UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Aleksander Daniel Veske

Development and Implementation of a Full-Stack Food Delivery Website

Bachelor's Thesis (9 ECTS)

Supervisor: Lidia Feklistova, PhD

Tartu 2023

Development and Implementation of a Full-Stack Food Delivery Website

Abstract:

The aim of the bachelor’s thesis is to develop a full-stack food delivery website that can be used by users, both selling and buying products. With the difference from existing websites being the usage of weather data to calculate the delivery fee. The thesis written part includes a plan for the website, website requirements, the architecture of front-end and back-end and a plan for prospective enhancements.

Keywords:

Spring Boot, Spring Security, Angular, PostgreSQL, AWS, Docker, Swagger UI, Website, Testing, Full-Stack Development

CERCS: P175 Informatics

Full-Stack toidu kohaletoimetamise veebisaidi arendamine ja juurutamine

Lühikokkuvõte:

Bakalaureusetöö eesmärk on välja töötada full-stack toidu kohaletoimetamise veebisaidi, mida kasutajad saavad kasutada nii tooteid müües kui ka ostes. Erinevus olemasolevatest veebisaitidest seisneb ilmaandmete kasutamises kohaletoimetamistasu arvutamisel. Lõputöö kirjalik osa sisaldab veebisaidi plaani, veebisaidi nõudeid, esi- ja tagaosa arhitektuuri ning tulevaste täienduste plaani.

Võtmesõnad:

Spring Boot, Spring Security, Angular, PostgreSQL, AWS, Docker, Swagger UI, Veebisait, Testimine, Full-Stack arendus

CERCS: P175 Informaatika

Table of Contents

Introduction

In the modern day, the popularity of ordering food is more than ever. People find it convenient to order food online and have it delivered next to their doorstep. There is no stopping the industry as long as there is the demand, so there are no signs of the food delivery industry declining. Meanwhile, there is a competition between food delivery companies to have various restaurants in their lists, and enough drivers who can deliver the order. Since in many companies there are a lot of drivers-cyclists, they should receive fair pay, which, in turn, should take into account weather conditions.

In this context, the bachelor’s thesis is focused on the development and implementation of a full-stack food delivery website. The website is designed to provide a platform for both sellers and buyers and has a unique feature to calculate the delivery fees based on weather and other rules. The point of this study is not only to plan and construct the website, but also to address various technical aspects of the website, such as front-end development, back-end development, security considerations and the hosting of the website.

This thesis combines a range of technologies, such as Java, Spring Boot, Angular, PostgreSQL, AWS. It offers to the author of the current thesis a learning opportunity and contributes a practical solution to the ever changing online food delivery industry.

The bachelor's thesis consists of nine chapters: in the first chapter, to help the reader, the meanings of some more complicated terms that may be unfamiliar to the reader are given; the second chapter is about existing food delivery applications and websites, the brief analysis of the food delivery phenomenon and the reasoning about the development of the website; the third chapter describes the the prototype of the website, the plan of its development along with functional and non-functional requirements; the fourth chapter focuses on the technologies used to create the website; the fifth chapter shows the architecture of the front-end, its pages and core functionality; the sixth chapter focuses on back-end architecture, database structure and the components of the back-end; the seventh chapter is about the chosen method of hosting the web application; the eight chapter is written about testing of the website, both on front-end and back-end, also with the provided user feedback; the final, ninth chapter outlines possible future development directions of the website.

1. Terms and abbreviations

Term	Definition
API	Application Programming Interface
AWS	Amazon Web Services
EC2	Amazon Elastic Compute Cloud
DTO	Data Transfer Object
HTTP	Hypertext Transfer Protocol, a protocol used for transfering data on the internet
HTTPS	Secure version of HTTP
JSON	JavaScript Object Notation, lightweight data-interchange format
JWT	JSON Web Token
OOP	Object-oriented programming
RDS	Amazon Relational Database Service
S3	Amazon Simple Storage Service
SEO	Search Engine Optimisation
Swagger UI	UI that allows to visualize and interact with the API's resources
UI	User Interface
URL	Uniform Resource Locator
VPS	Virtual Private Server

2. Overview of food delivery platforms

In this chapter, the landscape of food delivery platforms is discussed. As well as the perspectives of customers and restaurants while also examining the inner workings of these platforms from a business standpoint. Additionally, with the discussion about the future of the industry and introduction of an innovative concept, for a fairer food delivery platform.

There are many food delivery websites that cater to a wide variety of restaurants, as well as websites dedicated to individual restaurants. Most of the big food delivery websites are intuitive, with modern design, big selection of restaurants and a large pool of the delivery drivers. In Estonia, the most famous food delivery websites and applications are: Wolt, Bolt Food, Fudy and Tellitoit [1] [2] [3] [4].

Meanwhile, the individual restaurants often have a poorly designed website, without the option of the order on the site and are not intuitive (for example, the website of Shaurma Kebab [5]). For these reasons, most individual restaurants use the services of the big food delivery companies. This brings both good and bad factors to the business.

2.1. Customer experience with food delivery platforms

2.1.1. Advantages

Food delivery platforms offer customers a wide variety of choices, so the customers can always find food for their liking. Additionally, the ordering of food is time-saving, which is crucial for some people. Because some of the restaurants are not offering the delivery and/or order option, the usage of food delivery website/app makes ordering food more convenient, fast and secure.

2.1.2. Disadvantages

Because companies make money on commissions, the price of each item can be significantly higher, than ordering directly from the restaurant (Figure 1 and Figure 2). However, it may not be the case at all times. For some users price might be a deciding factor in the decision of ordering or not ordering.

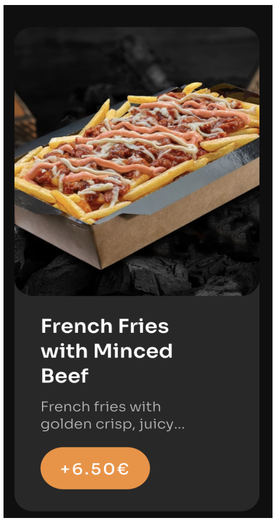

French Fries with Minced Beef
French fries with golden crisp, juicy...
+6.50€

Figure 1. Item from LIT Pirita website [6].

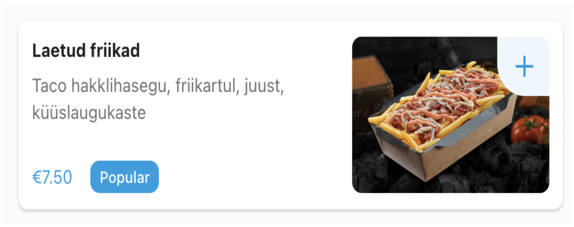


Figure 2. Item from Wolt website [7].

2.2. Restaurant experience with food delivery platforms

2.2.1. Advantages

On the good side, the restaurant owners get a reach to a wider client base, can increase their sales and can focus only on cooking, without worrying about the delivery [8].

2.2.2. Disadvantages

However, there are some consequences for this: food delivery companies take a commission on each product, causing the restaurants to scale up the prices, which can cause the customers not to consider buying at all. Also, the result of the way the food was delivered is also often projected to the restaurant, not the actual person, who delivered the food. The incidents that can occur: spilled drinks, damaged packaging, the food that fell apart, the long time of the delivery. All those factors can cause the irritation of the customer and the decision to not order the food from the said restaurant anymore.

2.3. Experience and business dynamics of food delivery platforms

2.3.1. Advantages

Having a big selection of known restaurant chains and local restaurants makes the customers accustomed with the brand of food delivery and using it often. If not on a daily, but on a weekly or monthly basis, gaining the loyal customer for the company.

2.3.2. Disadvantages

Food delivery companies are often not profitable. They compete with each other for restaurants, drivers and customers. Any special promotion that is launched in any website/app is to attract customers while losing money. Because of the competition, companies spend significant sums on marketing and other means to attract restaurants, drivers and customers [9]. Looking at the net income graphs of the USA companies DoorDash and Uber (UberEats) it is clear that they lose money, while maybe this not being the case with their counterparts in Europe, it is still a concerning matter [10] [11]. There is not much information about the net income of the food delivery companies in the free access on the web, but even the tech and food delivery giant in Estonia, Bolt, is reportedly losing money while growing the revenue, so it can be the case with its competitors too [12].

2.4. What can be done

The trend of ordering and delivering food is not slowing down, as well as the convenience of such methods of getting the food. So, there is no stopping the industry from evolving and right now there is just a very small chance that the industry will become obsolete [13].

Because of this, new food delivery websites, companies and applications will be made to provide services for the customers and restaurant owners.

The concept of the website in this thesis is to make an easy platform for small restaurants to display their dishes for the customers. The website will provide the service to migrate the data about restaurants, its menus and items into the website, managing this data and ordering process for the restaurant owners. For the customers the website will provide the display of the restaurants and ordering process as well.

There will be no commissions on items, but a fixed amount of money every month (subscription). This will allow restaurants to maintain lower prices while still paying for the service.

The restaurants can use their own delivery drivers or use the provided drivers by the service. While being fair to the customers and restaurants, it is also important to be fair for the delivery drivers. With the API of the website, the delivery fees will be calculated based on a set of rules: base fee rules (vehicle type and city) and weather fee rules (weather phenomenon, wind speed and temperature). The weather data will be fetched from Estonian Environment Agency Weather API every hour, saved and used to calculate the delivery fee for each order.

3. Website development plan and requirements

In this chapter, the plan of the development process is described. The plan includes the webpage prototype, the plan for the development and the requirements for the website.

3.1. Webpage prototype

It is important to have a prototype of the pages while building a website. Alongside the planning of the development of the website, the prototype was done and consisted of several pages.

3.1.1. Common web layout

Food delivery website is not a unique website that needs a special layout. The prototype of the website was developed based on suggestions provided by Mozilla and Digiworks. According to Mozilla [14] and Digiworks [15], the standard web page should contain the following parts:

1. Header

The part of the page which is always shown at the top of the page. It contains the information about the website, like the logo and the name and the navigation to the other pages.

2. Main content

The main content of the page that is unique to the current page.

3. Side menu

It can be a dropdown menu from the header or the part of the page that is always on the side. It is an optional part of the page, but usually it contains: additional information about the main content or information and links to the other pages.

4. Footer

The part of the page which is shown when the user scrolls at the bottom of the page. It contains additional information about the website, like contact information or legal pages, that is less prominent for the user to be used in the header.

5. Call To Action

Calls to action (hereafter, CTAs) are important to grab the user's attention and navigate the user to seeked pages. CTAs can be used as buttons, links within the text and as images.

6. Social links

Links to social media are popular and an important addition to the website. Usually being in the footer, the links provide a fast and reliable way to connect between the user and the website. Users that are interested will follow the social media further engaging with the website.

3.1.2. Prototype pages

The prototype consisted of 3 pages: home, user panels (admin, restaurant owner, customer) and user details. Each prototype page contained the default layout of the web page: the header (name, navigation bar, sign in), the main content, footer and the side menu. The side menu is a dropdown from the header, that has additional navigation to user-specific pages and to the logout page.

The home page prototype main content consisted of an image and the text for the image (Figure 3). The prototype was expanded with adding the new section to the home page.

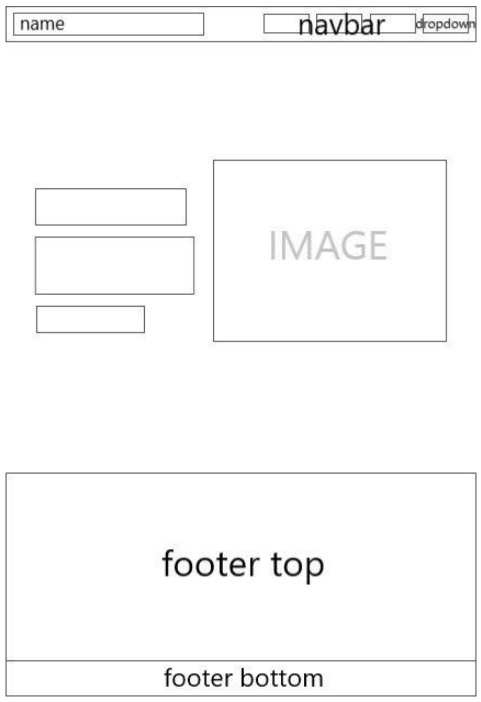


Figure 3. Prototype of home page.

The panel pages prototype main content consisted of buttons that linked to the different panel pages. The prototype was used in building the panels for admin and owner roles (Figure 4).

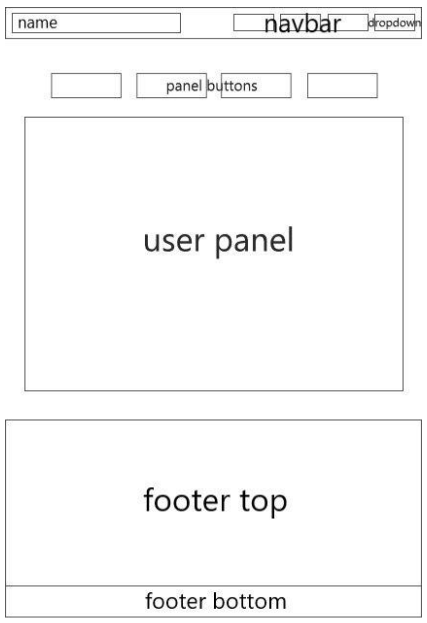


Figure 4. Prototype of panel pages.

The user details prototype main content consisted of two fields: user detail field and address detail field. The user detail field was used in the all role user detail pages, but the address field was only used in the customer role page (Figure 5).

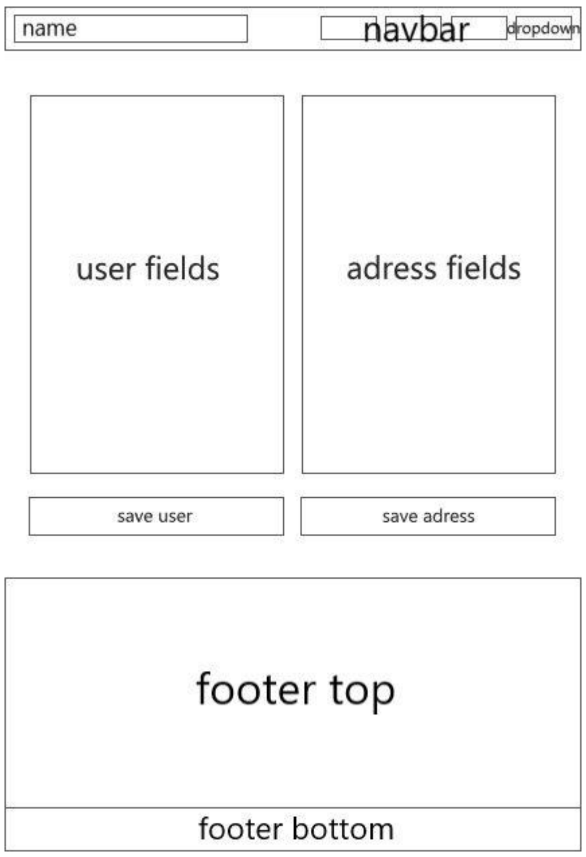


Figure 5. Prototype of user details page.

The color palette was built around the color red, because it is a widely used color in branding and marketing of the various food brands and restaurants [16]. The visual look of the website was done, for it to be minimal, easily readable and not so flashy, so the end user is not distracted and focused on the content and functionality of the website. It should be noticed, that the design and layout similar to the prototype was found on the internet. So, part of the code was used from the guide on the codevoweb webpage [17].The primary prototype was seen by the supervisor and discussed to add further improvements to the final design of the website.

3.2. Website development plan

The plan of the website development consisted of:

1. planning the security measures for the website;
2. plan and implementation of the back-end:
    - planning the architecture of the code for back-end;
    - writing the core part of the back-end (including security);
3. plan and implementation of the front-end:
    - planning the architecture of the code for front-end;
    - writing the core part of the front-end (including security);
4. the functionality of the webpage (adding features both on front- and back-end);

Firstly, the security measures for the website were planned. These measures should include user authentication and authorization via JWT and Spring Security, restriction of the endpoints and paths via role-based access on front- and back-end.

Secondly, the plan for the code of the back-end was done: components (users, restaurants, menus, items, orders), configurations, exceptions. Each component should have a controller, entity, DTO(s), repository and service classes. The planning was heavily influenced by the general approach to building such applications [18] [19]. After the planning stage, the core of the code was written, while keeping in mind that improvements/fixes will be needed in the future.

Thirdly, the plan of the code for the back-end was done: core (clients, guards, handlers, interceptors, services), models (the DTO(s) from back-end), shared pages (home, header, footer, authentication, other pages) and user pages (admin, owner, customer). The initial structure of the project was as in the official angular documentation, but further expansion was influenced by some of the angular file structure best practices [20] [21]. After the planning stage, the code was written and further improved in the following stages of the development.

The plan also provided information that after the main code would be written, the adding of the new features, fixing of the bugs, the validation of the data, the routing of the pages, and improvement to the code could be required. Thus, the plan also included time for that.

3.3. Functional requirements

Functional requirements describe the way in which the system interacts with its users by providing tasks or functions that it needs to perform [22], and the things that users can do on the pages [23]. These requirements often include detailed specifications of the user interactions with the website.

With the website that is for both owners of the restaurants, and the customers, it is important to have a separation between the pages and the functionality that users can use. Thus, the functional requirements for the website are provided below.

Site navigation:

- users can navigate through the page via links to the other pages that are placed in the header, footer or in the main container of the page;

User profile settings:

- all users can edit their profile information: name, username, password, telephone, address;

Admin panel:

- display and usage of owner panels - admin can use all the functionality provided in the owner panels;
- display of all users - admin can see the information about all users and be able to sort it by needed fields;
- display of all orders - admin can see the information about all the orders that were made on the site and be able to sort it by needed fields;
- accept/decline of owner applications - admin can see the page with the new users that want to become owners and decide whether to accept or decline the owners;

Owner panel:

- display the overview of the restaurants - owner can see the overview of his/her restaurants;
- add/edit/delete the restaurants, menus, items - owner can manage all the assets he/she owns;

- accept/decline incoming orders - owner can decide whether he/she accepts the orders of customers or no;
- display all orders - owner should be able to see all the incoming orders to his/her restaurants.

Customer panel:

- display the list of restaurant to choose from - customer can choose between all available restaurants to make an order;
- add/remove the items in the cart - customer can add or remove items from cart;
- checkout - customer can checkout and submit the order;
- display orders - customer can see all the orders that he/she has ordered.

Error handling:

- the errors that can occur must be shown to the user so the user is informed about the possible problems either from his/her side or the website side.

3.4. Non-functional requirements

The non-functional requirements mean the broad and abstract things that should be implemented on the website [23].

The non-functional requirements for the created website:

- mobile-friendly design - the website should be accessible from desktop and mobile, the layout must adjust to the screen size of the used device; the buttons must be appropriately sized for comfortable usage of the website;
- responsiveness - the website must load fast (under 5 seconds) and have quick response time (under 2 seconds) for all the actions that are performed;
- intuitive navigation - the website should have a logical flow and easily accessible links to pages; with that user will be able to quickly move between different sections of the website and understand how to navigate back;
- security - the security must be provided via secure storage of the passwords, validation, authentication and authorization;
- informativity - the informativity must be implemented with well-organized, helpful and easily understandable content of the pages.

4. Technologies used

In this chapter, the overview of the technologies used in the created website is shown. Additionally, the reason behind the choice of each technology is explained.

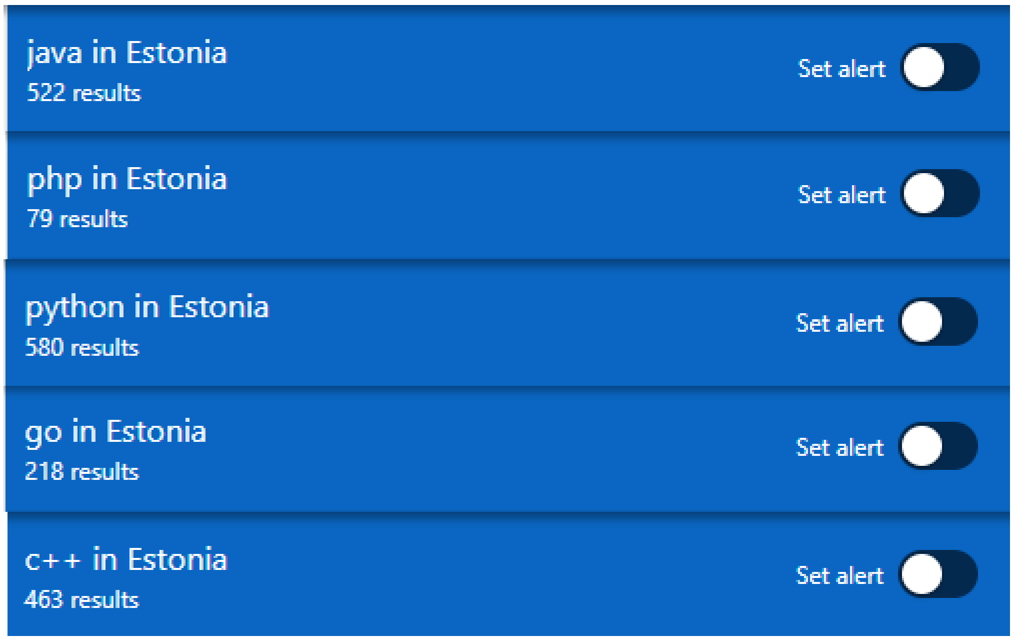
The choice of technology stack for a project plays a role in determining the extent of customization and the overall scope of the application that needs to be developed. Various combinations of technologies or tech stacks lead to the creation of applications that cater to a range of objectives [24].

When choosing the technology stack, it is important to choose the tools and technologies that one is comfortable with, while not binding oneself into strict bounds. For this project, the next stack was chosen: Java, Spring, Angular, PostgreSQL, AWS.

4.1. Java

Java is a widely used object-oriented programming (hereafter, OOP) language and software platform that runs on billions of devices. The rules and syntax of Java are based on the C and C++ languages [25]. Java is rather simple to learn and many universities are using Java as a preferred language to teach students OOP, including the University of Tartu [26] [27].

The decision to use Java was because the author is already familiar and comfortable with it, and Java is the platform independent, performant, scalable and is widely used as a preferred language in production at many software development companies [28]. While being on the market for a long time, Java is one of the most popular languages in the world, also being one of the most popular back-end languages [29] and one of the most seeked languages on the software engineer market in Estonia (Figure 6).


java in Estonia
522 results
Set alert

php in Estonia
79 results
Set alert

python in Estonia
580 results
Set alert

go in Estonia
218 results
Set alert

c++ in Estonia
463 results
Set alert

Figure 6. Job listings in Estonia on LinkedIn based on programming language [30].

For the current project, the package manager Maven was chosen.

4.1.1. Spring and its components

Spring, also referred to as Spring Framework, is a framework that provides comprehensive infrastructure support for developing Java applications. It has many modules, such as Spring

JDBC, Spring MVC, Spring Security, Spring Test and many more [31] [32]. It is a most popular framework for Java and is used to build production-grade applications [29] [33]. The decision to use Spring Framework was because it is the most developed, maintained and easy-to-use framework in the Java ecosystem.

Although Spring Framework is a capable framework, it still requires significant time and knowledge to configure, set up, and deploy the applications [31][33]. Spring Boot is a tool that makes it easy to create stand-alone, enterprise-level Spring applications that can run without any additional configuration. It takes an opinionated view of the Spring platform and third-party libraries and, thus, developers can get started with minimum requirements [31].

Spring Security is a powerful and highly customizable authentication and access-control framework. It is a standard for securing Spring-based applications, and is often used with the JWT to provide both authentication and authorization to Java applications [34].

4.2. Angular

Angular is a comprehensive development platform and framework, built on TypeScript for building client-side applications with HTML, CSS, and JavaScript/TypeScript [35]. It is one of the most popular frameworks to write front-end [29]. It is performant, cross-platform and is perfect for building the single-page applications. In addition, the choice was also due the fact that the author had prior experience with it.

4.3. PostgreSQL

PostgreSQL is an advanced, enterprise-class, and open-source relational database system [36]. It is the most widely adopted database in the world [29]. PostgreSQL is famous for its object-relational nature, so because of it, it's a great pairing with Java, which is also heavily object-oriented [37]. All these caused the decision to use PostgreSQL in the current project.

4.4. Docker

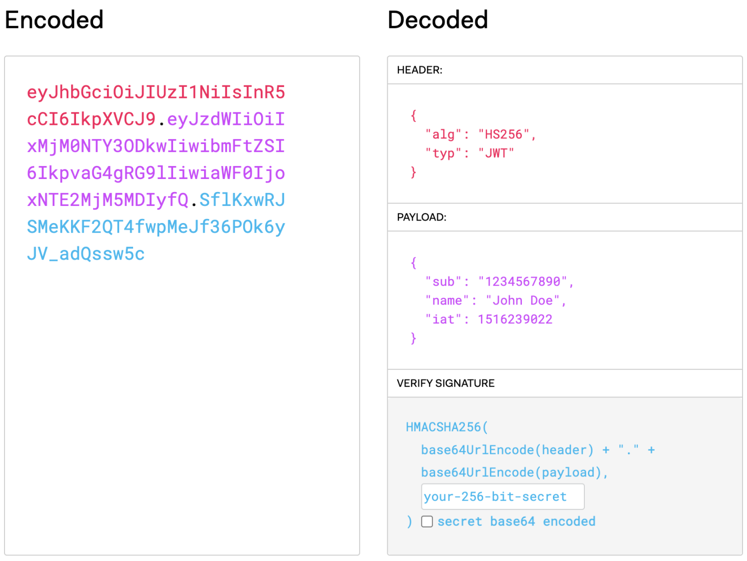
Docker is a software platform that allows one to build, test, and deploy applications quickly and almost everywhere. Docker packages software into units called containers that have everything the software needs to run including libraries, system tools, code, and runtime [38].

While there was no actual usage of Docker in the deployment and hosting of an application, the both front- and back-end have the configuration files for the website to be deployed using docker.

4.5. JWT

JSON Web Token (hereafter, JWT) is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed and the signature is provided inside the token [39].

The main usage of the JWT is authorization and authentication. The JWT can be used as a token that will be applied to validate the user, usually on the back-end side of the application. The JWT token can contain any fields, such as username, user id and others (Figure 10).


Encoded

```
eyJhbGciOiJIUzI1NiIsInR5
cCI6IkpXVCJ9.eyJzdWIiOiI
xMjM0NTY3ODkwIiwibmFtZSI
6IkpvaG4gRG9lIiwiaWF0Ijo
xNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6y
JV_adQssw5c
```

Decoded

HEADER:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Figure 10. The example of encoded and decoded JWT token [40].

4.6. AWS

Amazon Web Services (hereafter, AWS) is the world’s most comprehensive and broadly adopted cloud platform [29] [41]. AWS provides a wide variety of different services and solutions that can be used to cater to every need that companies and individuals need to implement in the cloud. In the project the following services were used.

Amazon Simple Storage Service (hereafter, S3) is an object storage service offering industry-leading scalability, data availability, security, and performance [42]. It can be used as just data storage, but also S3 bucket can be used as storage for a static website. On a static website, individual web pages include static content and might contain client-side scripts [43].

Elastic Beanstalk is a service for deploying and scaling web applications and services that also automatically handles the deployment [44]. In Elastic Beanstalk the environment is created, where user can put, for example, the .jar file after building the Java application in the local environment. Elastic Beanstalk will automatically deploy the code to the EC2 instance and it will be accessible to the static website in the S3 bucket [45]. Amazon Elastic Compute Cloud (EC2) is the service that is used to create and run virtual machines in the cloud (Amazon calls these virtual machines 'instances') [46].

Amazon Relational Database Service is a collection of managed services that makes it simple to set up, operate, and scale databases in the cloud [47]. It was used to host a PostgreSQL database.

5. Front-end

In this chapter, the simple explanation of the Angular structure and elements is provided alongside all the pages, components and services that were used to build a website.

The application structure in Angular mainly consists of components and dependency injections. Components can be pages, parts of the page, dialogs. Some dependency injections are the services that are injected into components. Others are the services that are declared in modules and run in the background [48]. For example, a request interceptor that intercepts every request that is made by the user and handles it.

5.1. Components

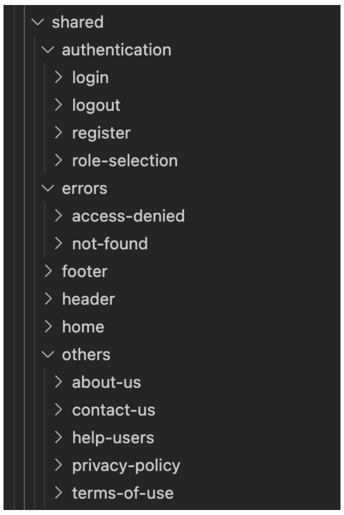
Components are the main building blocks for Angular applications. They are annotated with the "@Component". Each component consists of:

- an HTML template that declares what renders on the page;
- a TypeScript class that defines behavior of the component;
- a CSS selector that defines how the component is used in a template [49].

In this section the overview of almost all the components that the website consists of is provided.

5.1.1. Shared

The directory shared consists of components provided in Figure 11.



```
⌄ shared
  ⌄ authentication
    > login
    > logout
    > register
    > role-selection
  ⌄ errors
    > access-denied
    > not-found
  > footer
  > header
  > home
  ⌄ others
    > about-us
    > contact-us
    > help-users
    > privacy-policy
    > terms-of-use
```

Figure 11. Components of a shared directory.

5.1.1.1. Header

The header of the website is present at all times. It has the name of the website, the navigation bar that has links to the common pages and the button that triggers the dropdown of user-specific pages (Figure 12).

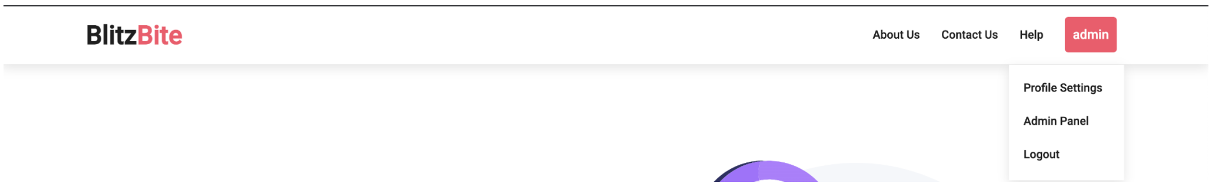
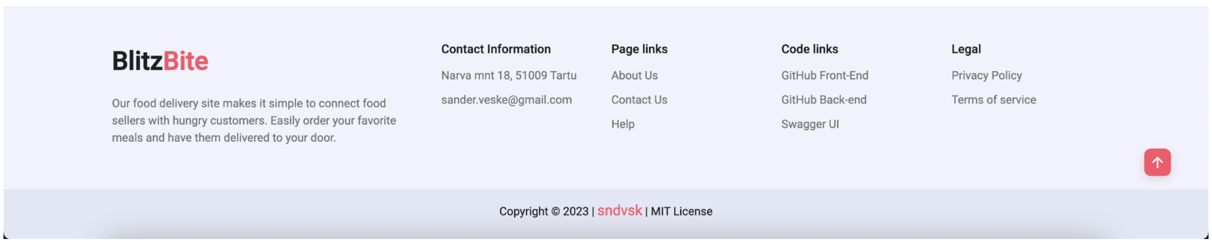


Figure 12. Header.

5.1.1.2. Footer

The footer of the website is presented when scrolled down to. Parts are placed as most of the real websites place them. The footer has the small description of the website, the contact information, the links to common pages, links to the code of the webpage, and the legal information (Figure 13).


BlitzBite

Our food delivery site makes it simple to connect food sellers with hungry customers. Easily order your favorite meals and have them delivered to your door.

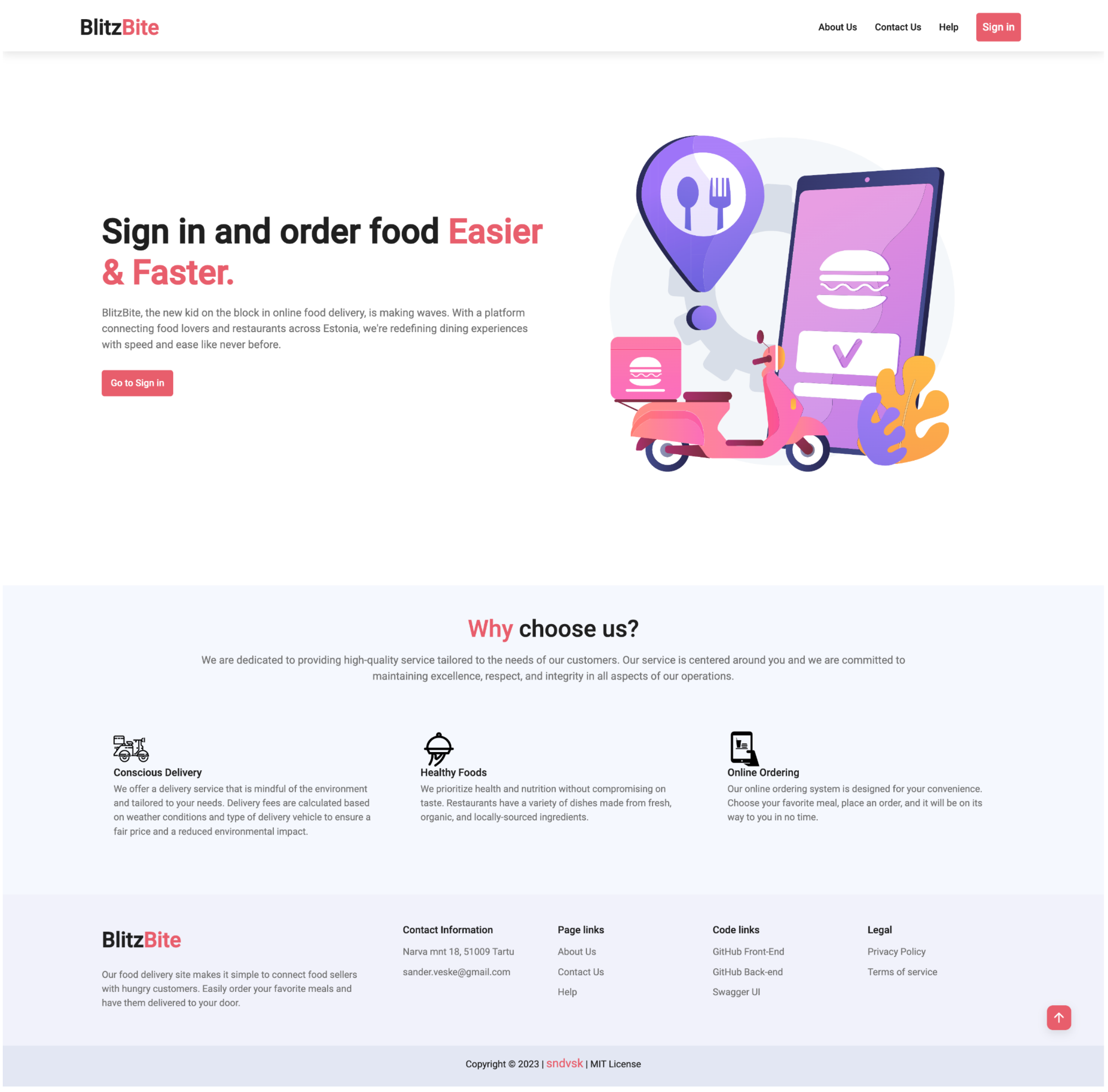
Contact Information	Page links	Code links	Legal
Narva mnt 18, 51009 Tartu	About Us	GitHub Front-End	Privacy Policy
sander.veske@gmail.com	Contact Us	GitHub Back-end	Terms of service
	Help	Swagger UI	

Copyright © 2023 | sndvsk | MIT License

Figure 13. Footer.

5.1.1.3. Home

The home page is the default page and is the page that the user sees when he/she visits the website. It has the user-specific information, depending on the role of the logged user. When the user is not logged in, the home page shows the main content with the “Sign in” button and text (Figure 14). This indicates that the user needs to log in. For the logged in customer there is the button and text to go to the restaurant menus, for the logged in owner the button to go to the owner panel and for the logged in admin the button redirect to the admin panel.



BlitzBite

About Us Contact Us Help Sign in

Sign in and order food Easier & Faster.

BlitzBite, the new kid on the block in online food delivery, is making waves. With a platform connecting food lovers and restaurants across Estonia, we're redefining dining experiences with speed and ease like never before.

Go to Sign in

Why choose us?

We are dedicated to providing high-quality service tailored to the needs of our customers. Our service is centered around you and we are committed to maintaining excellence, respect, and integrity in all aspects of our operations.

Conscious Delivery
We offer a delivery service that is mindful of the environment and tailored to your needs. Delivery fees are calculated based on weather conditions and type of delivery vehicle to ensure a fair price and a reduced environmental impact.

Healthy Foods
We prioritize health and nutrition without compromising on taste. Restaurants have a variety of dishes made from fresh, organic, and locally-sourced ingredients.

Online Ordering
Our online ordering system is designed for your convenience. Choose your favorite meal, place an order, and it will be on its way to you in no time.

BlitzBite

Our food delivery site makes it simple to connect food sellers with hungry customers. Easily order your favorite meals and have them delivered to your door.

Contact Information
Narva mnt 18, 51009 Tartu
sander.veske@gmail.com

Page links
About Us
Contact Us
Help

Code links
GitHub Front-End
GitHub Back-end
Swagger UI

Legal
Privacy Policy
Terms of service

Copyright © 2023 | sndvsk | MIT License

Figure 14. Home page for the unauthorized user.

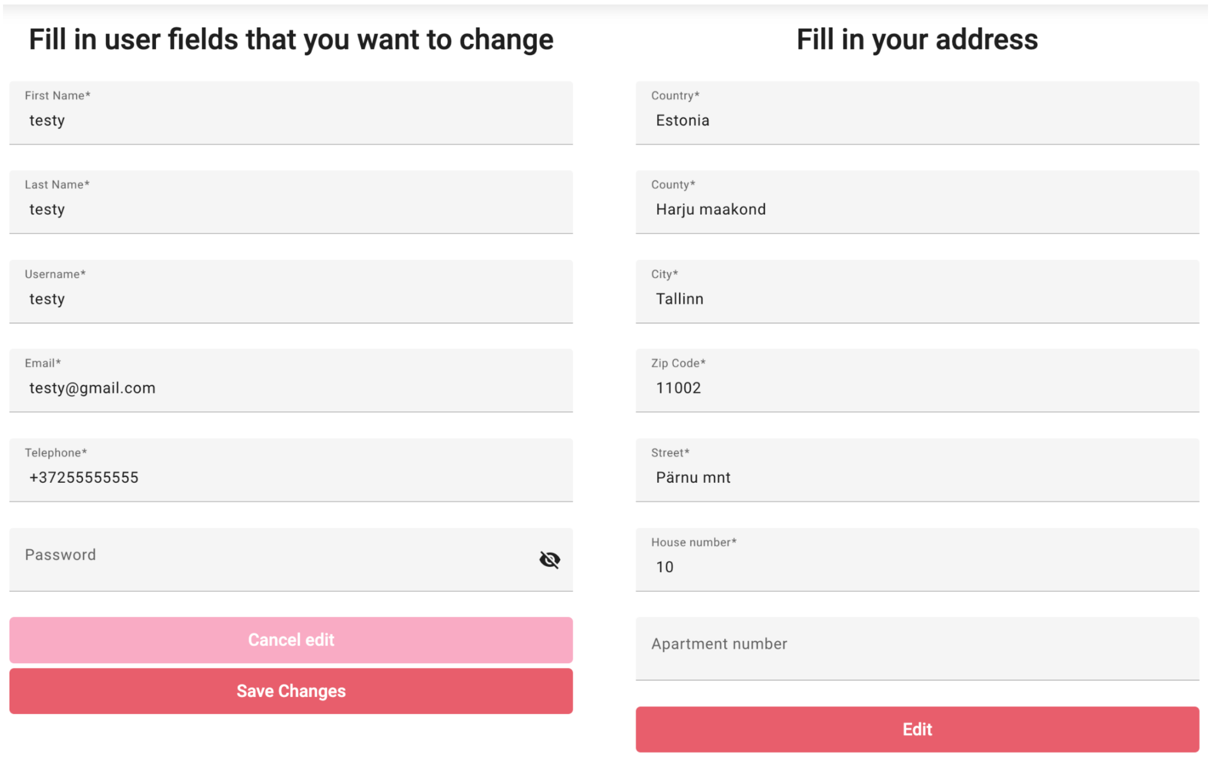
5.1.2. Users

Each type of the users (customer, owner, admin) has its unique pages and has a shared page for logged in users called “Profile Settings” in the header.

5.1.2.1. User details

The user details page is called “Profile Settings” and allows users to edit their personal information stored on the site. There are 2 different types of user details page: one is for the customers and another for the owners and admin. The user details page is the one with the address details (Figure 15) and the page without the address field is for owners and admin.

The functionality of the page allows the user to change the details. The user fields and address fields have validation on every field, so the text can’t be added to the phone number and the numbers can’t be added to the first or last name.


Fill in user fields that you want to change

First Name*	testy
Last Name*	testy
Username*	testy
Email*	testy@gmail.com
Telephone*	+37255555555
Password	

Fill in your address

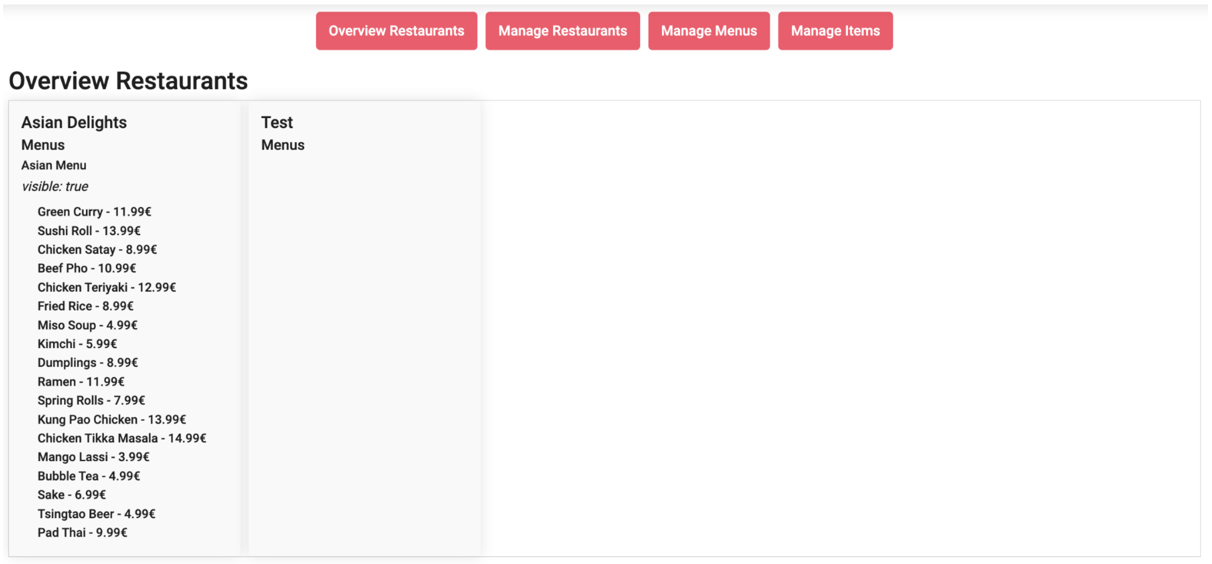
Country*	Estonia
County*	Harju maakond
City*	Tallinn
Zip Code*	11002
Street*	Pärnu mnt
House number*	10
Apartment number	

Figure 15. User details page for the customers.

5.1.2.2. Owner

On the website, under owner pages, there are the following pages and components. The “Owner Panel” page, which has “Overview Restaurants”, “Manage Restaurants”, “Manage Menus” and “Manage Items” components, and the “Orders” page.

The component “Overview Restaurants” provides the grid list of the restaurants that the owner has (Figure 16).



Overview Restaurants

Asian Delights
Menus
Asian Menu
visible: true

Green Curry - 11.99€
Sushi Roll - 13.99€
Chicken Satay - 8.99€
Beef Pho - 10.99€
Chicken Teriyaki - 12.99€
Fried Rice - 8.99€
Miso Soup - 4.99€
Kimchi - 5.99€
Dumplings - 8.99€
Ramen - 11.99€
Spring Rolls - 7.99€
Kung Pao Chicken - 13.99€
Chicken Tikka Masala - 14.99€
Mango Lassi - 3.99€
Bubble Tea - 4.99€
Sake - 6.99€
Tsingtao Beer - 4.99€
Pad Thai - 9.99€

Test
Menus

Figure 16. Owner panel: “Overview Restaurants” component.

The component “Manage Restaurants” has the list of the restaurants the owner has. The owner is able to change details of the restaurants, add new restaurants and delete restaurants (Figure 17). When pressing the add or edit restaurant the page similar to “Profile Settings” opens where user can add or edit the necessary fields. When pressing “Delete Restaurant” the dialog window asking the user to confirm deletion of the selected restaurant pops up. This approach helps prevent the owner from deleting the restaurant on accident.

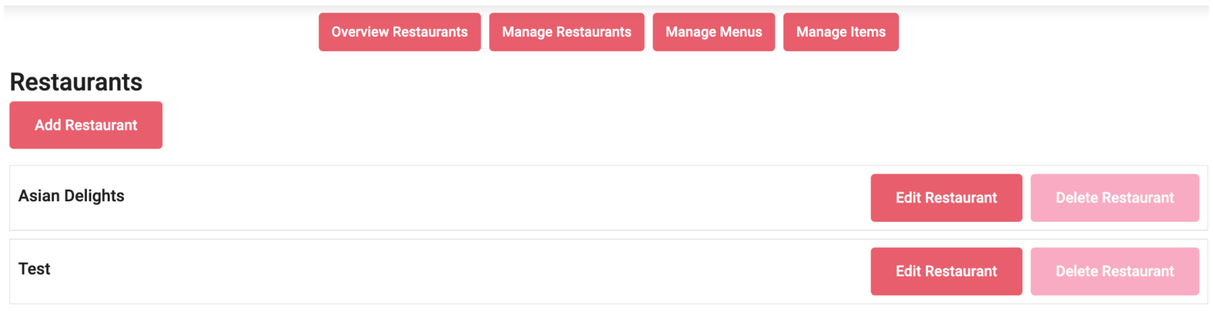


Figure 17. Owner panel: “Manage Restaurants” component.

The component “Manage Menus” has the list of the menus that the owner has. The owner is able to change details of the menus, add new menus, delete menus, assign menus to restaurants, remove menus from restaurants (Figure 18). When pressing the add or edit menu the page similar to “Profile Settings” opens and the owner can add or edit the necessary fields. When pressing “Delete Menu” the dialog window asking the owner to confirm deletion pops up. This approach helps prevents the owner from deleting the menu on accident. When pressing “Assign Restaurant” the dialog window pops up and the owner is able to choose which restaurant will have this menu. It is important to keep in mind that the menu is not visible to the users by default and the owner needs to manually make it visible through the “Edit Menu” option. When pressing “Remove from Restaurant” there is no pop up warning window for the owner, because it is not an irreversible action and the owner simply can assign the restaurant back. With the current implementation, the menu can belong only to one restaurant.

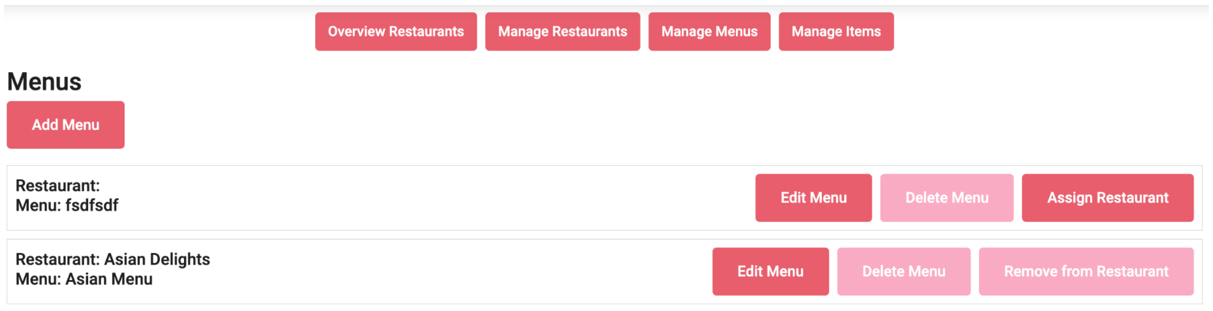
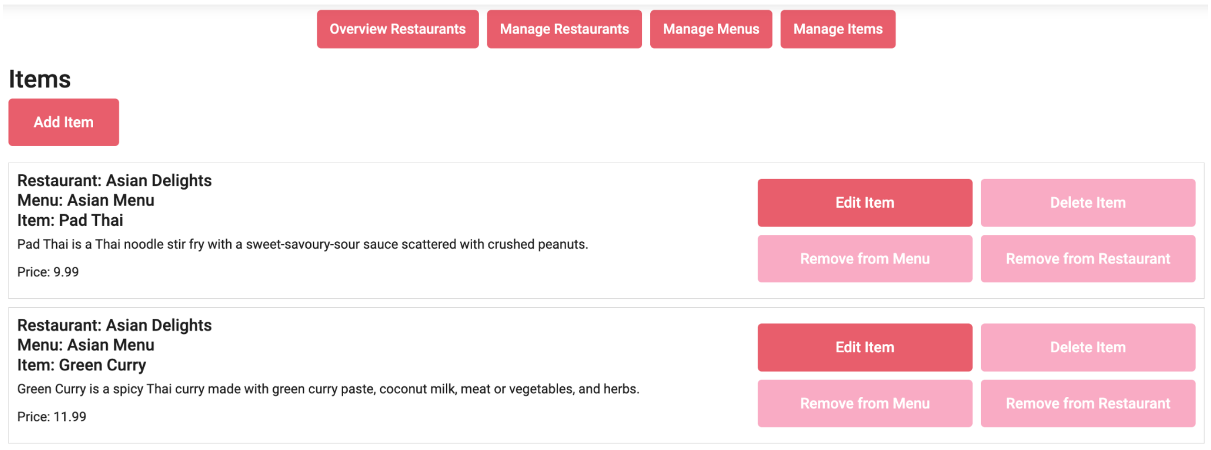


Figure 18. Owner panel: “Manage Menus” component.

The component “Manage Items” has the list of the items that the owner has. The owner is able to change details of the items, add new items, delete items, assign items to menus, remove items from menus, assign menus to restaurant and remove items from restaurant (Figure 19). When pressing the add or edit item the page similar to “Profile Settings” opens and the owner can add or edit the necessary fields. When pressing “Delete Item” the dialog window asking the owner to confirm deletion pops up. This approach helps prevent the owner from deleting the item on accident. When pressing “Assign Restaurant” the dialog window pops up and the owner is able to choose which restaurant will have this item. When pressing “Remove from Restaurant” there is no pop up warning window for the owner, because it is not an irreversible action and the owner simply can assign the restaurant back. Nevertheless, the item will be automatically removed from the assigned menu as well. When pressing “Remove from Menu” there is no pop up warning window similar to “Remove from Restaurant“ and the restaurant will be left assigned as it was. When pressing “Assign to Menu” or “Assign to Restaurant” the dialog window pops up and the owner can choose the menu or the restaurant from those that he/she owns, if the assigned menu was chosen, the restaurant will be assigned automatically. With the current implementation, the item can belong only to one restaurant and only one menu.

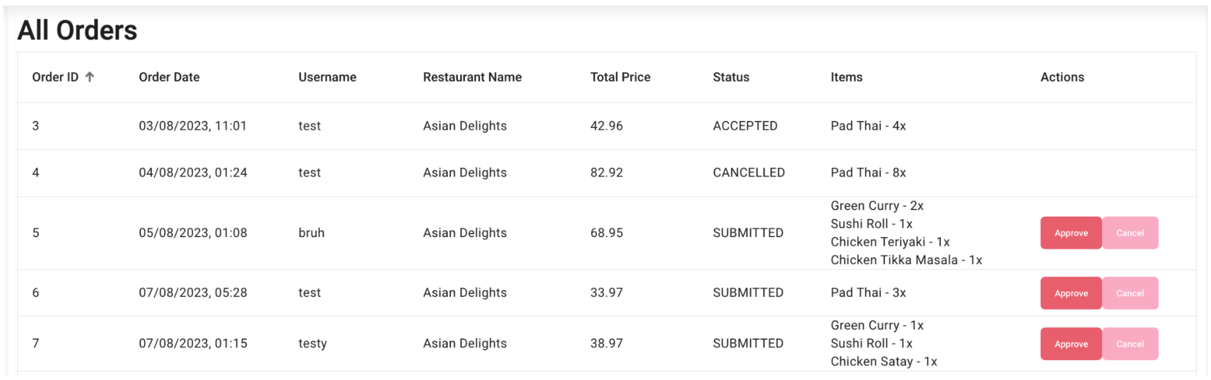


Restaurant: Asian Delights
Menu: Asian Menu
Item: Pad Thai
Pad Thai is a Thai noodle stir fry with a sweet-savoury-sour sauce scattered with crushed peanuts.
Price: 9.99

Restaurant: Asian Delights
Menu: Asian Menu
Item: Green Curry
Green Curry is a spicy Thai curry made with green curry paste, coconut milk, meat or vegetables, and herbs.
Price: 11.99

Figure 19. Owner panel: “Manage Items” component.

The page “Orders” allows the restaurant owner to see all the orders that were submitted by the customers of his/her restaurants. The owner can sort the orders by id, order date, username, restaurant name, price and status. In the actions column the owner can either choose to fulfill the order by pressing “Approve” or decline by pressing “Cancel” (Figure 20).



All Orders

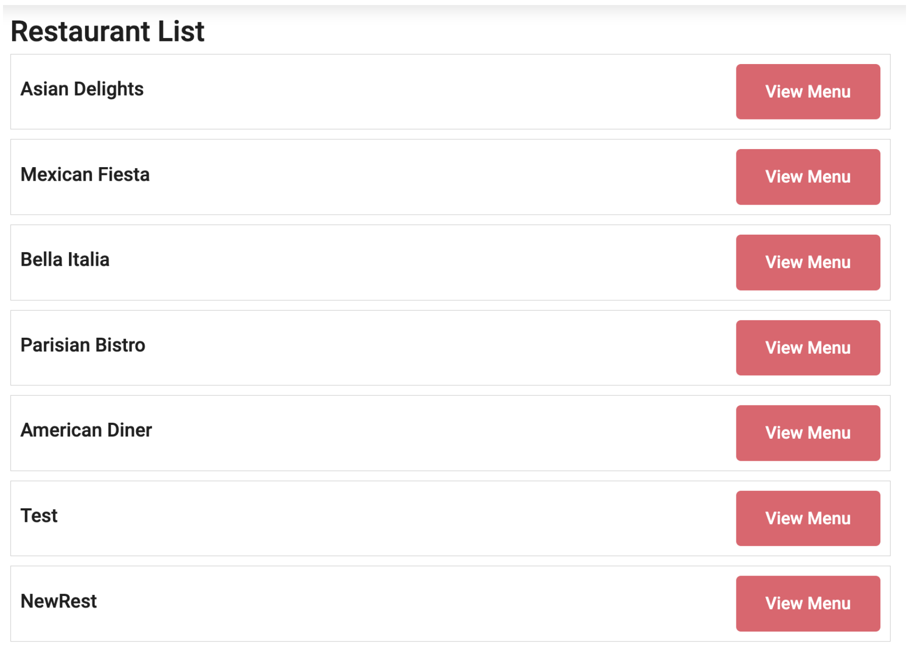
Order ID ↑	Order Date	Username	Restaurant Name	Total Price	Status	Items	Actions
3	03/08/2023, 11:01	test	Asian Delights	42.96	ACCEPTED	Pad Thai - 4x	
4	04/08/2023, 01:24	test	Asian Delights	82.92	CANCELLED	Pad Thai - 8x	
5	05/08/2023, 01:08	bruh	Asian Delights	68.95	SUBMITTED	Green Curry - 2x Sushi Roll - 1x Chicken Teriyaki - 1x Chicken Tikka Masala - 1x	Approve Cancel
6	07/08/2023, 05:28	test	Asian Delights	33.97	SUBMITTED	Pad Thai - 3x	Approve Cancel
7	07/08/2023, 01:15	testy	Asian Delights	38.97	SUBMITTED	Green Curry - 1x Sushi Roll - 1x Chicken Satay - 1x	Approve Cancel

Figure 20. Owner orders table page.

5.1.2.3. Customer

On the website, under customer pages, there are the following pages and components. The "Restaurants" page, the page of the chosen restaurant in the "Restaurants", the "Checkout" page and the "Orders" page.

The page "Restaurants" allows the customer to decide on the restaurant that he/she wants to make an order in (Figure 21). The click on the "View Menu" redirects the customer to the page of the chosen restaurant. With the current implementation, the customer can simultaneously order from one restaurant at a time.



Restaurant List

Asian Delights

Mexican Fiesta

Bella Italia

Parisian Bistro

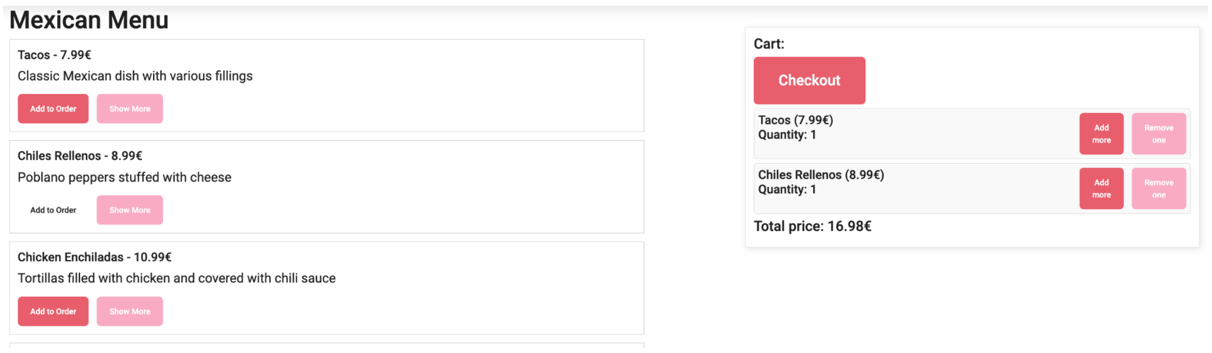
American Diner

Test

NewRest

Figure 21. Restaurant list page.

On the page of the chosen restaurant, the user sees all the menus of the restaurant alongside the items of displayed menus (Figure 22). If the restaurant does not have any items or the menus are empty, the text stating the emptiness of the restaurant will be shown on the page. On this page the user can add items to the cart by pressing "Add to Order". By pressing the "Show More", dialog window, displays all the info about the chosen item (Figure 23). In the cart, the customer can add more of the already chosen items or remove them one by one. After pressing the "Checkout" button, the customer is redirected to the checkout page.



Mexican Menu

Tacos - 7.99€
Classic Mexican dish with various fillings

Chiles Rellenos - 8.99€
Poblano peppers stuffed with cheese

Chicken Enchiladas - 10.99€
Tortillas filled with chicken and covered with chili sauce

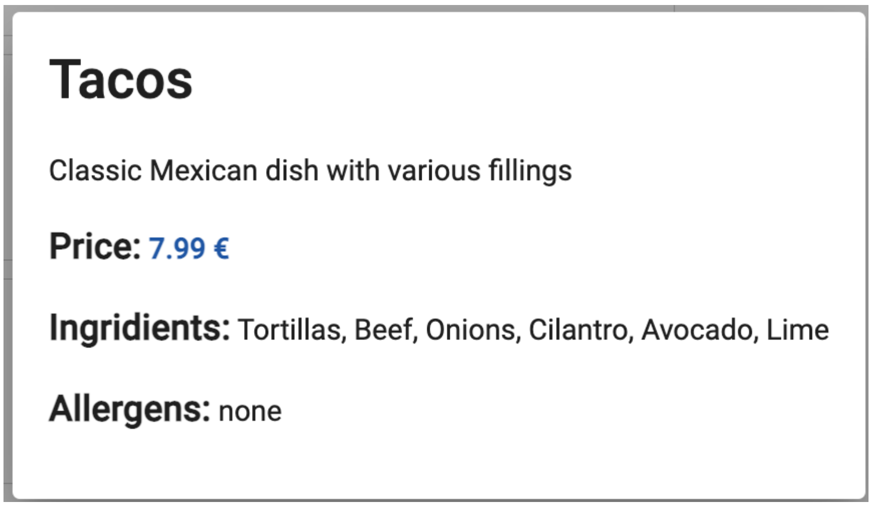
Cart:

Tacos (7.99€)
Quantity: 1

Chiles Rellenos (8.99€)
Quantity: 1

Total price: 16.98€

Figure 22. Restaurant menu and item page.



Tacos

Classic Mexican dish with various fillings

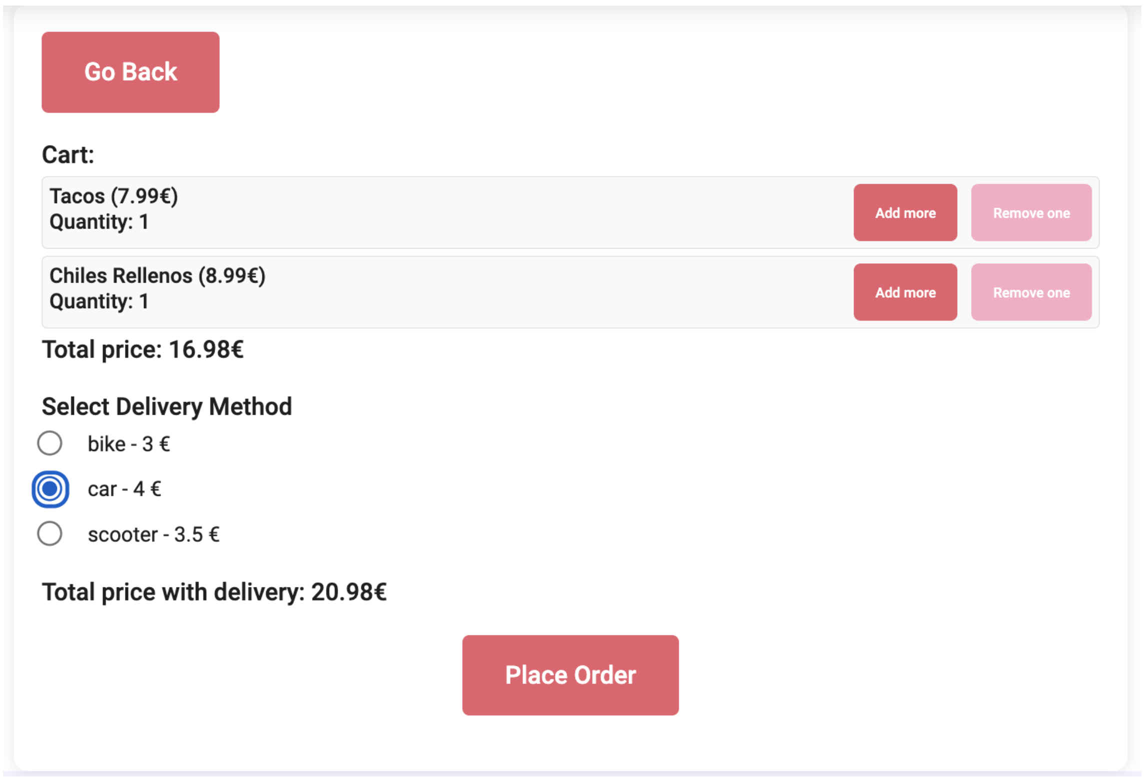
Price: 7.99 €

Ingridients: Tortillas, Beef, Onions, Cilantro, Avocado, Lime

Allergens: none

Figure 23. “Show More” dialog.

The checkout page consists of several elements. The “Go Back” button takes the customer to the last page where he/she was. The main part is the chosen item list in cart, with the same functionality as in the restaurant page (Figure 24). Also, selection of the delivery method is provided. The values of the price are not fixed and are calculated based on the set of rules: base fee rules (city and delivery method) plus weather fee rules (weather phenomenon, wind speed and temperature).



Cart:

Tacos (7.99€)
Quantity: 1

Chiles Rellenos (8.99€)
Quantity: 1

Total price: 16.98€

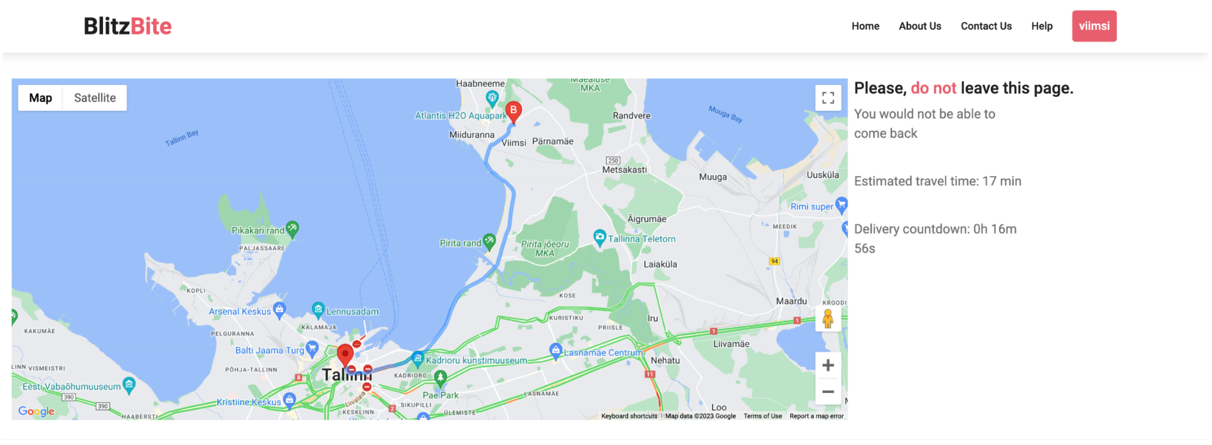
Select Delivery Method

bike - 3 € [✓ car - 4 €] scooter - 3.5 €

Total price with delivery: 20.98€

Figure 24. Checkout page.

When a customer presses the “Place Order” button, the order is submitted and the customer is taken to the page with the direction from the restaurant to the customer saved address (Figure 25). The page also shows the time of delivery, which is provided by the Google Directions API.



Please, do not leave this page.

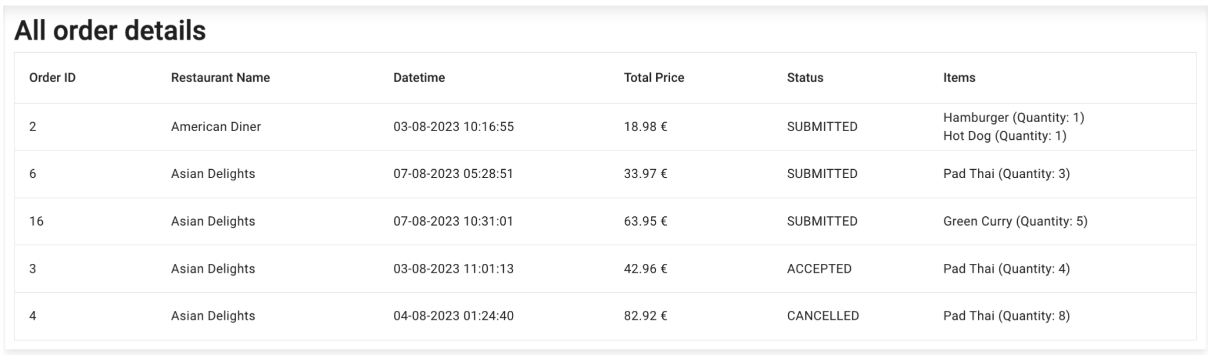
You would not be able to come back

Estimated travel time: 17 min

Delivery countdown: 0h 16m 56s

Figure 25. Directions page.

Also, the customer has the page with the table of all orders that the user placed (Figure 26). The table can be sorted by order id, restaurant name, datetime, price and status.



All order details

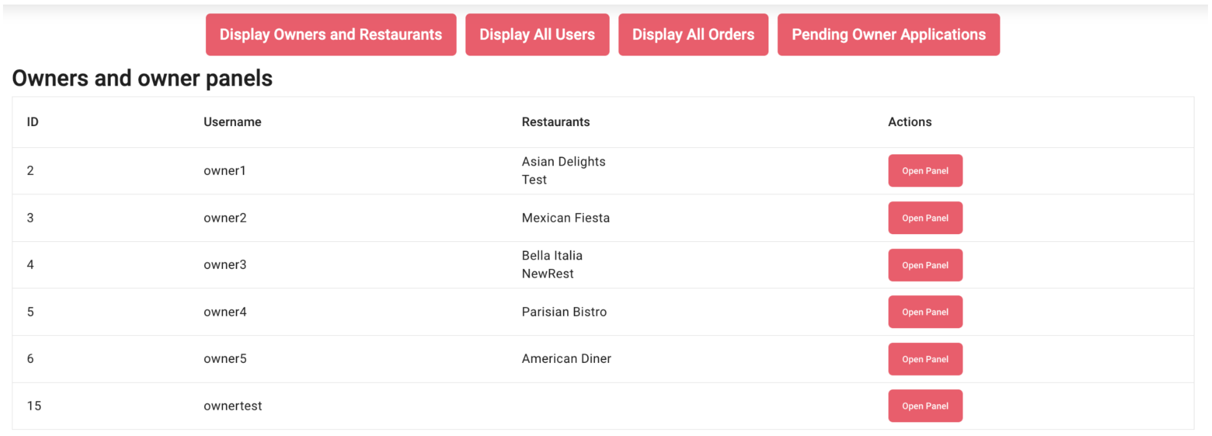
Order ID	Restaurant Name	Datetime	Total Price	Status	Items
2	American Diner	03-08-2023 10:16:55	18.98 €	SUBMITTED	Hamburger (Quantity: 1) Hot Dog (Quantity: 1)
6	Asian Delights	07-08-2023 05:28:51	33.97 €	SUBMITTED	Pad Thai (Quantity: 3)
16	Asian Delights	07-08-2023 10:31:01	63.95 €	SUBMITTED	Green Curry (Quantity: 5)
3	Asian Delights	03-08-2023 11:01:13	42.96 €	ACCEPTED	Pad Thai (Quantity: 4)
4	Asian Delights	04-08-2023 01:24:40	82.92 €	CANCELLED	Pad Thai (Quantity: 8)

Figure 26. Customer order details table page.

5.2.2.4. Admin

On the website, under admin pages, there are the following pages and components. The “Admin Panel” page, which has “Display Owners and Restaurants” component, “Display All Users” component, “Display All Orders” component, “Manage Items” components and “Pending Owner Applications” component.

In the display of owners and restaurants component, the admin can see all owners as well as the restaurants that owners have (Figure 27). By pressing “Open Panel” in the actions column, the admin is redirected to the owner panel of the selected owner and can use every function that the owner panel has. The table also can be sorted by owner id, owner username and restaurants.

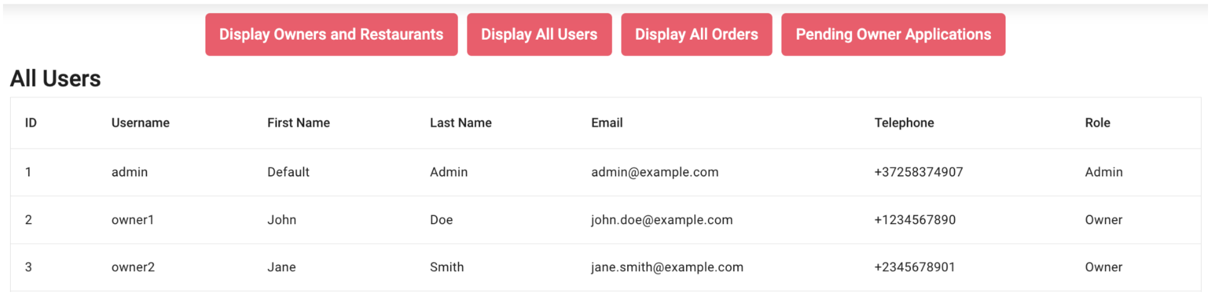


Owners and owner panels

ID	Username	Restaurants	Actions
2	owner1	Asian Delights Test	Open Panel
3	owner2	Mexican Fiesta	Open Panel
4	owner3	Bella Italia NewRest	Open Panel
5	owner4	Parisian Bistro	Open Panel
6	owner5	American Diner	Open Panel
15	ownertest		Open Panel

Figure 27. Admin panel: “Display Owners and Restaurants” table component.

In the display all users component the admin can see all the users that are registered on the site (Figure 28). The table can be sorted by the user id, username, first name, last name, email, telephone and role.

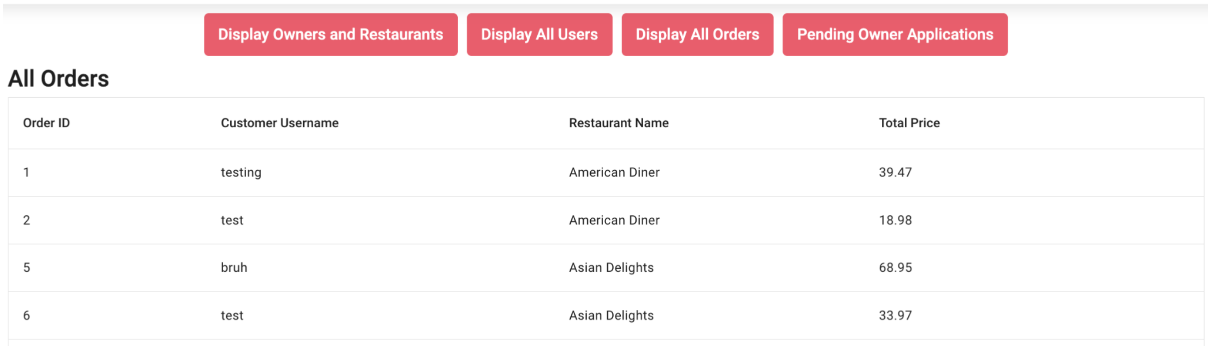


All Users

ID	Username	First Name	Last Name	Email	Telephone	Role
1	admin	Default	Admin	admin@example.com	+37258374907	Admin
2	owner1	John	Doe	john.doe@example.com	+1234567890	Owner
3	owner2	Jane	Smith	jane.smith@example.com	+2345678901	Owner

Figure 28. Admin panel: “Display All Users” component.

In the display all orders component the admin can see all orders that were done on the site (Figure 29). The table can be sorted by order id, customer username, restaurant name and total price.

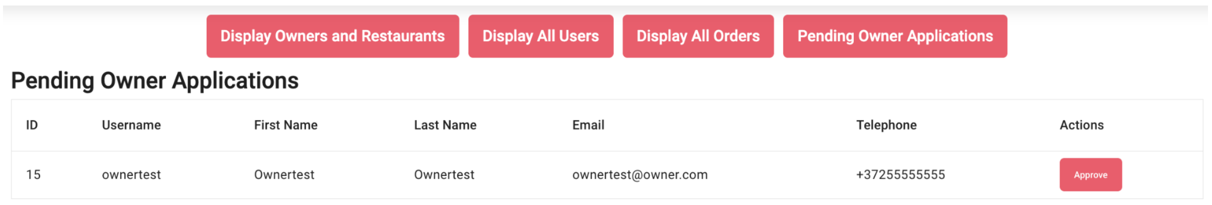


All Orders

Order ID	Customer Username	Restaurant Name	Total Price
1	testing	American Diner	39.47
2	test	American Diner	18.98
5	bruh	Asian Delights	68.95
6	test	Asian Delights	33.97

Figure 29. Admin panel: “Display All Orders” component.

In the pending owner applications component the admin can see and manage the approval of the owner accounts (Figure 30). By pressing the button “Approve” in the actions column, the admin approves the owner and the owner can start using the owner panel. There was functionality for rejecting the owner, but it seemed unnecessary, so it was commented out.



Pending Owner Applications

ID	Username	First Name	Last Name	Email	Telephone	Actions
15	ownertest	Ownertest	Ownertest	ownertest@owner.com	+37255555555	Approve

Figure 30. Admin panel: “Pending Owner Applications” component.

5.1.3. Others

5.1.3.1. Authentication

Authentication pages encompass register, login and logout. The role selection is a component of the register page.

When the user goes to the registration page, he/she chooses the role (Figure 31) and then fills out the registration form fields (Figure 32). All registration form fields are with validation, so the faulty data is not accepted. After registering, the user is redirected to the home page. If it is a user with a customer role, the notification about providing the information about address is necessary to make an order.

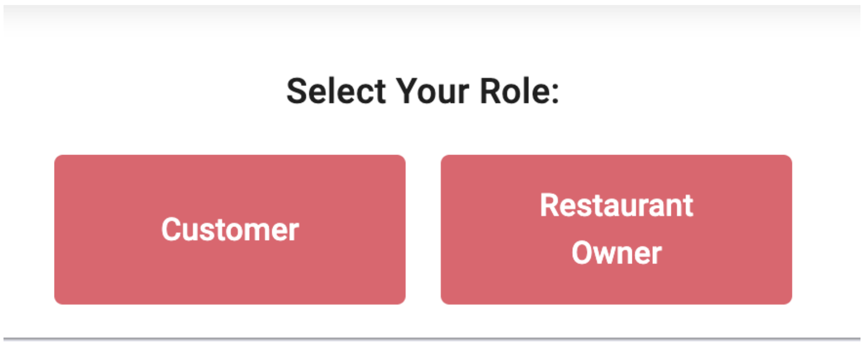


Figure 31. Role selection component.

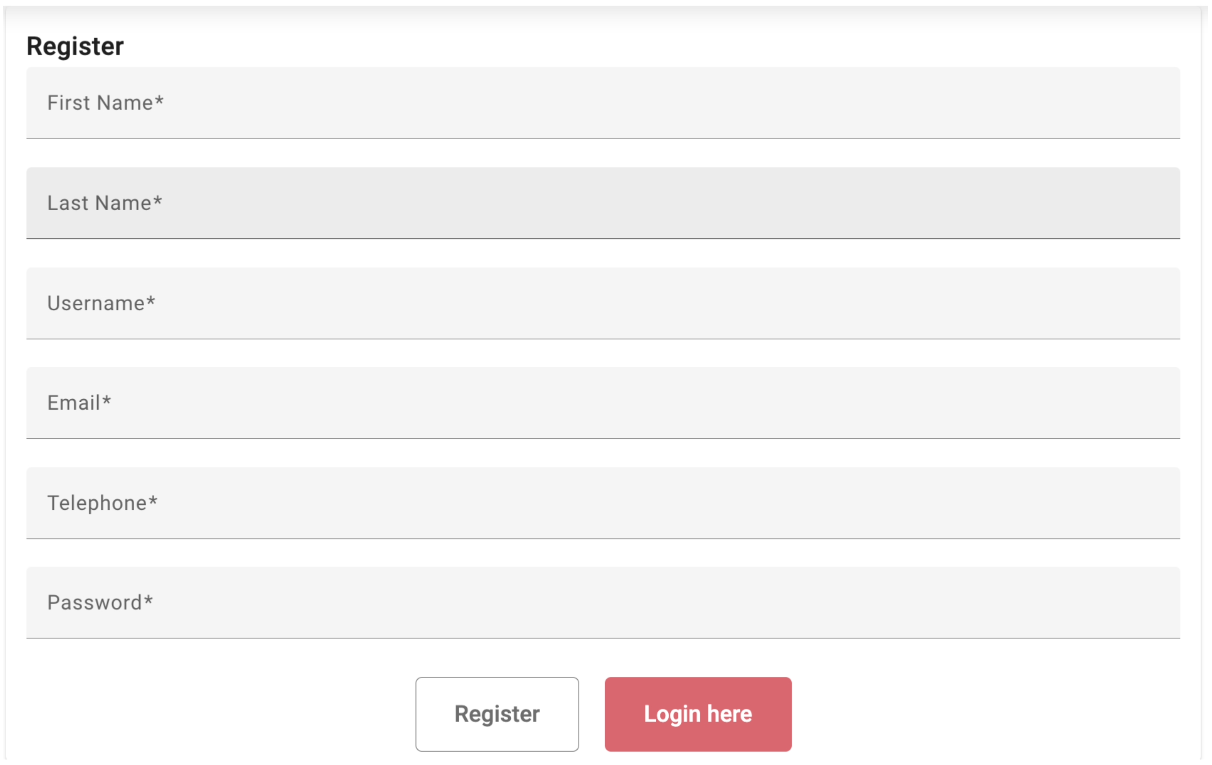


Figure 32. Form of registration page.

When the user goes to the login page, he/she needs to fill the login fields that are validated to login (Figure 33). After logging in, the user is redirected to the home page.

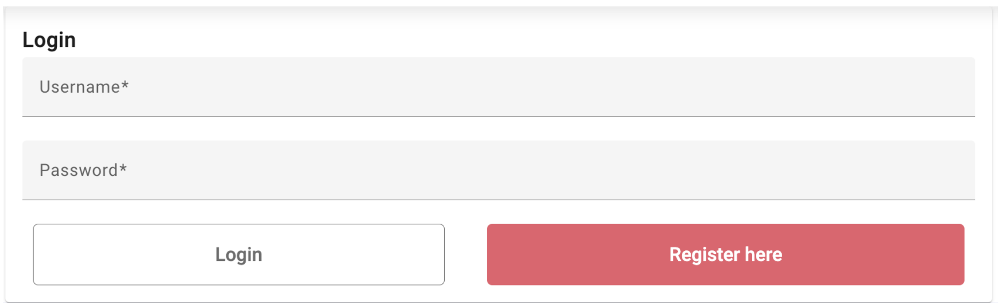


Figure 33. Form of login page.

When the user goes to the logout page, the user is asked whether he/she wants to log out (Figure 34). If the user logs out, the local storage will be cleared.

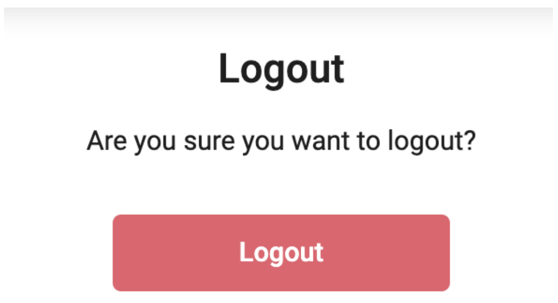


Figure 34. Logout page.

5.1.3.2. Errors

There are two error pages: access denied and not found. The access denied page is shown when a user without the necessary permissions tries to access the restricted page (Figure 35, left). For example, a user with a customer role tries to go to the admin panel. The not found page is shown when the user tries to open a page that does not exist (Figure 35, right).

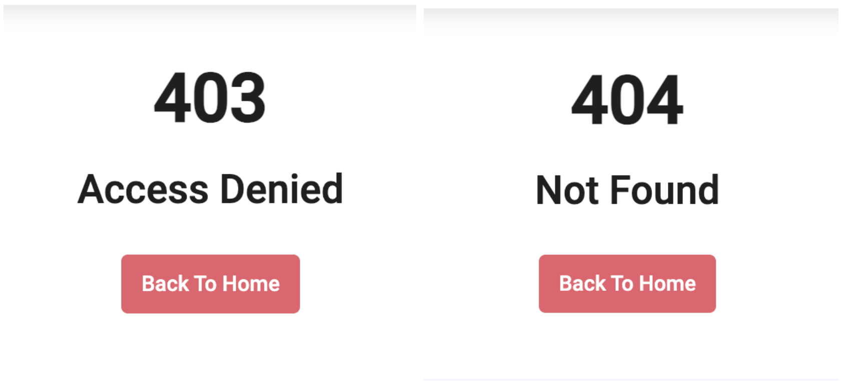


Figure 35. "Access Denied" page on the left and "Not Found" page on the right.

5.1.3.3. Contact, about, legal, help pages

The "About us", "Contact", "Help" and legal pages are placed in the footer with the helpful information to the user.

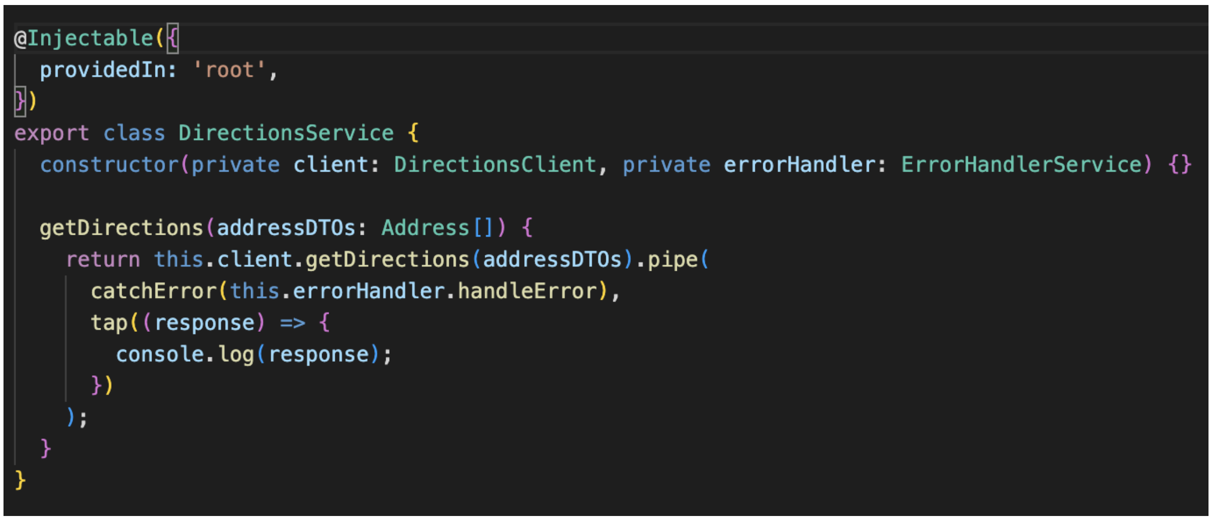
The "About us" page provides the brief information about the website company. The "Contact" page provides the necessary contact information so the users that want to have a chat with the company can do that. "Help" page provides the guide for each user role, the admins, owners and customers are given the necessary information to use the site.

5.2. Dependency injections

Dependency Injection (hereafter, DI) is a core design pattern used by Angular. DI promotes a modular and maintainable code structure. It allows classes with Angular-specific decorators, such as Components, Directives, Pipes, and Injectables, to receive dependencies. This system allows the injectables to be used inside other injectables to provide the needed functionality [50].

5.2.1. Services and clients

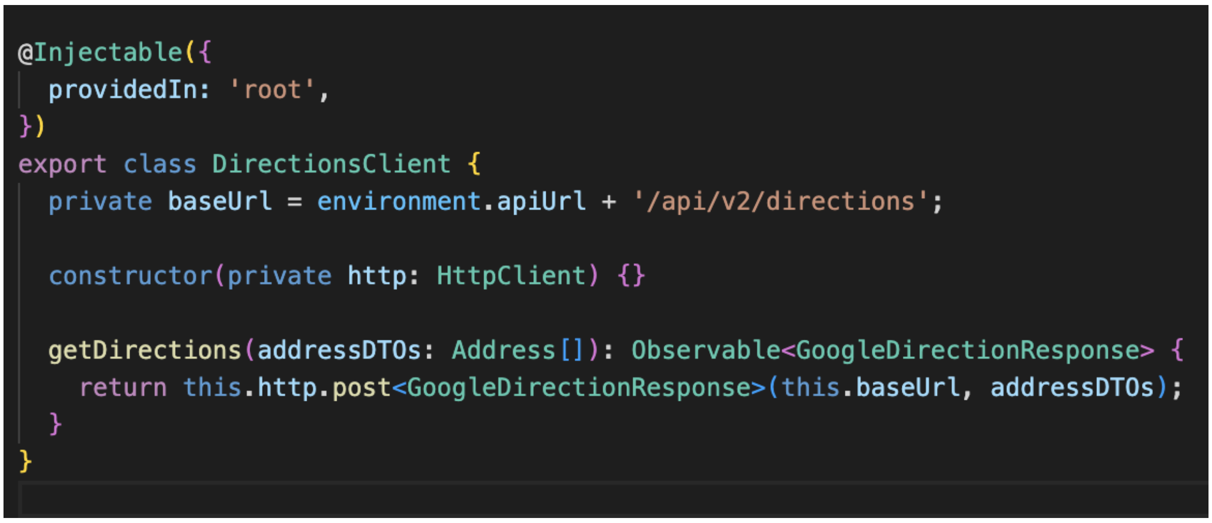
The services provide the communication with the API or maintain the state of some objects [51]. In the code, the author decided to separate the service into service and client. The service is responsible for handling the errors and passing the data to the component (Figure 36). The client is communicating with the API and passes the data to the service (Figure 37).



```
@Injectable({
  providedIn: 'root',
})
export class DirectionsService {
  constructor(private client: DirectionsClient, private errorHandler: ErrorHandlerService) {}

  getDirections(addressDTOs: Address[]) {
    return this.client.getDirections(addressDTOs).pipe(
      catchError(this.errorHandler.handleError),
      tap((response) => {
        console.log(response);
      })
    );
  }
}
```

Figure 36. Directions service.



```
@Injectable({
  providedIn: 'root',
})
export class DirectionsClient {
  private baseUrl = environment.apiUrl + '/api/v2/directions';

  constructor(private http: HttpClient) {}

  getDirections(addressDTOs: Address[]): Observable<GoogleDirectionResponse> {
    return this.http.post<GoogleDirectionResponse>(this.baseUrl, addressDTOs);
  }
}
```

Figure 37. Directions client.

The list of services and clients:

- Restaurant - get/add/update/delete (address included into add/edit);
- Item - get/add/update/delete, assign and remove from restaurant/menu;
- Menu - get/add/update/delete, assign and remove from restaurant, toggle visibility;
- Order - get/add/update/delete;
- Admin - approve/reject owner, get owners;
- Customer - get/add/update address;
- Owner - get, approve and reject order;
- Users - get/update;
- Authentication - login, register (add user), logout, refresh token and many helper methods;
- DeliveryFee - get;
- Directions - get.

Also there are services without clients. Those services are working inside the application and do not communicate with the API. Those services are:

- ErrorHandler - displays errors to the users with the help of ErrorHandler and HotToastService(import);

- CartState - allows the cart to be saved across every page;
- DeliveryFeeState - allows the delivery fee not to be calculated after leaving the cart and going there again, causing the unnecessary requests to the API;
- Navigation - sets and gets the state from checkout page to directions page.

5.2.2. Guards

The guards are interfaces that provide route based access to the pages when used in pair with routing modules [52].
The guard that are used in the current project are as following:

- Authentication - prevents authenticated users from accessing the login and register pages and unauthorized users from accessing the logout page;
- Routing - provides role-based access to pages. For example, only admins can use admin paths. Also prevents unauthorized users from accessing the user pages;
- CheckoutNavigation - makes sure that the only place from where users can access the directions page is from pressing the button "Place Order" in the checkout page.

5.2.3. Interceptors

The interceptors are used to intercept the HTTP requests and responses and handle them according to implemented logic [53].

The interceptors that are used in the current project are as following:

- Error - catches errors and checks whether the user is authorized and logs the user out if the user is not authorized, refreshes the access token if the user is authorized;
- JSONP - is used to send requests to the Google Maps API;
- Request - is used to set the authorization header on every request, instead of specifying it in the clients.

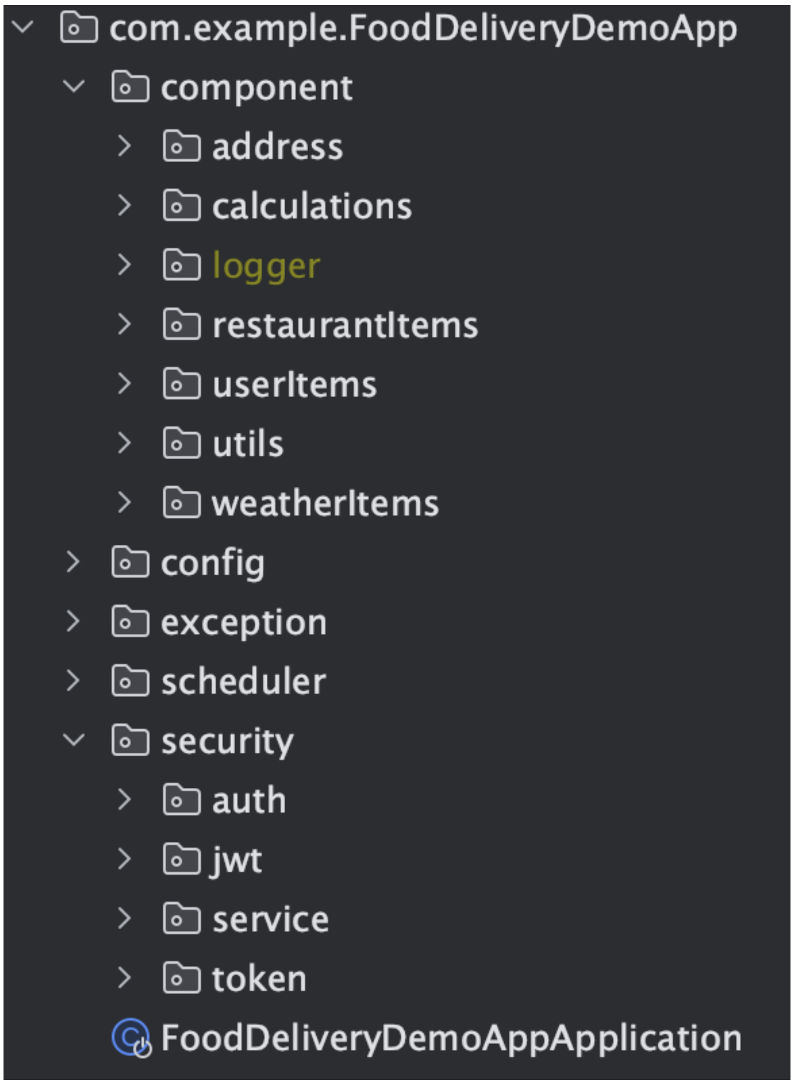
6. Back-end

In this chapter, the structure of the back-end part of the webpage is discussed. It includes the code repository structure, database structure and the components of back-end.

6.1. Code repository structure

There are two main approaches for structuring the Spring boot projects: by feature and by layer. Structure by feature means that all classes that are connected to a particular feature are placed in the same package. The other way is to place the classes based on its layer (Controller, Service) in the one package [19].

In the code of the website both approaches are used and modified for convenience. The main directories for code, such as config, exception and security directory are structured by layer and the directories and files inside them are structured by feature (Figure 38).



```
com.example.FoodDeliveryDemoApp
    component
        address
        calculations
        logger
        restaurantItems
        userItems
        utils
        weatherItems
    config
    exception
    scheduler
    security
        auth
        jwt
        service
        token
    FoodDeliveryDemoAppApplication
```

Figure 38. Structure of the code repository.

6.2. Database structure

The database schema is rather big, but it contains all the necessary tables to provide the functionality. The most important tables for the website are: users table, including admins, owners and customers tables; tokens table; restaurants, menus, items, orders tables (Appendix 2).

The users table and its dependent tables users_admins, users_owners and users_customers are needed to provide role-based access to the website. While sharing all the fields from the users table, role tables have the unique fields. The tokens table is necessary to store the tokens (that are JWT tokens) for the security measures of the website. They are used to validate the user at all times [54]. The restaurants, menus, items and orders tables are needed to provide the main functionality on the page and allow owners to provide the food for the customers by listing the data of the restaurants on the website and customers to order the food. The weather_data, delivery_fee and fee rules (base_fee_rules, extra_fee_rules_(wind_speed, weather_phenomenon and temperature)) tables are used to store the weather data and rules. With the help of the data from those tables the calculation of the delivery fee is performed. The tables are only used for calculation, and, therefore, are not connected to the main block of tables.

6.3. Components

In this code, the components are directories of classes that were structured by feature. The main components in the back-end encompass calculations, with the calculation of the delivery fee, fee rules and the directions; restaurant components - restaurants, menus, items and orders; user components - admins, owners, customers and users; weather components - weather data and external weather API.

The purposes for each component are listed below. The purpose of the restaurant and user components as well as the functionality was discussed in the front-end chapter (subchapters 5.1.2, 5.1.3 and 5.2).

Calculations:

- delivery fee - calculation of the base and weather fees and saving the sum into database;
- fee rules - base and weather (phenomenon, wind speed and temperature) fee rule management;
- google maps - get the route and time for the order deliveries via Google Directions API.

Weather:

- external API - takes the weather data from the Estonian Environment Agency Weather API [55];
- weather data - processes the data from the external API and saves the latest weather data every hour into the database.

7. Hosting

This chapter explains the way the created website is hosted.

There are different ways of hosting the infrastructure for the websites, applications and data. The traditional hosting includes the dedicated servers, shared server and VPS. Cloud hosting consists of many different methods. Cloud hosting is the place where the website or the app runs on the virtual servers in the cloud [56].

For the current project, all three parts necessary for the functioning website were hosted in the AWS. Front-end was hosted in the S3 bucket, back-end was hosted in Elastic Beanstalk environment and the database was hosted in RDS for PostgreSQL.

7.1. Front-end hosting

The easy way to host a front-end in the AWS is the static website in the S3 bucket. The process of setting up the bucket takes minutes and after the required files, such as "index.html" are uploaded to the bucket, the website is up and running [43]. The created website is hosted in the URL: http://fd-app-test.s3-website.eu-north-1.amazonaws.com/.

7.2. Back-end hosting

Due to the Elastic Beanstalk it is possible to deploy the back-end of the website via upload of the code. After uploading the code, the service will handle the deployment by itself [45]. The back-end API is shown in the Swagger UI in the URL: http://fd-app.eu-north-1.elasticbeanstalk.com/swagger-ui/index.html.

8. Testing

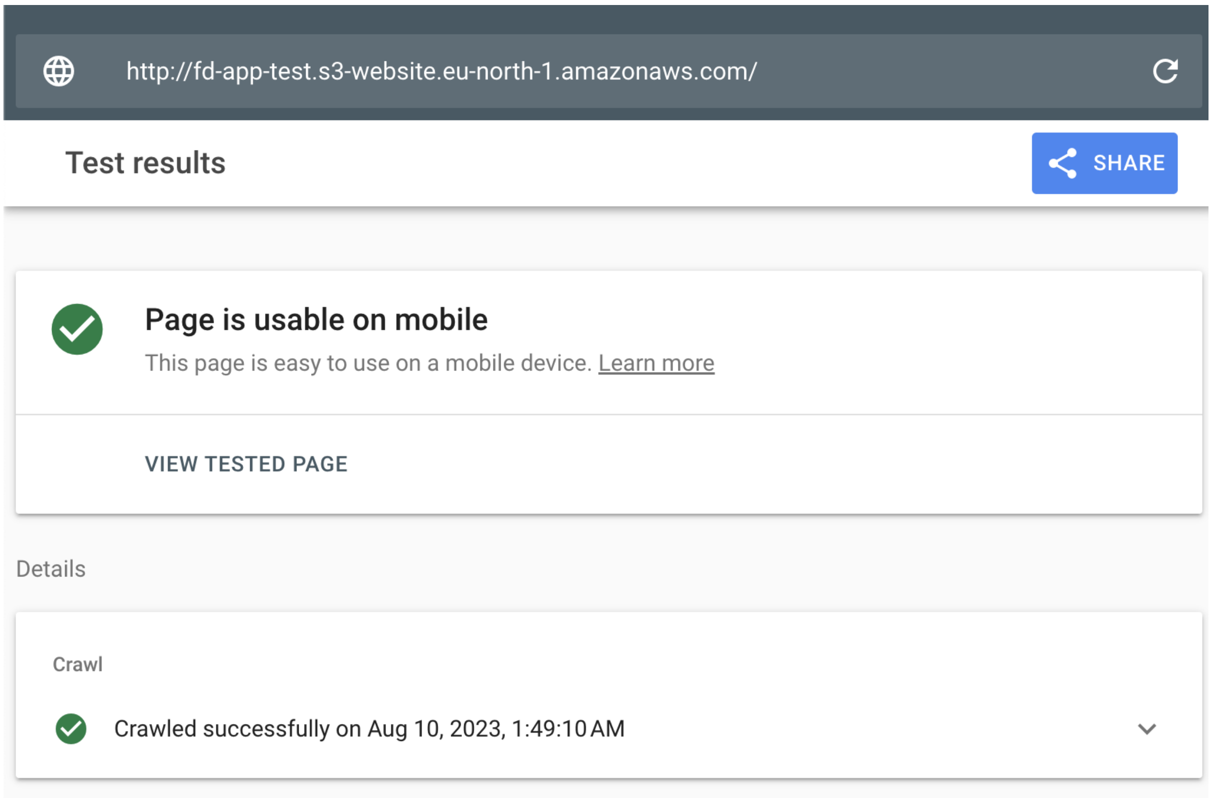
In this chapter, the results of the tests of the website are shown along with the feedback from the users.

It is important to test both parts, the front-end and the back-end, before actually letting the users use it. For this reason, several tests like website security, performance, mobile compatibility were applied. For the back-end, the tests should consist of unit and integration tests. Unit tests check the services, while integration tests check the controller endpoints.

8.1. Front-end testing

8.1.1. Mobile-friendly

To test mobile-friendliness, a testing tool called Mobile-Friendly Test [57], was used to evaluate how mobile-friendly the created website is. The test indicated that the website is mobile-friendly (Figure 39).



http://fd-app-test.s3-website.eu-north-1.amazonaws.com/

Test results

Page is usable on mobile

This page is easy to use on a mobile device. Learn more

Details

Crawl

Crawled successfully on Aug 10, 2023, 1:49:10 AM

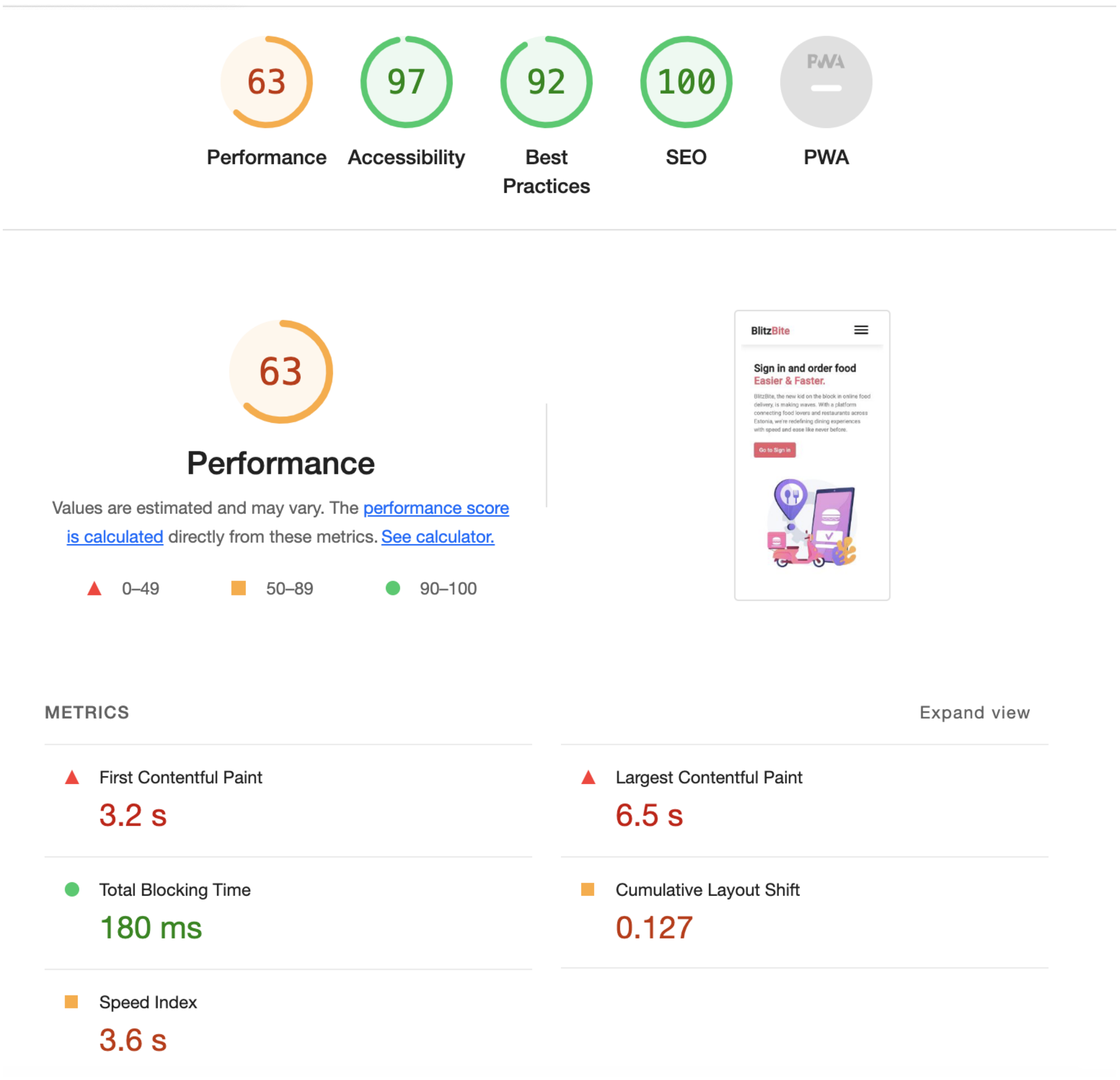
Figure 39. Mobile-friendliness test result.

8.1.2. Performance

To test the speed, performance and other aspects, the Lighthouse tool was used. This tool evaluates performance, accessibility, best practices and SEO [58]. The test was done both for mobile devices (Figure 40) and desktop devices (Figure 41).

The results indicated that the website is less performant in the mobile devices, than in desktop, while other metrics are good in both cases.

The score of performance in mobile is lower because generally mobile devices are less powerful than desktop devices and they require highly optimised websites, to have the maximum performance.



63 Performance

97 Accessibility

92 Best Practices

100 SEO

PWA

63

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

BlitzBite

Sign in and order food
Easier & Faster.

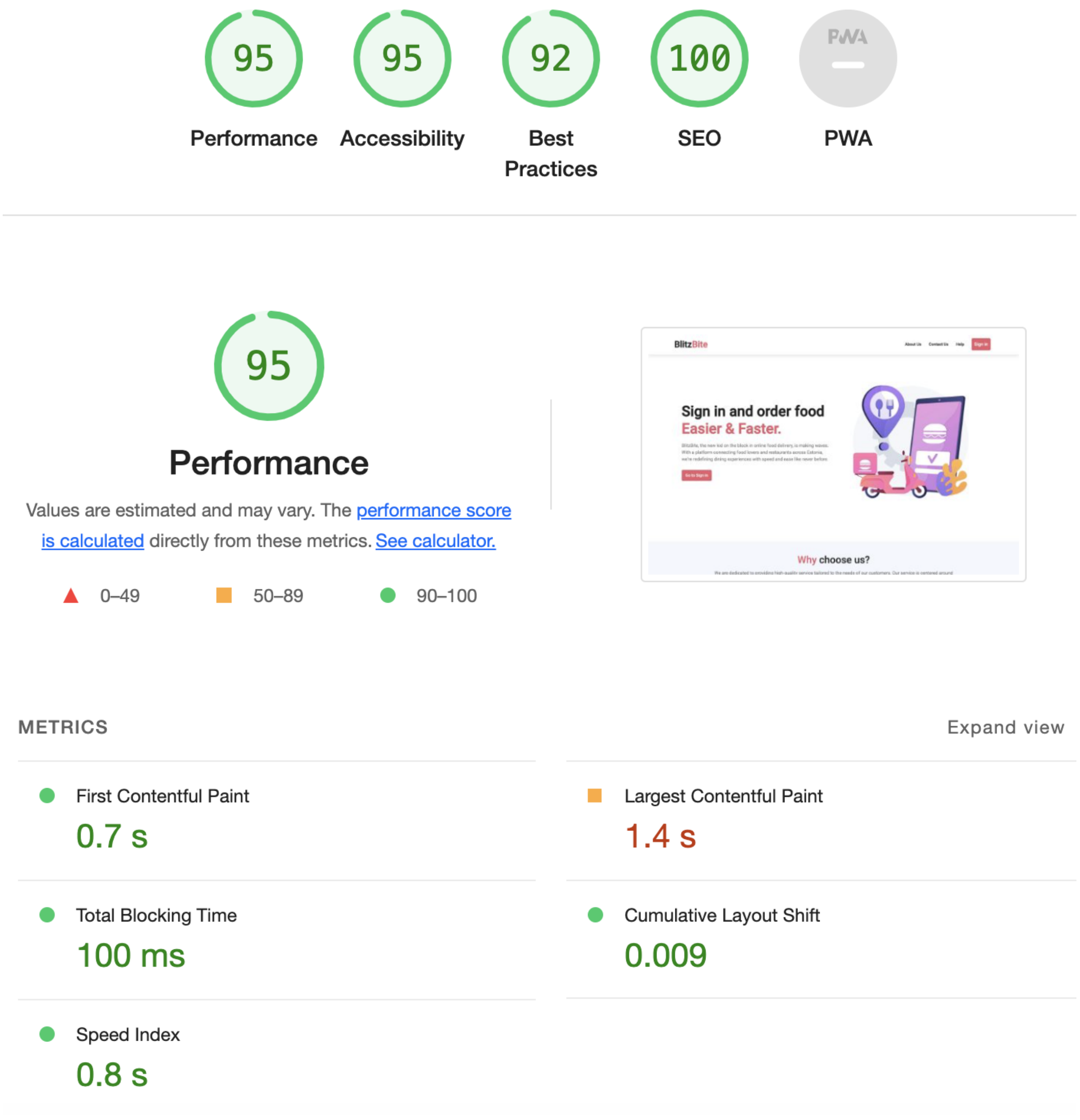
BlitzBite, the new kid on the block in online food delivery, is making waves. With a platform connecting food lovers and restaurants across Estonia, we're redefining dining experiences with speed and ease like never before.

Go to Sign in

METRICS Expand view

▲ First Contentful Paint 3.2 s	▲ Largest Contentful Paint 6.5 s
● Total Blocking Time 180 ms	■ Cumulative Layout Shift 0.127
■ Speed Index 3.6 s	

Figure 40. Lighthouse mobile device test result.


95 Performance

95 Accessibility

92 Best Practices

100 SEO

PWA

95

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

● First Contentful Paint
0.7 s

■ Largest Contentful Paint
1.4 s

● Total Blocking Time
100 ms

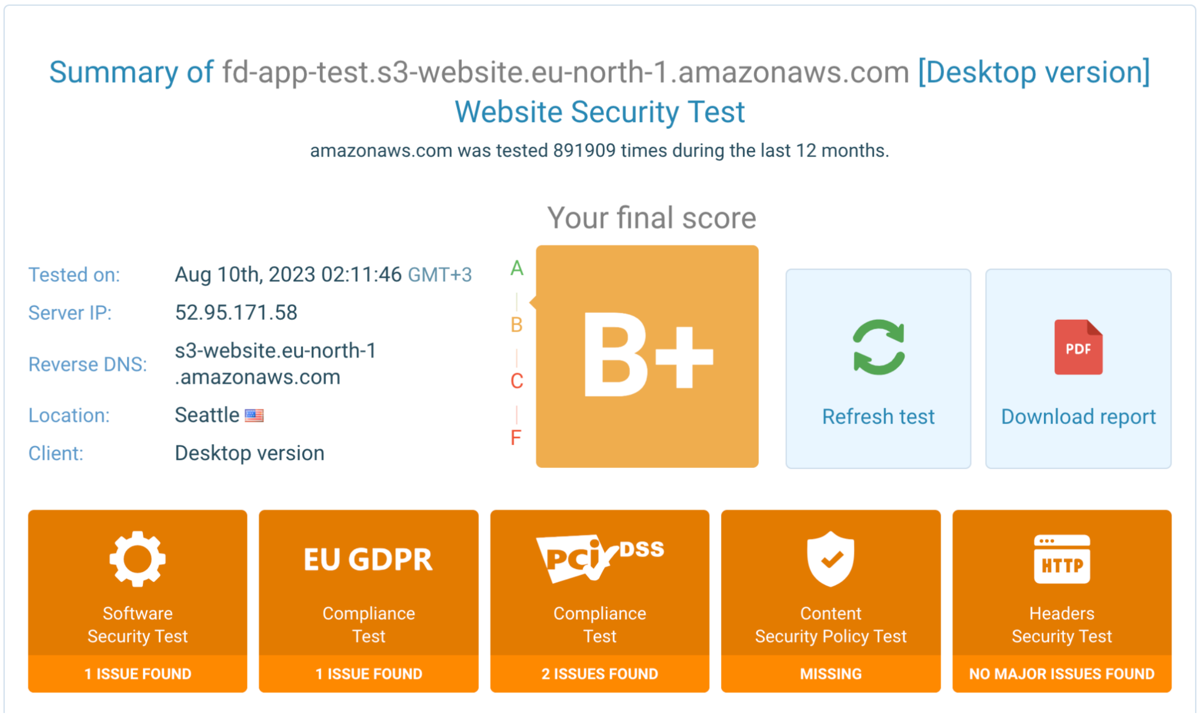
● Cumulative Layout Shift
0.009

● Speed Index
0.8 s

Figure 41. Lighthouse desktop device test result.

8.1.3. Security

To assess the security of the website, the website security test by the company ImmuniWeb was chosen [59]. The test is giving the result of B+ (Figure 42), which means the security is not perfect. The outcome is not unexpected, because the website is not hosted with the SSL certificate and is on HTTP protocol.


Summary of fd-app-test.s3-website.eu-north-1.amazonaws.com [Desktop version]
Website Security Test
amazonaws.com was tested 891909 times during the last 12 months.

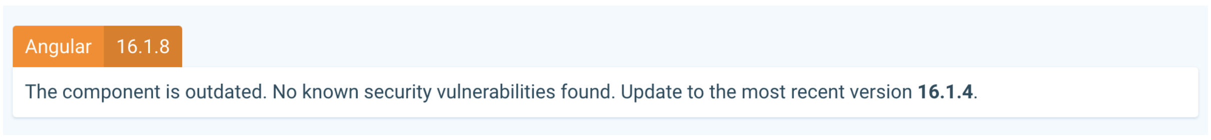
Tested on:	Aug 10th, 2023 02:11:46 GMT+3
Server IP:	52.95.171.58
Reverse DNS:	s3-website.eu-north-1 .amazonaws.com
Location:	Seattle
Client:	Desktop version

Your final score
B+

Software Security Test	EU GDPR Compliance Test	PCI DSS Compliance Test	Content Security Policy Test	Headers Security Test
1 ISSUE FOUND	1 ISSUE FOUND	2 ISSUES FOUND	MISSING	NO MAJOR ISSUES FOUND

Figure 42. ImmuniWeb security test result.

The ImmuniWeb shows that there is a vulnerability in software, with an outdated Angular version. When looking at the error, it is shown that the project version of Angular is newer than the reported newest version of Angular in the ImmuniWeb security test (Figure 43).


Angular 16.1.8

The component is outdated. No known security vulnerabilities found. Update to the most recent version **16.1.4**.

Figure 43. ImmuniWeb security test Angular vulnerability error.

8.2. Back-end testing

For testing of the back-end the following Java libraries were used: Mockito, Spring Boot Test, JUnit and AssertJ.

All the methods in the back-end were tested manually via Postman, Swagger UI and the actual website.

The application has the necessary configuration to either allow unit or integration tests to run with the build of the application. As well as not allowing them to run at all, to save time while building the app.

The application is built in the way so there is a dedicated database for the tests. It is rewritten in every run and uses the schema and data specified in the sql files.

8.2.1. Unit tests

Unit tests are the tests that are written to test the specific parts of the code, and are usually tested with the services [60]. The unit tests were written for DeliveryFeeService, checking all possible use cases of the calculation of the delivery fee and all tests were passed successfully (Figure 44).

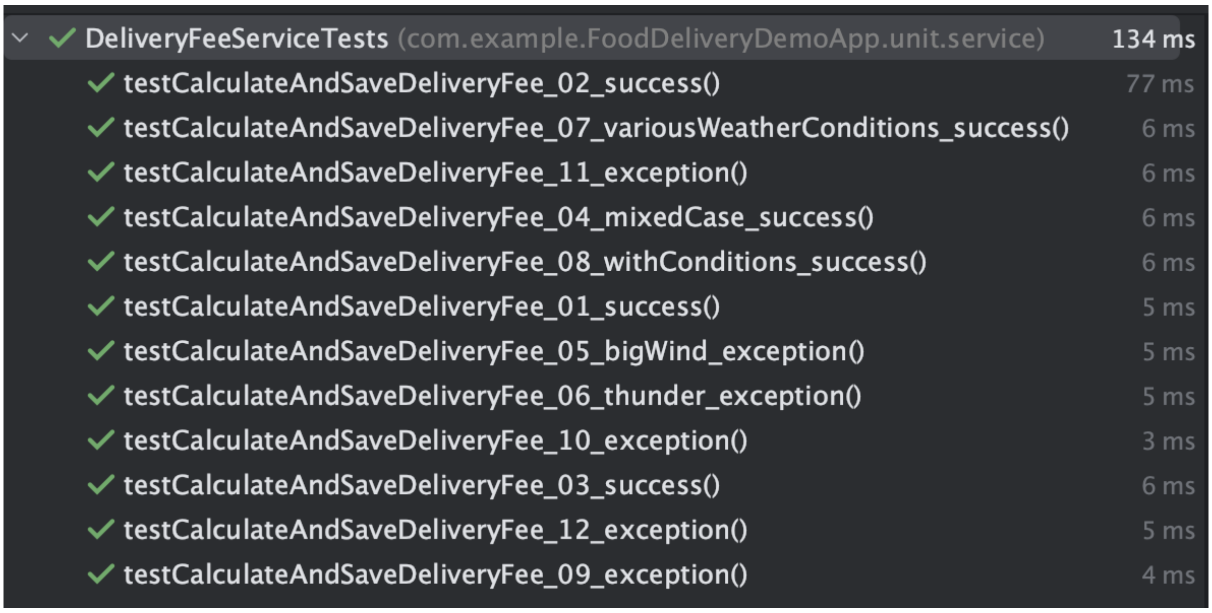

```
✓ DeliveryFeeServiceTests (com.example.FoodDeliveryDemoApp.unit.service)       134 ms
    ✓ testCalculateAndSaveDeliveryFee_02_success()                               77 ms
    ✓ testCalculateAndSaveDeliveryFee_07_variousWeatherConditions_success()       6 ms
    ✓ testCalculateAndSaveDeliveryFee_11_exception()                              6 ms
    ✓ testCalculateAndSaveDeliveryFee_04_mixedCase_success()                      6 ms
    ✓ testCalculateAndSaveDeliveryFee_08_withConditions_success()                 6 ms
    ✓ testCalculateAndSaveDeliveryFee_01_success()                                5 ms
    ✓ testCalculateAndSaveDeliveryFee_05_bigWind_exception()                      5 ms
    ✓ testCalculateAndSaveDeliveryFee_06_thunder_exception()                      5 ms
    ✓ testCalculateAndSaveDeliveryFee_10_exception()                              3 ms
    ✓ testCalculateAndSaveDeliveryFee_03_success()                                6 ms
    ✓ testCalculateAndSaveDeliveryFee_12_exception()                              5 ms
    ✓ testCalculateAndSaveDeliveryFee_09_exception()                              4 ms
```

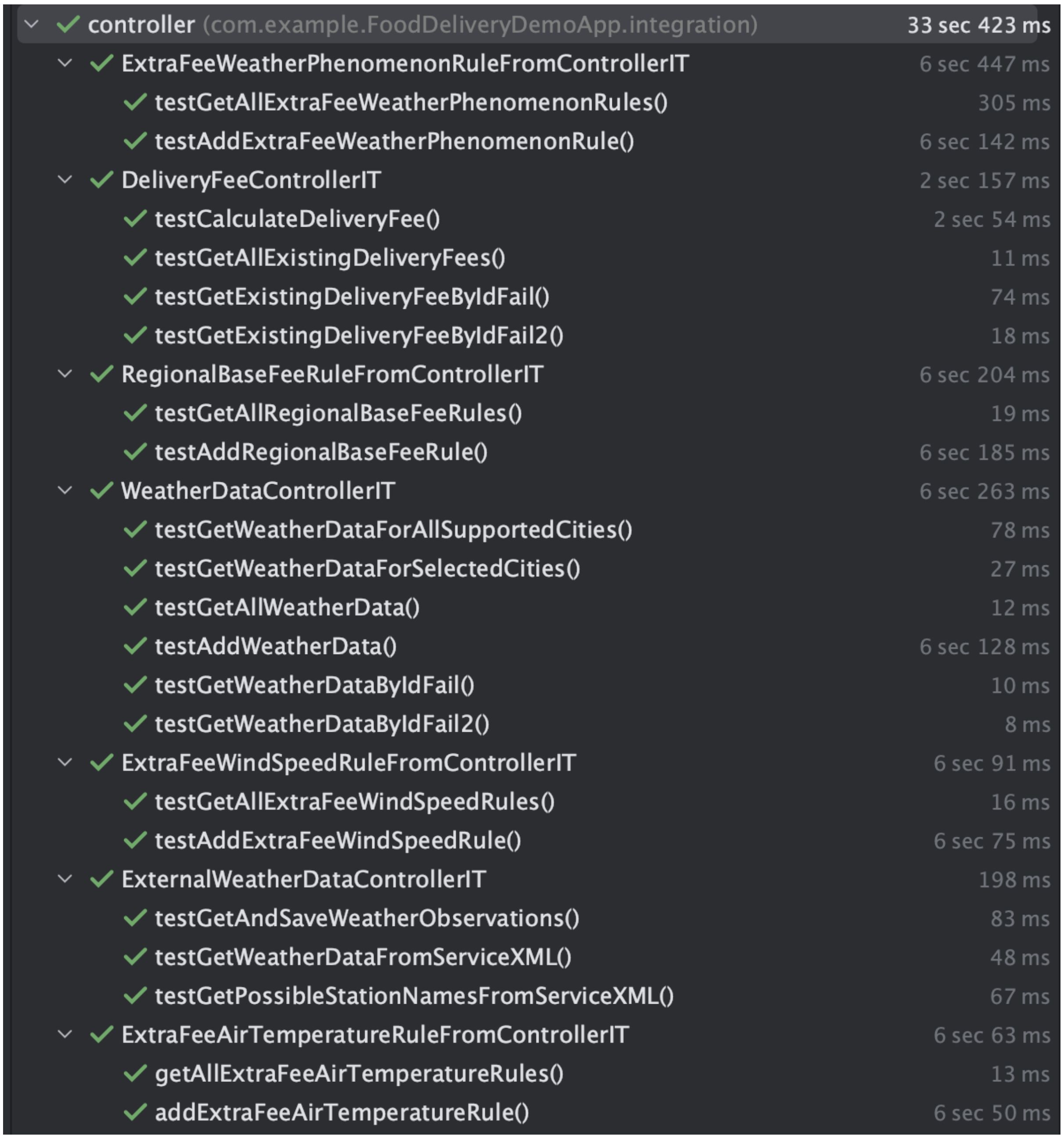
Figure 44. DeliveryFeeService unit tests results.

8.2.2. Integration tests

Integration tests are the tests that are written to test the combined modules of the application, to test how they work, and are usually tested with the endpoints of Controllers [61].

The integration tests were written for WeatherDataContoller, ExternalWeatherDataController, DeliveryFeeContoller and FeeRuleController and were successfully passed (Figure 45). The tests consisted of:

- get;
- add, get by id, edit by id, delete by id;
- get, add or edit with faulty ids or data.



```
✓ controller (com.example.FoodDeliveryDemoApp.integration)      33 sec 423 ms
  ✓ ExtraFeeWeatherPhenomenonRuleFromControllerIT                 6 sec 447 ms
    ✓ testGetAllExtraFeeWeatherPhenomenonRules()                        305 ms
    ✓ testAddExtraFeeWeatherPhenomenonRule()                      6 sec 142 ms
  ✓ DeliveryFeeControllerIT                                       2 sec 157 ms
    ✓ testCalculateDeliveryFee()                                   2 sec 54 ms
    ✓ testGetAllExistingDeliveryFees()                                   11 ms
    ✓ testGetExistingDeliveryFeeByIdFail()                               74 ms
    ✓ testGetExistingDeliveryFeeByIdFail2()                              18 ms
  ✓ RegionalBaseFeeRuleFromControllerIT                           6 sec 204 ms
    ✓ testGetAllRegionalBaseFeeRules()                                   19 ms
    ✓ testAddRegionalBaseFeeRule()                                6 sec 185 ms
  ✓ WeatherDataControllerIT                                       6 sec 263 ms
    ✓ testGetWeatherDataForAllSupportedCities()                          78 ms
    ✓ testGetWeatherDataForSelectedCities()                              27 ms
    ✓ testGetAllWeatherData()                                            12 ms
    ✓ testAddWeatherData()                                        6 sec 128 ms
    ✓ testGetWeatherDataByIdFail()                                       10 ms
    ✓ testGetWeatherDataByIdFail2()                                       8 ms
  ✓ ExtraFeeWindSpeedRuleFromControllerIT                          6 sec 91 ms
    ✓ testGetAllExtraFeeWindSpeedRules()                                 16 ms
    ✓ testAddExtraFeeWindSpeedRule()                               6 sec 75 ms
  ✓ ExternalWeatherDataControllerIT                                     198 ms
    ✓ testGetAndSaveWeatherObservations()                                83 ms
    ✓ testGetWeatherDataFromServiceXML()                                 48 ms
    ✓ testGetPossibleStationNamesFromServiceXML()                        67 ms
  ✓ ExtraFeeAirTemperatureRuleFromControllerIT                     6 sec 63 ms
    ✓ getAllExtraFeeAirTemperatureRules()                                13 ms
    ✓ addExtraFeeAirTemperatureRule()                              6 sec 50 ms
```

Figure 45. Integration tests results.

8.3. User feedback

A feedback survey was made to get the opinion on the created website. Feedback survey (Appendix 3) was created with the Google Forms [62]. The feedback survey was divided into two parts, where the first part was the mandatory questions and the second part was the open question for the feedback. The majority of the questions were with the 1 to 5 scale system, where “1” stands for awful and “5” stands for excellent.

The survey was sent to several chats, including the chat with the students and the 10 people answered. Most of the respondents used mobile phones to access the site (Figure 46).

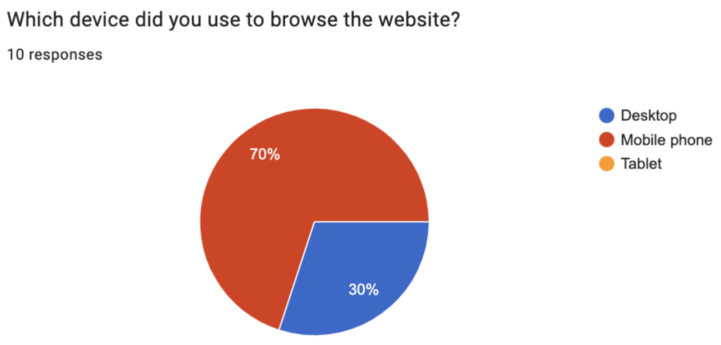


Figure 46. Comparison of devices used by website users.

The majority of the users were satisfied with the flow and navigation of the website and for nobody the navigation was awful (Figure 47).

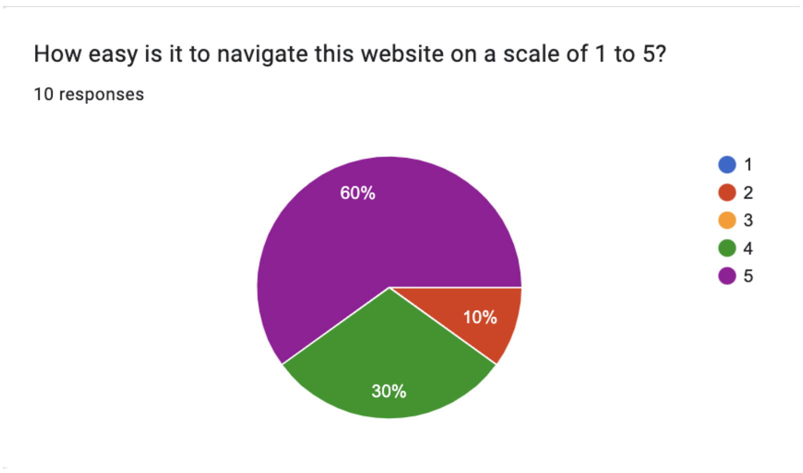


Figure 47. Website ease of navigation feedback.

Most of the respondents were satisfied with the design and visual appeal of the website (Figure 48).

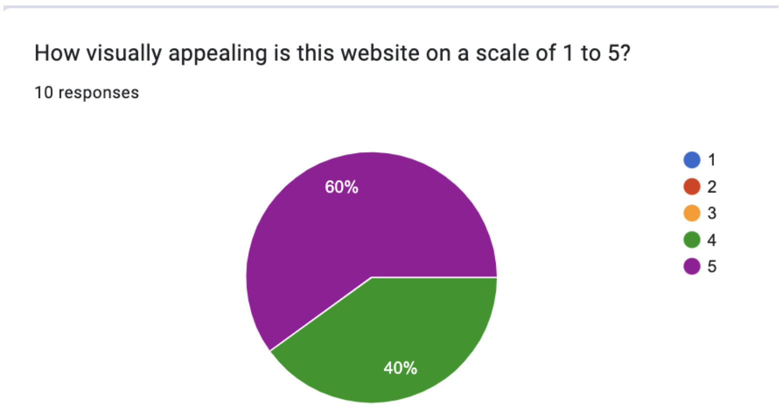


Figure 48. Website visual appeal feedback.

Also, the majority of the respondents did not have any problem with the loading time of the page and the response times on the page, meaning the website operated fast (Figure 49).

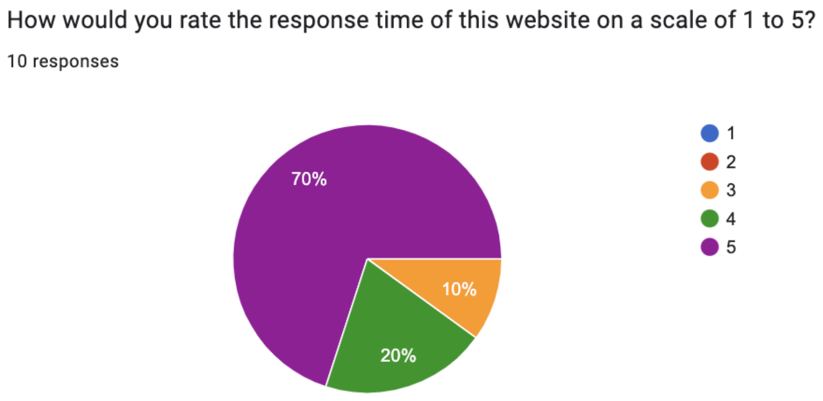


Figure 49. Website response time feedback.

According to the results, 90% of the people who responded to the survey had good and very good experience with the website (Figure 50).

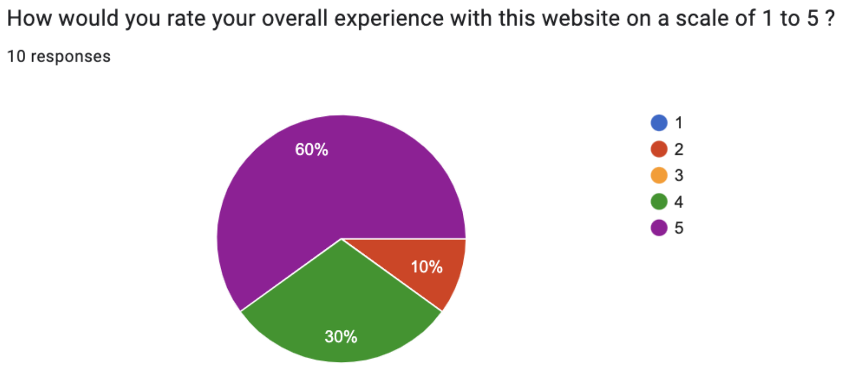


Figure 50. Website overall experience feedback.

People were also asked about possible improvements. There was a list with different options and it was possible to select multiple answers. The most picked answer was "Everything is fine", but respondents also picked at least one of each provided aspect that can be improved (Figure 51). Options like "Design" and "Functionality" are the most chosen ways how the website can be improved.

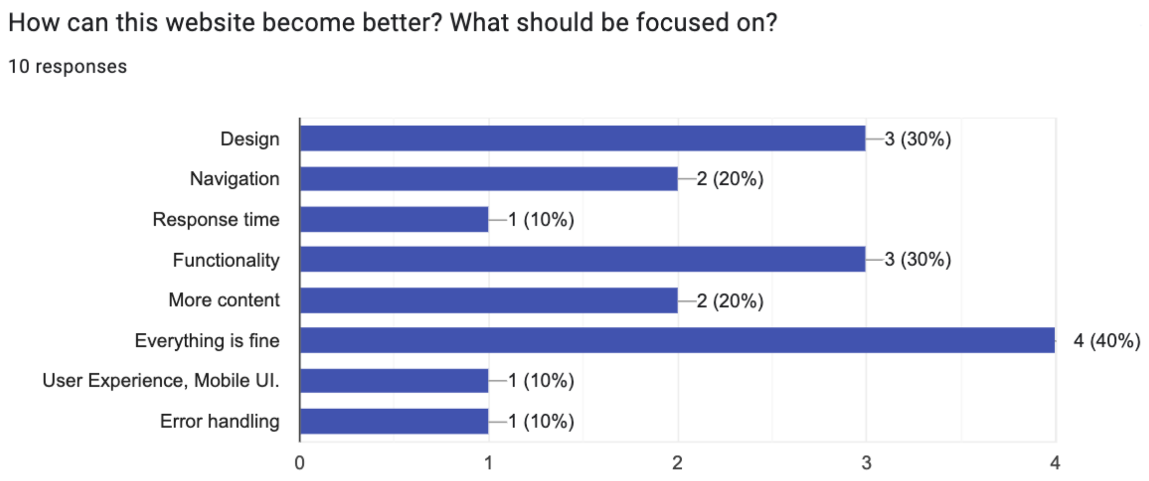


Figure 51. Suggestions for website improvement feedback.

At the end of the questionnaire, respondents could leave their comments. There were only 2 responses, one being the smiling emoji and the other response stated that: “the website is very nice, but the option for the password confirmation is missing, so it is easy to make a mistake and lose an account”.

So, based on the survey results, it could be concluded that overall most users were satisfied with the website. In the last chapter of the thesis, some possible future works are given.

9. Future works

In this chapter, the various ways that can be used to improve the website functionality, user experience and security are provided.

Currently, the owners can’t add images for the restaurants and items and it makes the website seem empty. It can be done by adding the feature to upload files on the AWS S3 and storing the URL of the image in the database.

As on every food delivery website and application, the rating and feedback system can be added.

Also, the implementation of the directions can be vastly improved, as right now the Google Directions API responses seem inconsistent and sometimes are not showing the proper output on the map.

The possibility to order food from different restaurants can also be added, with the logic that automatically makes the order into two orders and with the double delivery fee. It should be implemented this way, because the one delivery driver should not be able to deliver simultaneously from 2 different restaurants that can be in different parts of town.

For the hosting part, the website should be hosted using the SSL certificate and the protocol to be HTTPS instead of HTTP for the full security for the users.

Conclusion

The goal of the bachelor’s thesis was to develop a food delivery website that allowed the owners of the restaurants and the customers to use the site and its functionality, such as managing the restaurant and ordering of the food. The distinguishing feature of the created website was the taking into account weather conditions in the calculation of delivery fee.

During the process of writing the bachelor's thesis, a website prototype was developed, and a comprehensive development plan was composed. This plan consisted of the front-end and back-end components, as well as the functional and non-functional requirements.

To create a website the one of the most popular and seeked technologies on the job market were used, such as Java, Spring Boot, Angular, PostgreSQL and AWS. As a result of the work, the website was completed and hosted in the AWS on the domain http://fd-app-test.s3-website.eu-north-1.amazonaws.com/. Website was tested on both front- and back-end. The tests on the front-end consisted of mobile-friendliness, performance, accessibility and security. The front-end implementation showed good results on every test. The back-end was tested with the unit tests and the integration tests that all passed successfully. The feedback survey for the website showed that users thought that the website has good design, is performant and were overall satisfied with the experience on the website. However, the survey respondents highlighted several areas for improvement, which have been included into the list of potential future works.

References

[1] Bolt Food. https://bolt.eu/et-ee/food (09.08.2023)

[2] Wolt. https://wolt.com/en/est (09.08.2023)

[3] Fudy. https://fudy.ee (09.08.2023)

[4] Tellitoit. https://www.tellitoit.ee (09.08.2023)

[5] Shaurma Kebab. https://shaurmakebab.ee (09.08.2023)

[6] Lit. “Kebab Plate”. https://litwagon.ee/lit-pirita/kebab (09.08.2023)

[7] Wolt. “Lit Kebab Plate”. https://wolt.com/en/est/tallinn/restaurant/lit-pirita#kebab-3 (09.08.2023)

[8] Bolt Food. “Restaurant & Takeaway Sign Up”. https://partners.food.bolt.eu (09.08.2023)

[9] Vested Finance. Food Delivery Business - How to be Profitable in the Food Delivery. https://vestedfinance.com/in/blog/how-to-be-profitable-in-the-food-delivery-business (09.08.2023)

[10] MacroTrends. “DoorDash Net Income 2019 - 2023”. https://www.macrotrends.net/stocks/charts/DASH/doordash/net-income (09.08.2023)

[11] MacroTrends. “Uber Technologies Net Income 2019 - 2023”. https://www.macrotrends.net/stocks/charts/UBER/uber-technologies/net-income (09.08.2023)

[12] A.Whyte, B. Oja. ERR. “Bolt revenue up 152 percent on year to 2022, still made a loss”. https://news.err.ee/1609031954/bolt-revenue-up-152-percent-on-year-to-2022-still-made-a-loss (09.08.2023)

[13] K. Ahuja, V. Chandra, V. Lord, and C. Peens. McKinsey & Company. “Ordering in: The rapid evolution of food delivery”. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ordering-in-the-rapid-evolution-of-food-delivery (09.08.2023)

[14] Mozilla. MDN Web Docs. “What do common web layouts contain?”. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Design_and_accessibility/Common_web_layouts (25.04.2023)

[15] Digiworks. “13 Parts of a Website You Should Know About”. https://www.digiworks.co.za/13-parts-of-a-website-you-should-know-about (25.04.2023)

[16] Branding Compass. “Color Theory: Red as a Branding Color”. https://brandingcompass.com/branding/color-theory-red-as-a-branding-color (30.04.2023)

[17] Codevoweb. “LC24- Responsive Food Ordering Website with HTML, CSS, and JavaScript”. https://codevoweb.com/lc24-responsive-food-delivery-website-with-html-css-and-javascript (10.05.2023)

[18] Baeldung. “Learn Spring Boot”. https://www.baeldung.com/spring-boot (16.03.2023)

[19] Ganeshchowdharysadanala. GeeksForGeeks. “Spring Boot – Code Structure”. https://www.geeksforgeeks.org/spring-boot-code-structure (16.03.2023)

[20] Angular. “Workspace and project file structure”. https://angular.io/guide/file-structure (25.04.2023)

[21] S. Nath. Medium. “Angular File Structure and Best Practices (that help to scale)”. https://medium.com/@shijin_nath/angular-right-file-structure-and-best-practices-that-help-to-scale-2020-52ce8d967df5 (25.04.2023)

[22] AltexSoft. “Functional and Nonfunctional Requirements: Specification and Types”. https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types (30.04.2023)

[23] Y. Shvetsova. Elogic Commerce. “Functional and Non-Functional Requirements of Online Shopping System: List & Examples”. https://elogic.co/blog/functional-and-non-functional-requirements-for-ecommerce-websites (08.08.2023)

[24] Indeed Editorial Team. Indeed. “What is a tech stack? (Plus importance, types and examples)”. https://uk.indeed.com/career-advice/career-development/what-is-tech-stack (07.08.2023)

[25] IBM. “What is Java?”. https://www.ibm.com/topics/java (01.08.2023)

[26] Javatpoint. “Features of Java”. https://www.javatpoint.com/features-of-java (01.08.2023)

[27] University of Tartu Institute of Computer Science. “Object-oriented Programming”. https://courses.cs.ut.ee/2023/OOP/spring (01.08.2023)

[28] StackShare. “Java”. https://stackshare.io/java (01.08.2023)

[29] Stack Overflow. “Stack Overflow Developer Survey 2023”. https://survey.stackoverflow.co/2023 (15.07.2023)

[30] LinkedIn. “See all Jobs”. https://www.linkedin.com/jobs/search/ (10.08.2023)

[31] Baeldung. “A Comparison Between Spring and Spring Boot”. https://www.baeldung.com/spring-vs-spring-boot (16.03.2023)

[32] Spring. “Spring Boot”. https://spring.io/projects/spring-boot (16.03.2023)

[33] IBM. “What is Java Spring Boot?”. https://www.ibm.com/topics/java-spring-boot (01.08.2023)

[34] Spring. “Spring Security”. https://spring.io/projects/spring-security (19.05.2023)

[35] Angular. “Introduction to the Angular docs”. https://angular.io/docs (25.04.2023)

[36] Amazon Web Services. “What is PostgreSQL?”. https://aws.amazon.com/rds/postgresql/what-is-postgresql (29.04.2023)

[37] H. Kamau. StackAbuse. “Working with PostgreSQL in Java”. https://stackabuse.com/working-with-postgresql-in-java (29.04.2023)

[38] Amazon Web Services. “What is Docker?“. https://aws.amazon.com/docker (10.05.2023)

[39] jwt.io. “Introduction to JSON Web Tokens”. https://jwt.io/introduction (19.05.2023)

[40] jwt.io. https://jwt.io (19.05.2023)

[41] Amazon Web Services. “What is AWS”. https://aws.amazon.com/what-is-aws (28.06.2023)

[42] Amazon Web Services. “Cloud Object Storage - Amazon S3”. https://aws.amazon.com/s3 (28.06.2023)

[43] Amazon Web Services. "Hosting a static website using Amazon S3". https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteHosting.html (28.06.2023)

[44] Amazon Web Services. "Website & Web App Deployment - AWS Elastic Beanstalk". https://aws.amazon.com/elasticbeanstalk (28.06.2023)

[45] Amazon Web Services. "The Amazon EC2 instances for your Elastic Beanstalk environment". https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.managing.ec2.html (28.06.2023)

[46] Amazon Web Services. "Launch a Linux virtual machine in Amazon Web Services". https://aws.amazon.com/getting-started/launch-a-virtual-machine-B-0 (28.06.2023)

[47] Amazon Web Services. "Fully Managed Relational Database - Amazon RDS". https://aws.amazon.com/rds (28.06.2023)

[48] Angular. "Understanding Angular". https://angular.io/guide/understanding-angular-overview (25.04.2023)

[49] Angular. "Angular components overview". https://angular.io/guide/component-overview (25.04.2023)

[50] Angular. "Understanding dependency injection". https://angular.io/guide/dependency-injection (25.04.2023)

[51] Angular. "HTTP Server communication". https://angular.io/guide/http-server-communication (25.04.2023)

[52] Angular. "Common Routing Tasks". https://angular.io/guide/router (30.04.2023)

[53] Angular. "HttpInterceptor". https://angular.io/api/common/http/HttpInterceptor (30.04.2023)

[54] K. Karnatakapu. Medium. "How to secure Spring Boot with JWT Authentication and Authorization". https://medium.com/javarevisited/spring-security-role-based-access-implementation-with-spring-boot-3-0-and-jwt-34164bd593fd (19.05.2023)

[55] Estonian Environment Agency. "Observation data". https://www.ilmateenistus.ee/ilm/ilmavaatlused/vaatlusandmed/?lang=en (16.03.2023)

[56] Google Cloud. "What is cloud hosting?". https://cloud.google.com/learn/what-is-cloud-hosting (29.06.2023)

[57] Google Search Console. "Mobile-Friendly Test". https://search.google.com/test/mobile-friendly (03.08.2023)

[58] Chrome Developers. "Lighthouse Overview". https://developer.chrome.com/docs/lighthouse/overview (03.08.2023)

[59] ImmuniWeb. "Website Security Test". https://www.immuniweb.com/websec (03.08.2023)

[60] T. Hamilton. Guru99. "Unit Test vs Integration Test – Difference Between Them". https://www.guru99.com/unit-test-vs-integration-test.html (26.03.2023)

[61] Baeldung. "Integration Testing in Spring". https://www.baeldung.com/integration-testing-in-spring (16.03.2023)

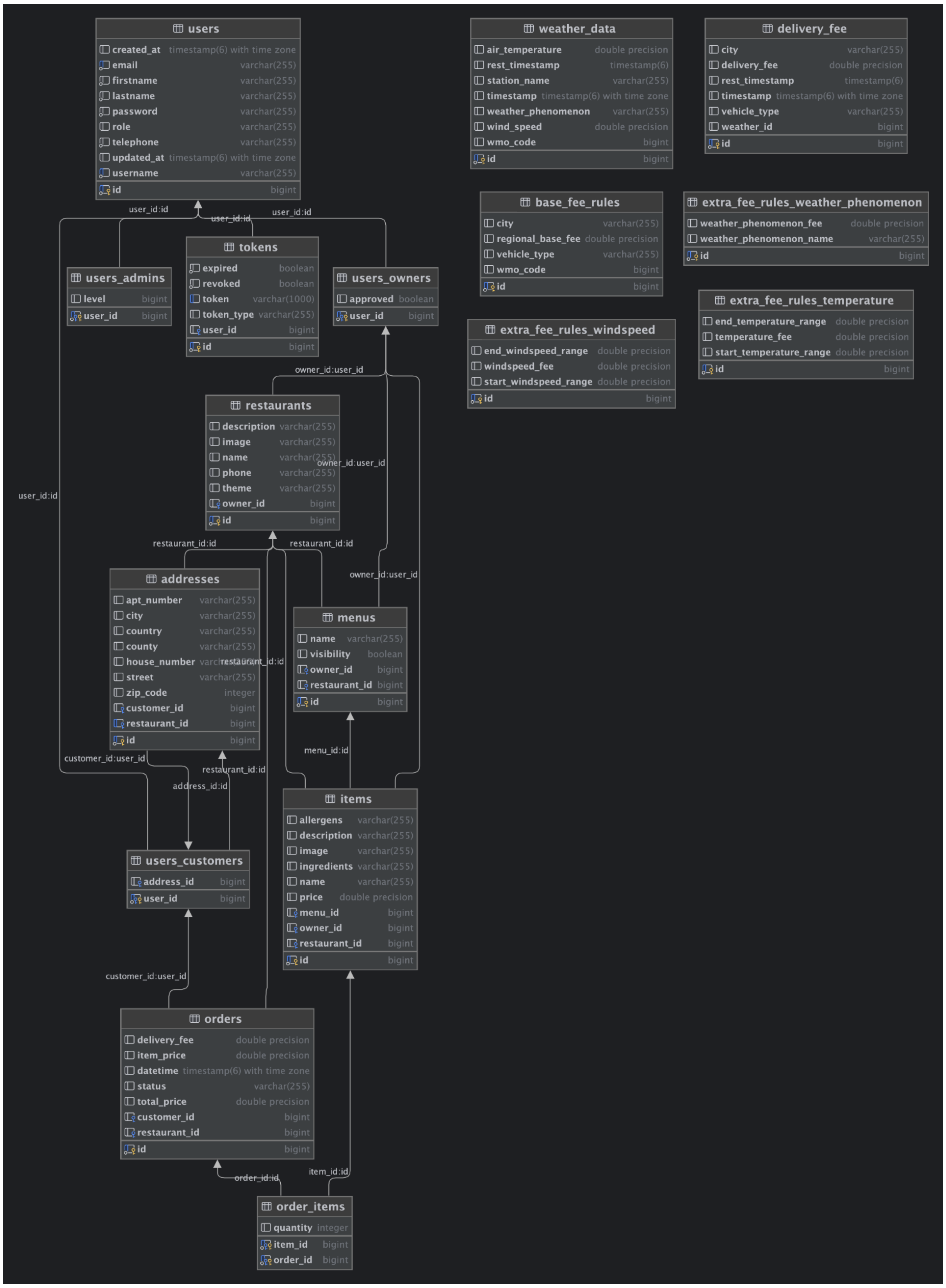
[62] Google Workspace. “Google forms: Online Form Creator”. https://www.google.com/forms/about (06.08.2023)

Appendix

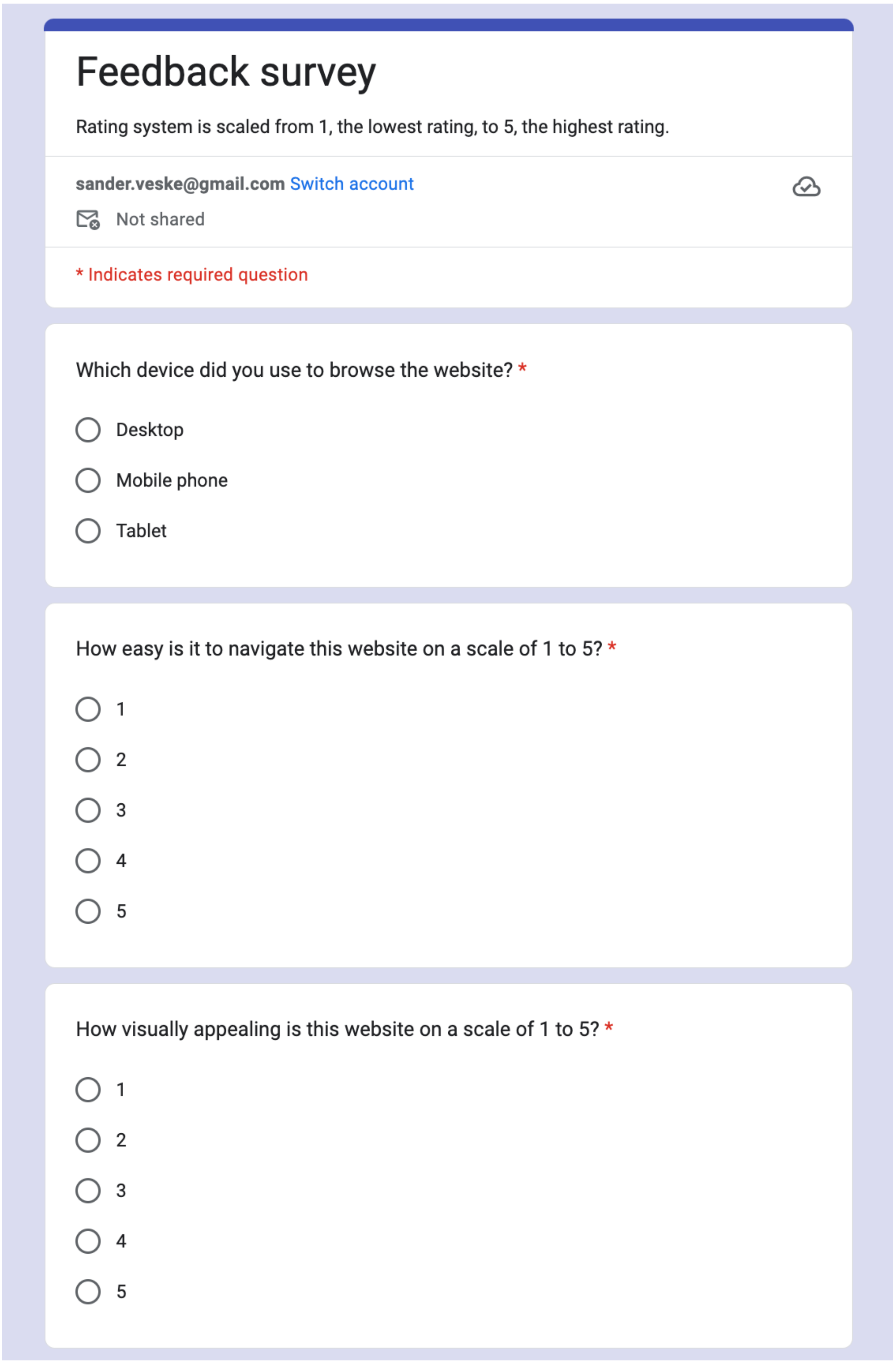
I. Access to Code

Front-end GitHub:	https://github.com/sndvsk/fd-frontend
Back-end GitHub:	https://github.com/sndvsk/foodDeliveryDemoApp
Website link:	fd-app-test.s3-website.eu-north-1.amazonaws.com
Swagger UI:	http://fd-app.eu-north-1.elasticbeanstalk.com/swagger-ui/index.html

II. Database structure



III. Feedback survey



Feedback survey

Rating system is scaled from 1, the lowest rating, to 5, the highest rating.

sander.veske@gmail.com Switch account

Not shared

* Indicates required question

Which device did you use to browse the website? *

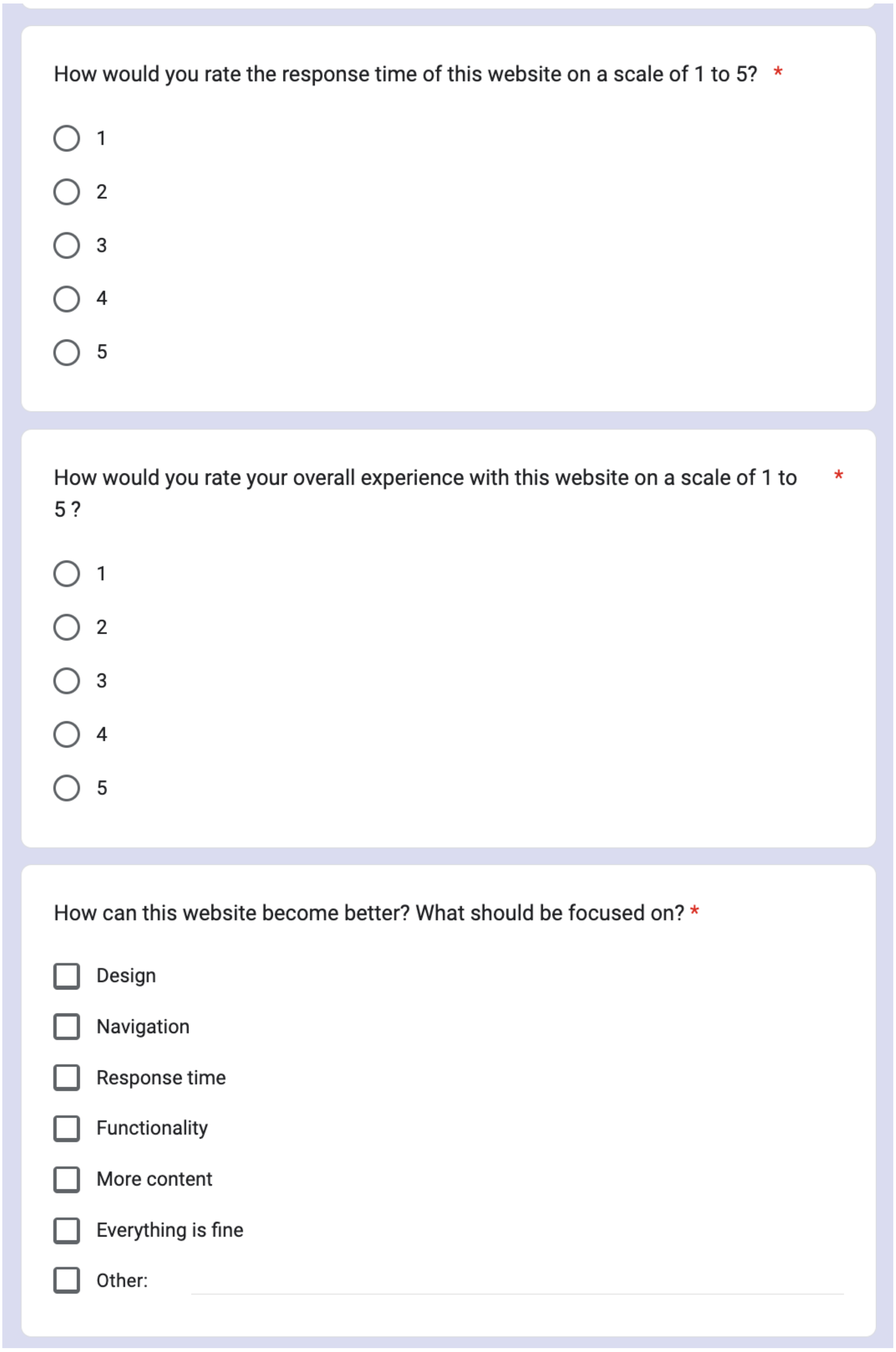
- ◯ Desktop
- ◯ Mobile phone
- ◯ Tablet

How easy is it to navigate this website on a scale of 1 to 5? *

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

How visually appealing is this website on a scale of 1 to 5? *

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5



How would you rate the response time of this website on a scale of 1 to 5? *

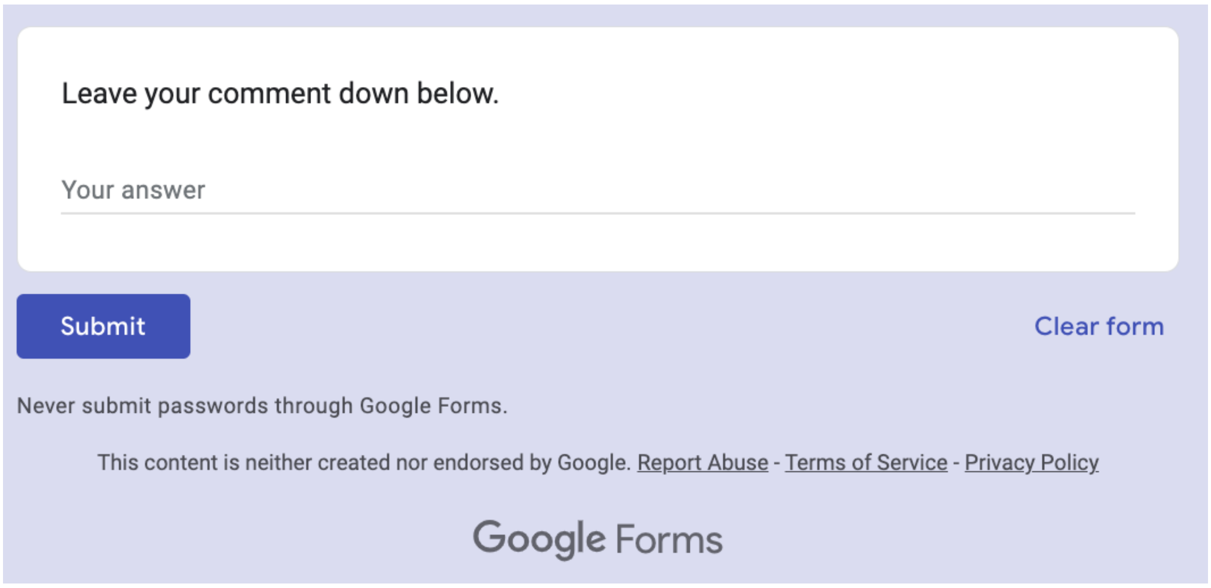
- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

How would you rate your overall experience with this website on a scale of 1 to 5 ? *

- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

How can this website become better? What should be focused on? *

- ☐ Design
- ☐ Navigation
- ☐ Response time
- ☐ Functionality
- ☐ More content
- ☐ Everything is fine
- ☐ Other:



Leave your comment down below.

Your answer

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. Report Abuse - Terms of Service - Privacy Policy

Google Forms

IV. License

Non-exclusive licence to reproduce thesis and make thesis public