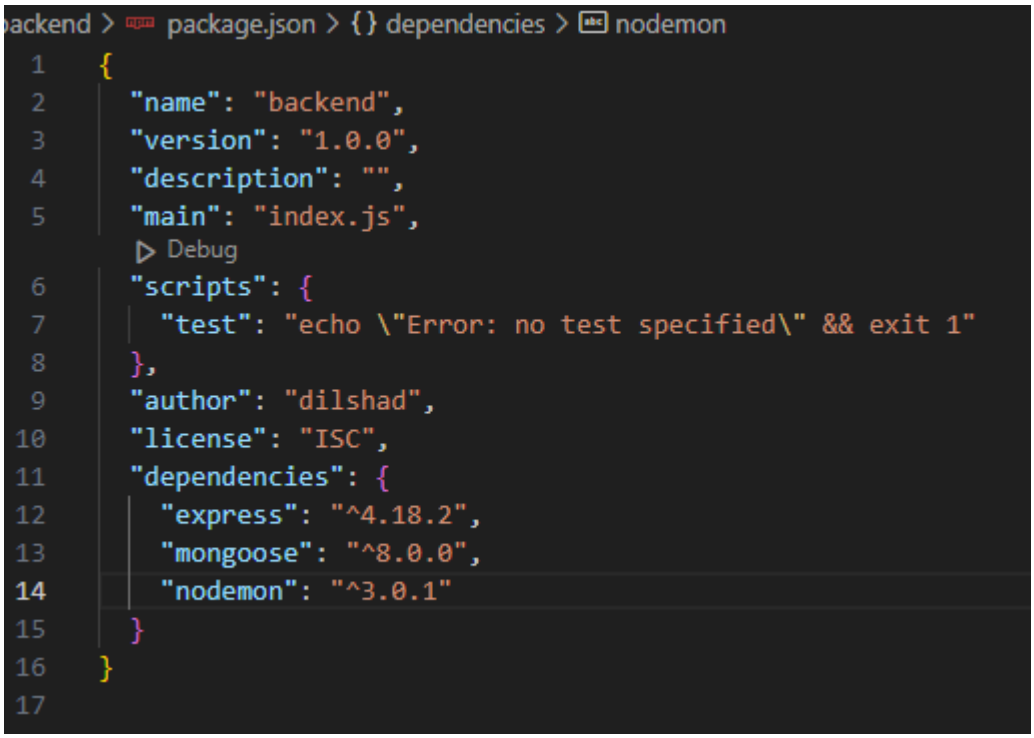

Figure 25 Backend’s package.json file

6.4 Front-end Implementation

Beginning the development of a full-stack food delivery application involves creating a new react application using the "npx create-react-app" command. This command helps set up the basic structure of the application. After that "npm start" command will be used to launch a special server that shows react app in web browser. The application appearance and functionality become visible immediately when "http://localhost:3000" is opened in a web browser. Additional features were incorporated into the project through React.js packages like react-router-dom and react-icons. React Router DOM, in particular, played a key role in managing the routing aspects within the application.

When React is all set up, the focus turns to making the app look good and making it easy for people to use. Elements like menus, order forms, and other parts of the app play a crucial role. They work quickly and respond well when people use them, which is especially important for a food delivery app that needs to show real-time updates for menus and new orders. Figure 26 front-end package.json.

Figure 26 Front-end´s package.json

6.5 Database Implementation

There is a choice of either installing mongoDB directly on local machine or using a cloud-based service like mongoDB Atlas. In this application development, mongoDB atlas was the chosen option. After creating an account, a new cluster was created, and all the required information was provided. Eventually, a connection string was obtained, which holds essential information like usernames and passwords for connecting to the mongoDB atlas cluster. The figure 27 below shows the cluster created for application.

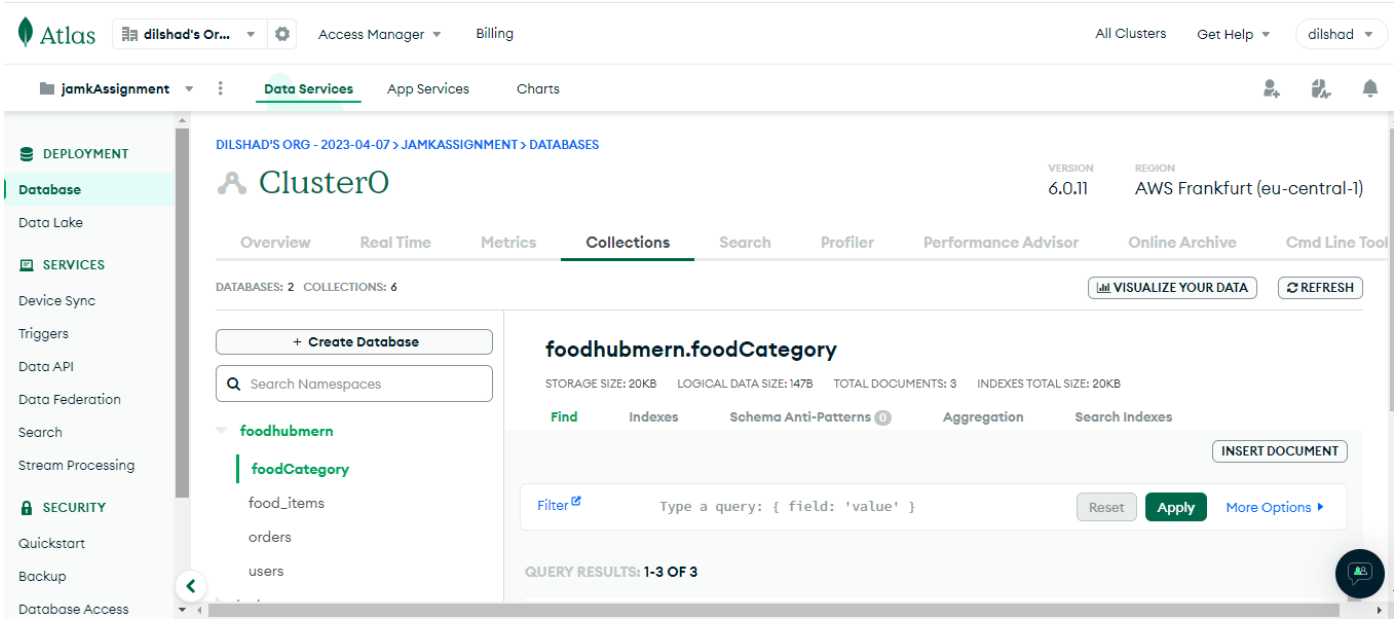

Figure 27 screen shot of MongoDB atlas cluster

Install Mongoose

- Open the terminal and navigate to the backend directory within the project.
- Execute the command "npm install mongoose" in the terminal to install Mongoose as a project dependency.

Install Nodemon

- Nodemon is a development tool that helps with automatic server restarts when any changes done in code. It is usually installed as a global package, so there is a need to install it just once.
- Run the “npm install nodemon” command to install nodemon globally.

Nodemon will monitor files for changes and automatically restart server when code will save after change, which is very helpful during development. Having Mongoose and Nodemon installed, it is all set to work on project, especially when dealing with databases (Mongoose) and server development (Nodemon).

Install bcrypt.js

- Open the terminal and go to the root directory.
- Type “npm install bcryptjs” command to install bcrypt.js as a project dependency using npm.

This command will download and add the bcrypt.js package to project. It is frequently utilized for hashing and securely storing passwords within applications.

6.6 Version Control System

A version control system is a software tool designed to assist in the management and tracking of changes to files and projects over time. It facilitates collaboration among multiple individuals working on the same project, maintaining a detailed history of all modifications. VCS provides features like version tracking, collaboration, and the ability to revert to earlier states of a project. Version control software keeps records each code modification in a specialized database. In case of errors, developers can revert back and compare previous code versions, facilitating error correction with minimal disruption to the entire team. Version control systems facilitate data sharing among nodes, ensuring that each node remains updated with the latest data version (Koc & Tansel, 2011).

According to Otte (2009) two primary categories of version control systems exist:

- Centralized version control Systems (CVCS)
- Distributed version control systems (DVCS)

In a centralized version control system (CVCS), a central repository stores all project files and their version history. Users can retrieve files, make modifications, and then update the repository, ensuring synchronization across all contributors.

A complete copy of the project repository is accessible to all users and includes its full history in distributed version control systems. Users can work independently and share changes with others by pulling and pushing revisions. Git and Mercurial are popular examples of DVCS.

Today, DVCS systems are more popular because they give more freedom and safety. They are used in lots of areas, like managing software code, writing together with others, and many more.

7 Results and Analysis

7.1 Thesis Results:

The study concluded that the MERN stack is an efficient choice for creating a responsive and scalable application. It provides real-time features, an appealing user interface, and strong data security. The final results of the thesis will provide more concrete findings.

- **MERN Stack for Food Delivery Application**

  The MERN stack is suitable for developing food delivery web applications for several reasons. MongoDB, the NoSQL database in the stack, enables flexible and scalable data storage, which is essential for managing various restaurant menus and user orders. Using Express.js as a backend framework enables rapid API development and contributes to efficient communication between server and client. React.js, a front-end library, provides a dynamic and responsive user interface that allows for smooth navigation and an engaging user experience. As a runtime environment, Node.js supports real-time data processing and efficient handling of concurrent user requests, which is essential for managing food orders and deliveries. This technology stack's use of JavaScript across both frontend and backend development simplifies the development process, offering consistency and ease of maintenance, making it an ideal choice for creating feature-rich and responsive food delivery web applications.

- **Key Technological Considerations**

  The development of a food delivery web app involves addressing various technological aspects and challenges to ensure a smooth user experience and effective order management. One main focus is the creation of user-friendly interface that facilitates easy navigation and browsing restaurants and menus. Real-time order tracking is a challenge that requires accurate geolocation integration for accurate delivery estimates and driver locations. Implementing secure payment gateways that accept various payment methods requires strict security measures to protect users' sensitive data. The dynamic cart system needs real-time updates and customization options for user preferences. Third-party service integration requires strong APIs protocols for effective data exchange. The backend systems need to handle order information accurately.

To overcome these challenges, a full-stack development approach incorporating technologies like JavaScript frameworks for frontend development, server-side scripting languages for backend logic, and secure database management systems is crucial. The application's architecture needs to be scalable to manage varying levels of demand during peak periods and guarantee a seamless user experience.

- **Enhanced Data Security**

Securing user data in a food delivery application is a crucial for maintaining trust and privacy. Implement strong authentication and authorization protocols to control that only authorized users have access to specific functionalities or data. Utilize HTTPS to encrypt data transmitted between the user's browser and the server which is preventing unauthorized access during communication. Furthermore, Secure API design, data minimization and having transparent privacy policies contribute to data security. By addressing to these measures a food delivery web application can secure user data and maintain a trustworthy reputation.

- **Scalability Considerations**

When developing a MERN based food delivery website with scalability, there are several important aspects need to be consider. It is important to use a strong and scalable architecture within the MongoDB, Express.js, React.js, and Node.js technology stack. Cloud based services and databases, with scalability features can handle more users when there is a lot of traffic which improve application performance during high traffic times. Implementing efficient data caching mechanisms and optimizing database queries helps manage more users using the app at once. The combination of technology selection, code optimization and continuous monitoring has led to the scalability of MERN web applications for food delivery.

- **MERN based food delivery application Features**

Developing a MERN-based food delivery web application requires many important features to ensure its effectiveness and user appeal. The integration of MERN is essential technology stack that enables a consistent and streamlined development process. Key features include a user-friendly interface that makes it easy for customers to navigate and order. Also, the application includes real-time order tracking, secure payment gateways and customization

options to suit user preferences and dietary requirements. Additionally, effective order management and seamless interaction with external services requires a strong backend infrastructure. Scalable and optimized data flow from front-end components to back-end components is critical to achieving the best performance. These core features form the core components required for a comprehensive and functional MERN food delivery web application.

7.2 Ethical Considerations in food delivery App Development

Ethical considerations in food delivery app development involve various aspects that developers and companies need to consider presenting in table 06.

Ethical Questions	**Ethical Considerations and Resolutions**
How can user data privacy be ensured in the application, especially given the sensitive nature of personal and financial information?	Implement robust data encryption and privacy policies to protect user data.
How can the application promote healthier food choices and address dietary restrictions, considering public health concerns?	Collaborate with restaurants to offer healthier menu choices and ensure transparent nutritional information
Are pricing structures transparent and free from hidden fees or price gouging?	Clearly display pricing details, including delivery fees and taxes.
What safeguards are in place against unfair competition, monopolistic practices, and the potential to harm local businesses?	Comply with local and international regulations to prevent monopolistic practices.
What measures will be taken to prevent fake reviews and maintain trustworthiness in user feedback and ratings?	Implement review verification mechanisms and report fake reviews.
What steps are in place to ensure food safety and quality during delivery, minimizing health risks?	Collaborate with restaurants to establish and adhere to food safety protocols.
What is the impact of food delivery applications on local businesses and communities, and how can potential negative consequences be mitigated?	Engage with local communities to understand their needs and concerns, and implement community-focused initiatives.

Table 06 Ethical Considerations

8 Conclusion and Discussion

8.1 Limitations

The MERN stack is commonly used for developing web applications. Although this has various benefits, there can be limitations to developing MERN based food delivery applications. Implementing real-time order tracking updates or live status updates can be complex and requires additional tools and libraries such as Socket.IO. Handling real-time data synchronization between multiple clients in the MERN stack can be difficult. React`s virtual DOM can cause performance issues in large applications or complex UI components. While MongoDB's flexible schema has its benefits, it can also result in unstructured data if not planned properly. Ensuring data consistency and integrity can be challenging without a predefined schema.

8.2 Future Recommendations

Here are some future recommendations for MERN based food delivery application development, multiple recommendations will significantly improve user experience.

- Addressing the limitations mentioned in the section 8.1.
- A secure payment system that offers a variety of payment methods.
- Adding delivery tracking support.
- Enhance search functionality for quicker food searches.
- Expand restaurant listings for a wider variety of choices.
- Incorporate user feedback mechanisms for continuous improvements.

8.3 Research Questions and Outcome

This research had five clearly defined objectives as mentioned in chapter 2. They have been rewritten below with thesis outcome:

1. **Why is the MERN stack consider suitable for developing food delivery web applications?**

   The MERN stack is suitable for food delivery web applications due to its use of MongoDB for flexible and scalable data storage. Express.js for rapid API development and efficient server-client communication, React.js for a dynamic user interface, Node.js for real-time data

processing and JavaScript's consistency in both frontend and backend, simplifying development of responsive applications.

2. **What are the key technological considerations and challenges in the development of a food delivery web application that aims to provide a seamless user experience and efficient order management?**

   Key challenges in food delivery app development involve creating a user-friendly interface, real-time order tracking, secure payments, dynamic cart systems, integrating third-party services, ensuring accurate backend order management, and building a scalable architecture. Meeting these challenges requires a full-stack approach using JavaScript frameworks, strong server-side scripting, secure databases, and scalable architecture for a smooth user experience even when demand changes a lots.

3. **How can user data be secured in a full stack web application?**

   Securing user data in a full-stack web app requires strong authentication, HTTPS encryption for data transmission, and secure API design. Reducing data and making clear transparent privacy policies also contribute to overall data security and trust in the application.

4. **What are the critical factors to consider when developing a food delivery web application for scalability?**

   Considerations for scalability in developing a MERN-based food delivery web app involve a strong architecture using MongoDB, Express.js, React.js, and Node.js. Using scalable cloud services and databases manages high traffic for better app performance. Efficiently storing data and improving database queries handle more users. The integration of technology, code optimization, and ongoing monitoring enables scalability in MERN-based food delivery applications.

5. **What are the key features required for a full-stack food delivery web application?**

   The main features of a full-stack food delivery web app are a user-friendly interface for easy ordering, real-time order tracking, secure payments, and customization for user preferences. Effective order management and a strong backend infrastructure are essential, along with smooth data flow between the front-end and back-end for top performance, making a complete and effective MERN food delivery app.

8.4 Conclusion

In conclusion, evaluating the suitability of the MERN stack in the development of food delivery applications represents a remarkable achievement in the web development field. As online shopping continues to gain popularity, the mechanisms for online delivery have successfully addressed challenges that researchers identified a decade ago. These problems included slow website loading, issues with transactions, making payments securely, and having very few food options available online (Israel et al., 2019). The project has demonstrated the strength and adaptability of the MERN stack bringing together MongoDB, Express.js, React, and Node.js to create a dynamic and feature-rich application. This effort began by setting up a robust development environment, including installing key tools and dependencies. React.js plays a key role in developing a user-friendly and responsive frontend, ensuring seamless interaction and real-time updates of menus and orders. The integration of MongoDB offered a scalable and efficient database solution, while node.js and express.js adeptly managed the server-side logic. The development process also considered the importance of security by incorporating technologies such as Bcrypt.js. In summary, the project achieved its objectives and served its purpose well. Most of the requirements relating to online food ordering from restaurants are expected to be met by this application.

References

Alimadadi, S., Mesbah, A., & Pattabiraman, K. (2016). Understanding asynchronous interactions in full-stack JavaScript: *In Proceedings of the 38th International Conference on Software Engineering*.

Ahuja, K., Chandra, V., Lord, V., & Peens, C. (2021, September 22). Ordering in: *The rapid evolution of food delivery. McKinsey & Company*.

https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ordering-in-the-rapid-evolution-of-food-delivery

Beyrouthy, L. (2023, 16 November). Revenue of the online food delivery market in Europe 2017-2027. Statista. https://www.statista.com/forecasts/696525/online-food-delivery-revenue-by-segment-in-eu rope

Creswell, J. W. (2009). Research design: Qualitative, quantitative, and mixed methods approaches. SAGE Publications Asia-Pacific Pte. Ltd.

https://www.ucg.ac.me/skladiste/blog_609332/objava_105202/fajlovi/Creswell.pdf

Canziba, E. (2018). Hands-On UX Design for Developers.

https://books.google.fi/books?id=DAlnDwAAQBAJ&printsec=frontcover&redir_esc=y#v=onepage&q&f=false

Chen, S., Thaduri, U. R., & Ballamudi, V. K. R. (2019). Front-End Development in React: An Overview. Engineering International, 7(2), 117-126.

Dawson, C. (2009). How to decide upon a methodology (14).

https://www.ascdegreecollege.ac.in/wp-content/uploads/2020/12/Introduction-to-Research-Methods.pdf

Dwyer, T. (2023). 4 Essential Tools for Visual Studio Code.

https://levelup.gitconnected.com/4-essential-tools-for-visual-studio-code-f6890f2e600d

Eyada, M. M., Saber, W., El Genidy, M. M., & Amer, F. (2020). Performance evaluation of IoT data management using MongoDB versus MySQL databases in different cloud environments. IEEE access, 8, 110656-110668.

Joo, H. (2017). A Study on Understanding of UI and UX, and Understanding of Design According to User Interface Change. https://www.ripublication.com/ijaer17/ijaerv12n20_96.pdf.

Kunst, A. (2023). Online food delivery bookings by brand in Finland as of June 2023 https://www.statista.com/forecasts/1188039/online-food-delivery-bookings-by-brand-in-finland

Hoque, S. (2020). Full-Stack React Projects (2nd Ed.). Packt Publishing Ltd

Hoque, S., & Hoque, S. (2018). Full-stack react projects: *Modern web development using react 16, node, express, and mongodb*. Packt Publishing, Limited

Herron, D. (2016). Node. js web development - third edition : *Create real-time server-side applications with this practical, step-by-step guide*. Packt Publishing, Limited.

Hahn, E. (2016). Express in Action: *Writing, building, and testing Node. js applications*. Simon and Schuster.

Indla, B. V. S., & Puranik, Y. (2021). Review on React JS. International Journal of Trend in Scientific Research and Development. (5). pp. 1137-1139. www.ijtsrd.com/papers/ijtsrd42490.pdf

Israel, D.J. et al. (2019). Consumer's Intention to Continuous use of mobile food delivery aggregator app. Journal of Advanced Research in Dynamical and Control Systems, 119(11).

Koc, A., & Tansel, A. U. (2011). A survey of version control systems. ICEME. https://www.iiis.org/cds2011/cd2011imc/iceme_2011/paperspdf/fb394vz.pdf

Mardan, A., & Mardan, A. (2018). Boosting node.Js and MongoDB with mongoose. Practical Node.js: *Building Real-World Scalable Web Apps*, 239-276.

MongoDB. (2023). Database. Deploy a multi-cloud database. https://www.mongodb.com/atlas/database

MongoDB. (2023). MongoDB Features. mongodb.com. https://www.mongodb.com/features

Mwita, M. K. (2022). Factors to consider when using qualitative interviews in data collection. Soc. Sci. Humanit. Educ. J(SHE J.), 3, 313-323.

DOI: 10.25273/she.v3i3.13919

Norris, H. (2020). Food Delivery App Development – Benefits, Features, Challenges, Cost, Process.

https://www.restroapp.com/blog/food-delivery-app-development-guide/

Naoum, S. G. (2007). Dissertation Research and Writing for Construction Students. 2nd Edition,

Butterworth-Heinemann, Cambridge.

Otte, S. (2009). Version control systems. Computer Systems and Telematics.

https://www.mi.fu-berlin.de/inf/groups/ag-tech/teaching/2008-09_WS/S_19565_Proseminar_Technische_Informatik/otte09version.pdf

Olson, K. (2016). Essentials of Qualitative Interviewing. Abingdon.

https://www.routledge.com/rsc/downloads/9781598745955_Olson1.pdf

Pärli, M. (2020). Interview | Wolt Baltic CEO: We should list before turning 40

https://news.err.ee/1101564/interview-wolt-baltic-ceo-we-should-list-before-turning-40

Ray, A., Dhir, A., Bala, P. K., & Kaur, P. (2019). Why do people use food delivery apps (FDA)? A uses and gratification theory perspective. Journal of retailing and consumer services, 51, 221-230. https://doi.org/10.1016/j.jretconser.2019.05.025

Reddy, C. S., & Aradhya, G. B. (2020). Driving forces for the success of food ordering and delivery apps: *a descriptive study. International Journal of Engineering and Management Research, 10,* https://doi.org/10.31033/ijemr.10.2.15.

Rubin, H. J., & Rubin, I. S. (2012). Qualitative interviewing: The art of hearing data (3rd ed.).

Thousand Oaks, CA: Sage.

Singh, G., Javed, M., & Dhaliwal, B. K. (2022). Full Stack Web Development: Vision, Challenges and Future Scope. 9(4), 3083-3088. https://www.irjet.net/archives/V9/i4/IRJET-V9I4398.pdf

Vailshery, L. S. (2023, July 19). Most popular web frameworks among developers worldwide 2023.

Wolt. (2023). Reach new customers and get more orders with Wolt. https://explore.wolt.com/en/fin/merchants

Wolt, (2023). Free delivery offer exclusively for new users https://wolt.com/en/fin/vaaksy/article/ilmaiset-kuljetukset-apr23

Appendices

Appendix 1. Professionnal Information of Respondent

Interviewee Designation	**Company**	**Company Profile**	**Website**
UI/UX Designer, Front-end Developer, Back-end Developer	Indicsoft technologies	Software and services based multinational company.	https://www.indicsoft.com/
Project Manager	BaseMark	Basemark is a Finnish software development company that provides automotive graphics software tools and services	https://www.basemark.com/
Developer	CloudValue Tree	Product and software development company. Which is based in Finland and India.	https://www.cloudvaluetree.com/
Sales head	FreskFoods	Manufacture of dairy products and cheese. It also work in food industry.	Freskfoods Oy (Y ID 3118232-9)
CTO	CloudKraft	Software development and services based company.	https://cloudkraft.eu/

Appendix 2. Interview Questions for IT experts and Sales Head

1. Explain the purpose of the interview and its significance in research.
2. Discuss their perspective on the impact of full-stack technology.
3. Technical challenges faced in developing Full-stack applications.
4. Discuss any innovative solutions or technologies they recommend for addressing these challenges.
5. Inquire about their insights on ensuring data security and privacy in application.
6. Explore their opinions on creating user-friendly interfaces.

7. Inquire about UI/UX design principles and their role in improving user experiences.
8. Ask for their recommendations on the ideal technology stack for full-stack web application development in the context of thesis application.
9. Discuss emerging trends and innovations in the IT industry related Full-stack application.
10. Inquire about technologies or approaches that can enhance application performance.

Interview question for sales head

1. Explain the purpose of the interview and its significance in my research.
2. Inquire about their approach to collecting and utilizing customer feedback and data to improve sales and customer satisfaction.
3. Explore their insights on the challenges and opportunities faced by sales teams in the food delivery app industry.
4. Ask about emerging trends and innovations in sales and marketing within the food delivery industry.
5. Discuss technologies or approaches that can enhance sales efforts.