HTBLVA, Wien V, Spengergasse

Höhere Lehranstalt für Medizininformatik

Ausbildungsschwerpunkt Medizininformatik

DIPLOMARBEIT

Gesamtprojekt

Erstellung einer Webapplikation für den biologischen Anbau für eine kenianische Frauenselbsthilfegruppe

Wie kann die Datenpersistenz in Webapplikationen mit Authentifizierung geplant und umgesetzt werden?
Everlyn Gathoni Njeri 5AHBGM Betreuer: Marcel Schwarz

Bereitstellung und Implementierung von Sicherheit, Planung und Entwicklung der User-Experience der Webapplikation.
Sarah Jessica Paukner 5AHBGM Betreuer: Marcel Schwarz

Wie kann eine Webapplikation implementiert werden, die HTTP-Endpunktebereitstellt und Ereignisse empfängt, sowie die Entwicklung dieses Systems?
Valerian Braun 5AHBGM Betreuer: Marcel Schwarz

Planung und Umsetzung/Entwicklung vom Deployment, Tests und Qualitätskontrolle der Webapplikation.
Elias Andrei 5AHBGM Betreuer: Marcel Schwarz

Abgabevermerk:

Schuljahr 2023/24

Datum: 04.04.2024 übernommen von: Marcel Schwarz

SPENGERGASSE	HÖHERE TECHNISCHE BUNDESLEHRANSTALT SPENGERGASSE	
	Fachrichtung:	**Medizininformatik**
	Ausbildungsschwerpunkt:	**Medizininformatik**

EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen benützt habe. Die Stellen, die anderen Werken (gilt ebenso für Werke aus elektronischen Datenbanken oder aus dem Internet) wörtlich oder sinngemäß entnommen sind, habe ich unter Angabe der Quelle und Einhaltung der Regeln wissenschaftlichen Zitierens kenntlich gemacht. Diese Versicherung umfasst auch in der Arbeit verwendete bildliche Darstellungen, Tabellen, Skizzen und Zeichnungen.

Für die Verbesserung der Diplomarbeit wurde Deepl Write genutzt, um den akademischen Stil des Textes anzupassen. Zusätzlich wurde ChatGPT ausschließlich für die Fehlerkorrektur im Code verwendet. Sofern für die Erstellung der Arbeit generative KI-Tools eingesetzt wurden, so sind auch diese im Quellenverzeichnis vollständig und wahrheitsgetreu inkl. Produktversion und Prompt ausgewiesen.

Wien, am 04.04.2024

Verfasser / Verfasserinnen:

Everlyn Gathoni Njeri

Sarah Jessica Paukner

Valerian Braun

Elias Andrei

SPENGERGASSE	**HÖHERE TECHNISCHE BUNDESLEHRANSTALT SPENGERGASSE**
	Fachrichtung: **Medizininformatik** Ausbildungsschwerpunkt: **Medizininformatik**

DIPLOMARBEIT

DOKUMENTATION

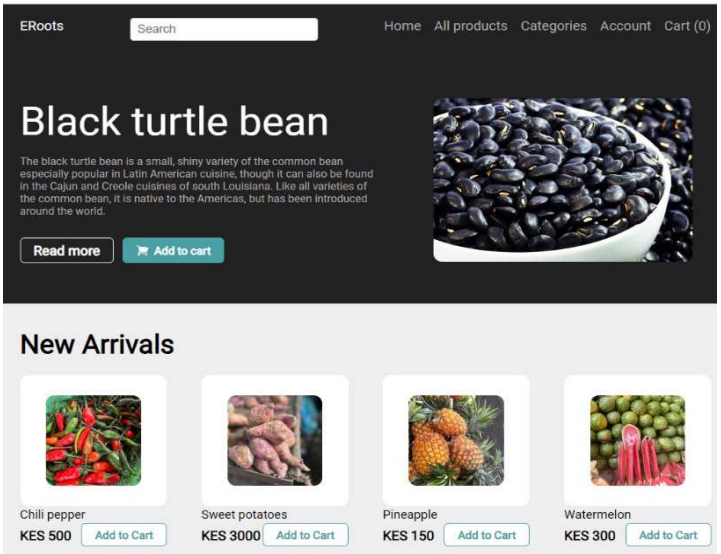
Namen der Verfasser/innen	Everlyn Gathoni Njeri Sarah Jessica Paukner Valerian Braun Elias Andrei
Jahrgang Schuljahr	2023/24
Thema der Diplomarbeit	Erstellung einer Webapplikation für den biologischen Anbau für eine kenianische Frauenselbsthilfegruppe
Kooperationspartner	Mutundu Women Self Help Group

Aufgabenstellung	**Everlyn Gathoni Njeri:** Wie kann die Datenpersistenz in Webapplikationen mit Authentifizierung geplant und umgesetzt werden? **Sarah Jessica Paukner:** Bereitstellung und Implementierung von Sicherheit, Planung und Entwicklung der User-Experience der Webapplikation. **Valerian Braun:** Wie kann eine Webapplikation implementiert werden, die HTTP-Endpunktebereitstellt und Ereignisse empfängt, sowie die Entwicklung dieses Systems? **Elias Andrei:** Planung und Umsetzung/Entwicklung vom Deployment, Tests und Qualitätskontrolle der Webapplikation.

Realisierung	**Next.js:** Ein React-Framework für das Erstellen von serverseitig gerenderten Webanwendungen. **MongoDB mit Mongoose:** Eine NoSQL-Datenbank für die Speicherung und Verwaltung von Daten. **Google OAuth:** Ein Authentifizierungsmechanismus von Google, der es Benutzern ermöglicht, sich mit ihren Google-Konten bei deiner Anwendung anzumelden. **Tailwind CSS:** Ein CSS-Framework, das auf Utility-Klassen basiert und es einfach macht, benutzerdefinierte Designs zu erstellen, ohne CSS zu schreiben. **Vercel:** Eine Plattform für das Bereitstellen und Verwalten von serverseitig gerenderten React-Anwendungen, die nahtlos mit Next.js integriert ist. **AWS S3:** Ein Objektspeicherdienst von Amazon Web Services, der es ermöglicht, große Datenmengen sicher zu speichern und von überall darauf zuzugreifen. **IONOS:** Ein Webhosting-Unternehmen, das verschiedene Dienste wie Domainregistrierung, Webhosting und Serveranwendungen anbietet.

SPENGERGASSE	**HÖHERE TECHNISCHE BUNDESLEHRANSTALT SPENGERGASSE**
	Fachrichtung: **Medizininformatik** Ausbildungsschwerpunkt: **Medizininformatik**

Ergebnisse	Funktionalität in folgenden Bereichen: • Administrationsinterface • Kundeninterface • Produktverwaltung • Verwaltung der Reservierungen • Datenbankverwaltung

Typische Grafik, Foto etc. (mit Erläuterung)	Logo EVERGREEN ROOTS *Figure 1-1 Logo* Die Webapplikation unter der Domain „eroots.store“.  *Figure 1-2: Client Webapplication*

Teilnahme an Wettbewerben, Auszeichnungen		

Möglichkeiten der Einsichtnahme in die Arbeit	In der Spengergassen Schulbibliothek.

Approbation (Datum / Unterschrift)	Prüfer/Prüferin Marcel Schwarz, Florian Hell	Direktor/Direktorin Abteilungsvorstand/Abteilungsvorständin Mag. DI Miriam Zidek

SPENGERGASSE	**HÖHERE TECHNISCHE BUNDESLEHRANSTALT SPENGERGASSE**
	Fachrichtung: **Medizininformatik** Ausbildungsschwerpunkt: **Medizininformatik**

DIPLOMA THESIS

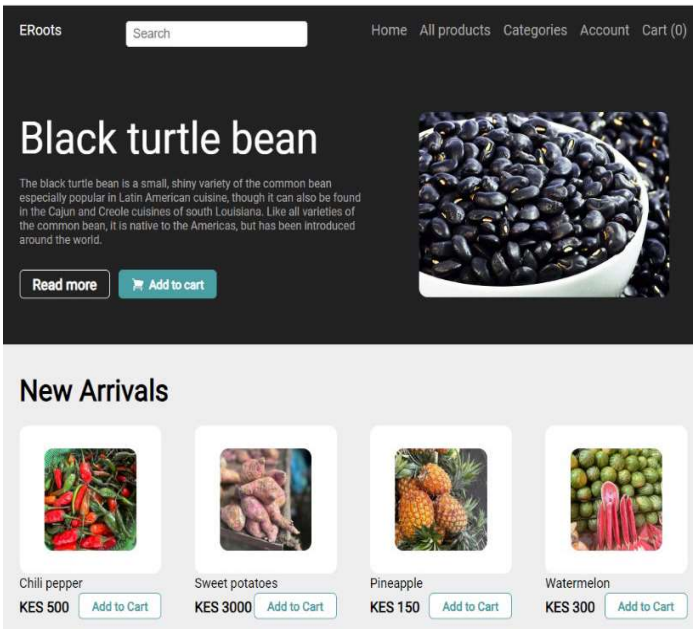
Documentation

Author(s)	Everlyn Gathoni Njeri Sarah Jessica Paukner Valerian Braun Elias Andrei
Form Academic year	Higher Technical Facility 2023/24
Topic	Creation of a web application for the organic cultivation for a Kenyan women self-help group
Co-operation partners	Mutundu Women Self Help Group

Assignment of tasks	**Everlyn Gathoni Njeri**: How can data persistence be planned and implemented in web applications with authentication? **Sarah Jessica Paukner**: Provision and implementation of security, planning and development of the user experience of the web application. **Valerian Braun**: How to implement a web application that deploys HTTP endpoints and receives events, as well as the development of this system? **Elias Andrei**: Planning and implementation/development of the deployment, testing and quality control of the web application.

Realisation	**Next.js:** A React framework for creating server-side rendered web applications. **MongoDB with Mongoose:** A NoSQL database for storing and managing data. **Google OAuth**: An authentication mechanism from Google that allows users to log in to your application with their Google accounts. **Tailwind CSS:** A CSS framework based on utility classes that makes it easy to create custom designs without writing CSS. **Vercel** : A platform for deploying and managing server-side rendered React applications that integrates seamlessly with Next.js. **AWS S3:** An object storage service from Amazon Web Services that allows large amounts of data to be securely stored and accessed from anywhere. **IONOS:** A web hosting company that offers various services such as domain registration, web hosting and server applications.

Results	Functionality in the following areas: • Administration interface • Customer interface • Product management • Reservation management • Database administration

Illustrative graph, photo (incl. explanation)	Logo EVERGREEN ROOTS *Figure 1-3: Logo* The web application under the domain „eroots.store“.  *Figure 1-4: Client webapplication*

Participation in competitions Awards	

Accessibility of diploma thesis	In the school library of the HTLuBVA Spengergasse

Approval (date / signature)	Examiner Marcel Schwarz, Florian Hell	Head of College / Department Mag. DI Miriam Zidek

Content

1 Management Summary 10
1.1 Project goals 10
2 Task description "How can data persistence be planned and implemented in web applications with authentication?" by Everlyn Gathoni Njeri 13
2.1 Basic principles and techniques? 14
2.1.1 Database Selection: 14
2.2 Authentication and Security Measures 16
2.2.1 Email 16
2.2.2 Google OAuth 16
2.3 Solution approaches 17
2.3.1 How does MongoDB work with Next.js? 17
2.3.2 How does Mongoose work with Next.js? 19
2.3.3 How does AWS S3 work with Next.js? 20
2.3.4 How does Google Oauth work with Next.js? 22
2.4 Implemented Components 24
2.4.1 Authentication and authorisation in Next.js with NextAuth.js and MongoDB 24
2.4.2 Connection to MongoDB in Next.js applications 26
2.4.3 Streamlined database interaction with Mongoose: An ODM for MongoDB in Node.js applications. 28
2.4.4 Modelling products in Node.js applications with Mongoose 28
2.5 Errors that occurred 31
2.5.1 Forbidden error when uploading a photo 31
2.5.2 Error while posting a new Product 33
2.5.3 Access blocked: The request of this app is invalid. 34
2.6 Lessons Learned 36
2.6.1 GitHub Challenges and Team Communication 36
2.6.2 Flexibility and Adaptability in Software Development 36
2.7 Accompanying protocol according to § 9 para. 2 PrO 37
3 Task "Security & User Experience" by Sarah Jessica Paukner 47
3.1 Security Fundamentals 48
3.1.1 Encryption 48
3.1.2 User Authentication 48
3.1.3 Domain Safety 48
3.2 User Experience Fundamentals 49
3.2.1 Responsive Design 49

3.2.2 E-Commerce Layout 49
3.3 Security Approaches 50
3.3.1 HTTP Security 50
3.3.2 Self-Implementation 50
3.3.3 Third-Party Options 50
3.4 User Experience Approaches 51
3.4.1 Next.js 51
3.4.2 TailwindCSS 51
3.4.3 Bootstrap 51
3.5 Security Results 52
3.5.1 IONOS SSL Certificate 52
3.6 User Experience Results 56
3.6.1 Admin Layout 56
3.6.2 End User Layout 57
3.7 Web Page „Categories“ Results 59
3.7.1 Use Case 59
3.7.2 Create, Delete Or Edit Categories With Data Persistence 60
3.7.3 Admin Layout 62
3.8 Web Page „Featured Product“ Results 63
3.8.1 Implementation 64
3.8.2 Featured Product Layout 64
3.9 Errors and their Solutions 66
3.9.1 Failed to compile 66
3.9.2 Create Next-App 67
3.9.3 Server Error 68
3.10 Lessons Learned 70
3.10.1 Observation of Stress Escalation: 70
3.10.2 Impact on Team Dynamics: 70
3.10.3 Insights on Time Management: 70
3.10.4 GitHub Integration Issues: 70
3.10.5 Handling Problems at Critical Junctures: 70
3.10.6 Challenges in Motivation: 70
3.10.7 Realization of Project Scope: 70
3.11 Accompanying protocol according to § 9 para. 2 PrO 71
4 Aufgabenstellung: Wie kann eine Webapplikation implementiert und entwickelt werden, die HTTP- Endpunkte bereitstellt und Ereignisse empfängt von Valerian Braun. 81

4.1 Grundlagen 82
4.1.1 Was ist eine Web-Applikation 83
4.1.2 Was sind http Endpoints 83
4.2 Lösungsansätze 84
4.2.1 Javascript/Node/Next.js 84
4.2.2 Python 88
4.2.3 Java 89
4.2.4 Ruby 90
4.2.5 PHP 91
4.2.6 Go (Golang) 92
4.2.7 C# 93
4.2.8 Ergebnis 94
4.3 Ergebnisse 95
4.3.1 Create product page with all and one product 95
4.3.2 Graph 104
4.3.3 Privacy Policy/ Impressum und Footer 108
4.3.4 Aufgetretene Fehler 114
4.4 Lessons Learnd 116
4.4.1 Was lief gut? 116
4.4.2 Was lief schlecht? 116
4.5 Begleitprotokoll gemäß § 9 Abs. 2 PrO 117
5 Task description “Planning and implementing/developing of deployment, testing and quality assurance” by Elias Andrei. 125
5.1 Basics 126
5.1.1 Deployment 126
5.1.2 Testing 127
5.1.3 Quality Assurance in Software Development 131
5.2 Solution Approaches 133
5.2.1 Deployment 133
5.2.2 Testing 138
5.2.3 Quality Assurance 147
5.3 Results 148
5.3.1 Deployment 148
5.3.2 Testing 148
5.3.3 Additional Results 150
5.3.4 Lessons Learned and Difficulties 153
5.4 Accompanying protocol according to § 9 para. 2 PrO 155

6 Index of Figures 160
7 Table of abbreviations 163
8 List of sources and references 165
9 Appendix 170

1 Management Summary

In Kiambu, Kenya, a dedicated women's self-help group actively promotes local produce sales in their community. They are on a mission. They want to expand their market reach and tackle the chronic problem of food waste caused by the spoilage of unsold produce. Recognising the transformative potential of technology, they're keen to build a robust web-based platform to do this. This digital marketplace will serve as a centralised tool, streamlining operations and increasing accessibility for both those producing and those consuming.

Showcasing the vibrant diversity of their agricultural offering, the proposed platform will offer a comprehensive range of locally sourced products. Customers will enjoy the convenience of browsing this virtual marketplace. They will be able to place orders with ease and choose to pay for products when they are collected. The women's self-help group aims not only to modernise their business but also to promote economic empowerment within their community by integrating modern technology into their sales strategy. They aim to create a sustainable solution that benefits both their business and the local agricultural ecosystem through this innovative approach.

1.1 Project goals

An E-Commerce web application with an admin dashboard to upload new products and a client dashboard to order products and make an appointment for the evaluation of the farmed good getting sold.

- Client interface
  - product page with all products created.
  - one product page created.
- User
  - user created and edited.
  - products ordered.
  - view previous orders with a history view.
  - implement a search functionality.
- Admin Interface
  - products with data persistence created, edited, and deleted.
  - categories with data persistence created, edited, and deleted.
  - appointments booked (reservations)
  - products, categories and admin and user login data in the database stored.

2 Task description “How can data persistence be planned and implemented in web applications with authentication?” by Everlyn Gathoni Njeri

The thesis explores data persistence planning and implementation in web applications. It focuses on those with robust authentication frameworks and examines the relationship between data persistence strategies and authentication mechanisms. The thesis highlights the seamless integration of these two aspects.

Data persistence is crucial for user interactions, system functionalities, and data integrity in web development. Sensitive information and access to privileged resources are protected by the authentication layer. Creating resilient and secure web applications requires aligning these two pillars.

This investigation explores the methods and techniques used to plan and execute data persistence in authenticated web environments. The text examines various database solutions, including relational databases and NoSQL options.

To bridge the gap between theory and practice, the text discusses theoretical concepts and provides implementation strategies. The aim is to provide a deeper understanding of the project in this section of the thesis.

2.1 Basic principles and techniques?

In this chapter, the focus is on identifying the most suitable technology for this project.

2.1.1 Database Selection:

When developing the Evergreen Roots application, the focus was on selecting a suitable database solution. The aim was to find a solution that could effectively fulfil the requirements for data persistence and the ability to use CRUD commands (Create, Read, Update, Delete). It also had to have a short learning curve and strong adaptability. After a careful evaluation, various databases were considered, with the decision depending on factors such as data type, scalability, performance requirements, and the skills of the development team.

MySQL
MySQL is a widely used open-source relational database management system (RDBMS) for web development applications.

Its robust feature set, strong data consistency, and ACID compliance make it suitable for various use cases, including Next.js applications. However, it was not chosen for this project. (cf. MySQL, 2001)

The database could have been saved locally, but since the web application was intended for publication, MongoDB was chosen as the database. This was because it was already available on a server, and we not only had access to it throughout development but also in deployment.

MongoDB
MongoDB is a popular NoSQL database management system known for its adaptability, scalability, and fast data handling capabilities. It can accommodate dynamic data structures and stores information in BSON format, allowing for versatile and complex data models. Unlike relational databases, MongoDB does not have rigid schema constraints, making it a more flexible framework for storing and accessing data.(cf. MongoDB, 2023)
MongoDB is the preferred choice because it seamlessly integrates with Next.js, making it highly compatible. It is also cost-effective, making it accessible to businesses with a low-entry barrier. Additionally, it is easily accessible over the internet. Businesses can use MongoDB as the cornerstone of their data infrastructure to achieve flexibility, scalability, and accessibility. This can help propel them towards innovation and growth in the digital realm.

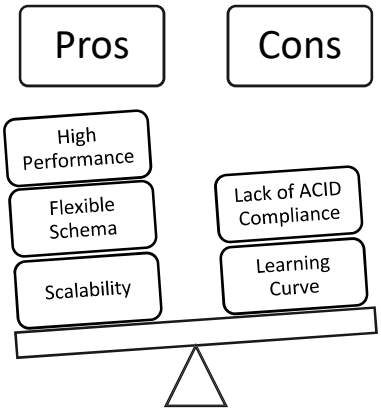


Figure 2-1: Pro and Cons of MongoDB

Mongoose

It was decided to use the Mongoose library, which is an ODM for MongoDB, to connect MongoDB to Next.js. Mongoose offers a user-friendly interface for interacting with MongoDB databases by adding an abstraction layer. (cf. Hall, 2022)

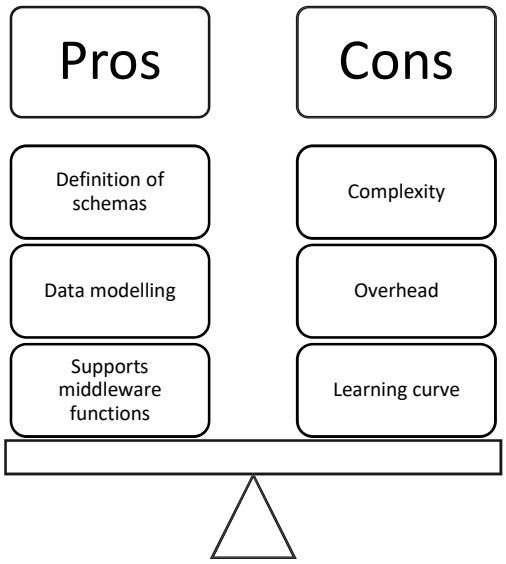


Figure 2-2: Pro and Cons of Mongoose

Amazon Simple Storage Service (Amazon S3)

It took a long time to think about how the images of the products should be stored. The images had to be accessed during development and deployment, and they also had to be linked to the MongoDB. AWS S3 is used to create a unique URL from the images, which can then b(cf. Roscoe, 2024)e saved as a string in the product shema.

“Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. “ (Amazon, 2024)

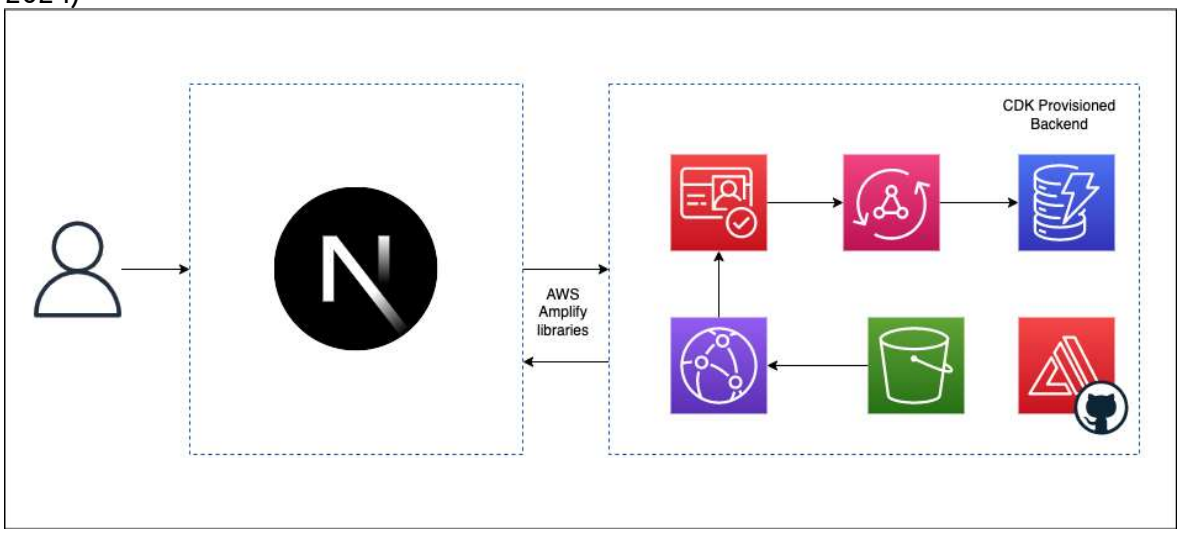


Figure 2-3: A possible Nextjs Application with AWS : (Liendo, Michael, 2022)

2.2 Authentication and Security Measures

Next.js was selected as the framework for developing the application. Two pages were created: one for product sales and an admin page exclusively for administrators. The sales page enabled users to view and order products, as well as view their order history. On the admin page, administrators could add and edit products and save them to the database.

Comprehensive authentication measures ensure the security of the Evergreen Roots application for both clients and administrators. Users can log in, while special admin user roles restrict access to authenticated administrators. This measure increases the security of the admin side by ensuring that only authorized personnel have access to the administration functions and by preventing unauthorised access by unauthenticated users.

2.2.1 Email

NextAuth provides the ability to use email for login, with authentication links sent through platforms such as Slack. This functionality allows users to access an application by providing their email address, and then using a unique authentication link sent to their email inbox. By clicking on this link, users gain access to their accounts without having to enter a password. However, this feature was not implemented due to security concerns related to the handling of passwords and the need to securely store and hash them. (cf. Roscoe Will, 2024)

2.2.2 Google OAuth

Google OAuth is an authorisation protocol developed by Google that enables users to grant an application access to their Google data without disclosing sensitive information such as login credentials. The decision to create an admin page and a client page meant that the both pages had have a login. On the client side google Oauth enables every Google User to login without disclosing their personal information, so that only authorised personnel could access it. Google was therefore chosen as the appropriate option.It allows applications to access Google services on behalf of the user.(cf. Google, 2024a)

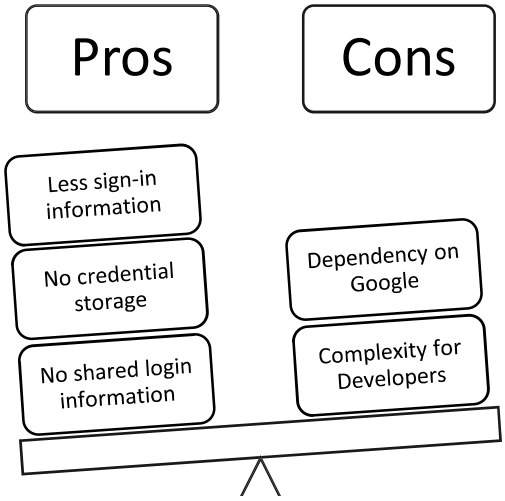


Figure 2-4: Pro and Cons of Google Oauth

2.3 Solution approaches

This section of the document is an outline of how the databases and authorisation methods that were used in this project work.

2.3.1 How does MongoDB work with Next.js?

Prerequisites:

Before starting to integrate MongoDB with a Next.js application, ensure that the following requirements are met: (cf. Kesav, 2022)

- MongoDB Atlas account or local installation: Get access to a MongoDB Atlas account or set up a local MongoDB installation on the target system.
- Node.js installation: Ensure that Node.js is installed on the system, with the minimum supported version of Node.js being 12+.

Integration Steps:

Database setup:

- Set up a MongoDB database environment, either via the MongoDB Atlas or via a locally installed instance of MongoDB. This will involve the creation of a new cluster and the definition of a database within the cluster to store the application data.

Retrieve the connection string: (cf. Kesav, 2022)

- Access the MongoDB Atlas dashboard, locate the designated cluster and navigate to the "Connect" option. Follow the instructions provided to retrieve the connection string, which is typically formatted as follows "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/<dbname>?retryWrites=true&w=majority". Replace the placeholders (<username>, <password>, <cluster>, <dbname>) with the appropriate credentials for the MongoDB atlas and the name of the database.
- Create the utility file:
- Create a new file, such as mongoDB.js, in the utils or lib folder within the Next.js project directory. This file serves as a utility for managing connections to MongoDB.
- Writing code within this file that connects with MongoDB using preferred libraries, like the official MongoDB node.js driver or any other compatible libraries.

```
import { MongoClient } from 'mongodb';

export default async function handler(req, res) {
  const uri = process.env.MONGODB_URI;

  const client = new MongoClient(uri, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  });

  try {
    await client.connect();
    console.log('Connected to MongoDB');

    // Do something with the connected client, like query or insert data

    res.status(200).json({ message: 'Connected to MongoDB' });
  } catch (error) {
    console.error('Error connecting to MongoDB:', error);
    res.status(500).json({ message: 'Error connecting to MongoDB' });
  } finally {
    await client.close();
    console.log('MongoDB connection closed');
  }
}
```

Figure 2-5: Example of a connection file: (ChatGPT, 2024)

2.3.2 How does Mongoose work with Next.js?

To use Mongoose in a Next.js application, one can proceed similarly to any other Node.js application. The following are the basic steps to integrate Mongoose into Next.js:

Installing Mongoose: Firstly, the mongoose package needs to be installed. This can be done by running npm install mongoose in a terminal.

```
Npm install mongoose
```

Figure 2-6: How to install mongoose.

Import Mongoose: The Mongoose package must be imported into the files where

```
import mongoose from ‘mongoose‘;
```

Figure 2-7: How to import mongoose.

Mongoose is to be in use. This is usually done at the top of the file with the command "import mongoose from 'mongoose'".

Connect to the database: Developers need to connect to the MongoDB database. Typically, this is done in a separate module that exports a connection establishment function. The files that access the database will use this function.

```
Export async function connectToDatabase( ) {
        Try {
                Const uri = process.env.MONGODB_URI;
                Await mongoose.connect (uri, {
                        useNewUrlParser: true,
                        userUnifiedTopology: true,
        });
        Console.log(‘Connected to MongoDB’);
} catch (error) {
        Console.error(‘Error connection to MongoDB:’, error.message);
        }
}
```

Figure 2-8: Exporting the database connection.

Defining schemas and models: Developers can define schemas and models for their data by using the Mongoose API. These schemas define the structure of the data and how it should be stored in the database. The models allow developers to access, create, read, update and delete the data. (cf. Hall, 2022)

```
Import { connectToDatabase } from ‘../../utils/mongoose’;
```

Figure 2-9: Importing the connection to the Database.

Developers can use Mongoose in their Next.js components to access the database once the connection has been made and the schemas defined. This can include reading data for display on the user interface, creating new data or updating existing data.

This procedure enables the use of Mongoose in a Next.js application for interaction with a MongoDB database.

```
const blog = new Schema({
  title: String,
  slug: String,
  published: Boolean,
  author: String,
  content: String,
  tags: [String],
  createdAt: Date,
  updatedAt: Date,
  comments: [{
    user: String,
    content: String,
    votes: Number
  }]
});
```

Figure 2-10: A possible schema. (Hall, 2022)

2.3.3 How does AWS S3 work with Next.js?

Before beginning the integration process, it is important to confirm that the following requirements are met:

- AWS account: Access to an AWS account with permissions to create and manage S3 buckets.
- Next.js application: Have a Next.js application set up and ready to integrate with AWS S3.

Integration Steps:

Create an S3 Bucket:

- Log in to the AWS Management Console and navigate to the S3 service.
- Click on the "Create bucket" button and follow the prompts to configure the bucket settings, such as region, bucket name, and access permissions.

Configure Bucket Policy:

- After the creation of the bucket, configure a bucket policy to define access permissions for the items stored in the bucket. This policy should specify who is allowed to access the objects and under what conditions they can access them.

Generate Access Credentials:

- Create an IAM user or role with access permissions to the S3 bucket. To grant the required permissions, assign appropriate permission policies, such as AmazonS3FullAccess or custom policies.

Install AWS SDK for JavaScript:

- Within the Next.js application directory, install the AWS SDK for JavaScript using npm or yarn:

```
npm install aws-sdk
```

Figure 2-11: Install aws in nextjs.

- Set up AWS credentials in the Next.js application either by using environment variables, AWS CLI configuration, or AWS IAM instance profile.

Initialize AWS S3 Client:

- In the Next.js application code, initialize an instance of the AWS S3 client using the configured credentials and the appropriate AWS region.

Perform CRUD Operations:

- The Next.js application uses the AWS S3 client instance to perform CRUD (create, read, update, delete) operations on objects stored in the S3 bucket.
- Examples of operations include: Uploading files to the bucket, downloading files from the bucket, listing objects in the bucket, and deleting objects from the bucket.

2.3.4 How does Google Oauth work with Next.js?

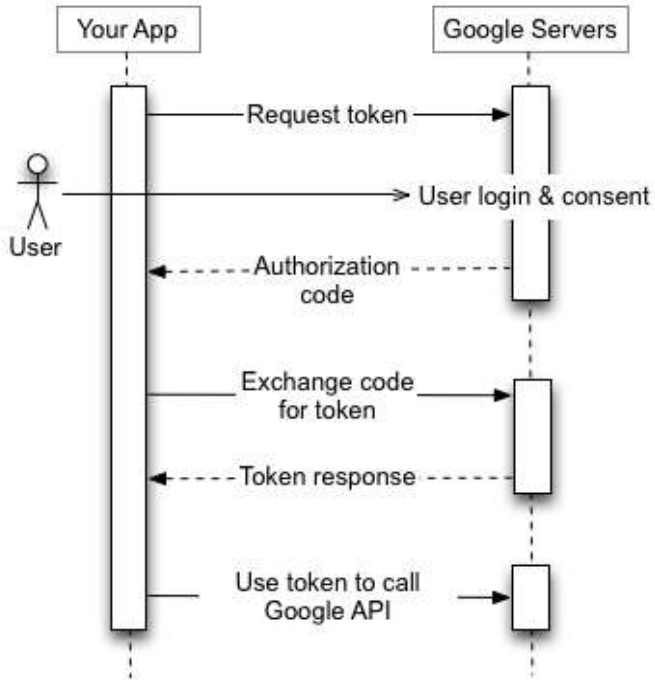


Figure 2-12: Nextjs with Google OAuth API

(Google, 2024b)
The OAuth 2.0 process, also known as implicit grant flow, is designed for applications that only access APIs when the user is present and do not store sensitive information. Before implementing OAuth 2.0 authorisation, it is necessary to enable the required APIs in the API Console and create authorisation credentials. This involves creating OAuth client IDs for a web application and identifying the privileges the application needs to access Google APIs. (cf. *Google | NextAuth.Js*, 2023; cf. Google, 2024a)
Next.js implements Google OAuth in several steps to facilitate user authentication and authorization seamlessly. Here is a breakdown of the typical flow:

- Initialization: The process begins when a user attempts login or access to a restricted part of the application that requires authentication. By redirecting the user to Google's OAuth 2.0 endpoint, a Next.js application initiates the OAuth flow.
- Redirect to Google: The Next.js application constructs a URL that includes parameters such as client_id, redirect_uri, response_type, and scope. This URL directs the user's browser to Google's authentication server.(cf. *Google | NextAuth.Js*, 2023; cf. *OAuth | NextAuth.Js*, 2023)

For production: `https://{YOUR_DOMAIN}/api/auth/callback/google`

For development: `http://localhost:3000/api/auth/callback/google`

Figure 2-13: redirect url.

- User authentication: The user is prompted to sign in if they have not already done so when they reach Google's authentication server. Google will verify the user's credentials and prompt the user to authorise access to their data according to the scope of access specified.
- Grant the permission: Google redirects the user back to the Next.js application using the specified redirect_uri if the user agrees to the requested permissions. The redirect includes an authorisation code as a query parameter in the URL.
- Token exchange: By sending a request to Google's token endpoint, the Next.js application extracts the authorisation code from the URL and exchanges it for an access token. The authorisation code, client credentials (client_id and client_secret) and the redirect_uri are included in this request.
  (cf. *Google | NextAuth.Js*, 2023; cf. *OAuth | NextAuth.Js*, 2023; cf. Google, 2024a)

```
import GoogleProvider from 'next-auth/providers/google'

    …
        providers: [
         GoogleProvider({
            clientId: process.env.GOOGLE_ID,
            clientSecret: process.env.GOOGLE_SECRET
        })]
```

Figure 2-14: Google provider

- Access token retrieval: Google responds with an access token upon successful exchange. The Next.js application stores this access token in a secure manner, typically in a server-side storage or in a secure HTTP cookie, for use in the future when making authorised API requests on behalf of the user.
- API requests: A Next.js application can now make authorised requests to Google APIs on behalf of the user with the access token in hand. To prove the user's identity and permissions, these requests include the access token in the authorisation header.
- Token Refresh (Optional): Access tokens have a limited lifespan for security reasons. If the access token expires, a Next.js application can use a refresh token (if provided during the initial token exchange) to obtain a new access token without requiring user reauthentication.
  (cf. Google, 2024a; cf. *OAuth | NextAuth.Js*, 2023)

This flow ensures a secure a seamless authentication and authorization of users in a Next.js application using Google OAuth.

2.4 Implemented Components

This chapter provides an overview of the implementation of the component.

2.4.1 Authentication and authorisation in Next.js with NextAuth.js and MongoDB

In this implementation, authenticating and authorising is done in a Next.js application using NextAuth.js and a MongoDB database. The code has two parts: the configuration of the authentication options and an administrator verification function.

In this section of the code, various modules and configurations for authentication and authorisation are imported and initialised.

```
import NextAuth, { getServerSession } from 'next-auth'
import { MongoDBAdapter } from "@next-auth/mongodb-adapter"
import clientPromise from "../../../lib/mongodb"
import GoogleProvider from 'next-auth/providers/google'


const adminEmails = new
Set(['***@gmail.com','***@gmail.com','***@gmail.com','***@gmail.com',
'***@gmail.com','Njugunapenina44@gmail.com']);
```

Figure 2-15: Only admin emails can login.

Authentication options (authOptions)

The authOptions define the configuration for authentication. These include:

- Secret : A secret for secure session management and the configuration of the Google provider for OAuth authentication.

```
export const authOptions = {
  secret: process.env.NEXTAUTH_SECRET,…
```

Figure 2-16: NextAuth secret

- Provider: Configuration of the Google provider for OAuth authentication.

```
providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_ID,
      clientSecret: process.env.GOOGLE_SECRET
    })
  ]
```

Figure 2-17: Provider in nextauth.

- Adapter: Use of the MongoDB adapter for storing session data and user information.

```
adapter: MongoDBAdapter(clientPromise)
```

Figure 2-18: Connection to the database.

- Callbacks: A function that is called to check the session and ensure that and ensure that only administrators have access.

```
callbacks: {
  session: ({ session, token, user }) => {
    if (adminEmails.has(session?.user?.email)) {
      return session;
    } else {
      return false;
    }
  },
}
```

Figure 2-19: Callback implementation

Administrator check (isAdminRequest)

The isAdminRequest function checks whether the request originates from an administrator. To do this, the user's session is retrieved via the server and the user's email address is compared with a predefined list of administrator emails. If the user is not identified as an administrator, an error message is returned, and the request is ended with a 401-status code.

```
import NextAuth, { getServerSession } from 'next-auth'
…
export async function isAdminRequest(req,res) {
  const session = await getServerSession(req,res,authOptions);

  if (!adminEmails.has(session?.user?.email)) {
    res.status(401);
    res.end();
    throw 'not an admin!';
  }
}
```

Figure 2-20: Function for only admin login

2.4.2 Connection to MongoDB in Next.js applications

This implementation provides an efficient and robust connection to a MongoDB database for Next.js applications. This approach was inspired by a best practice documented in the official Vercel sample project for Next.js.

Environment configuration

The code starts by checking an environment variable called "MONGODB_URI". If this variable does not exist or is invalid, an error message is triggered to inform the developer that the configuration is incomplete.

```
Import { MongoClient } from “mongodb“;


If (! Process.env.MONGODB_URI) {
        Throw ne Error(‘Invalid/Missing environment variable: “MONGODB_URI” ‘);
}
const uri = process.env.MONGODB_URI;
const options = {};o
```

Figure 2-21: Checking if there is a connection to MongoDB.

Connection establishment

The connection to the MongoDB database is established via a MongoDB client. The URI for the connection is read from the environment variable. No additional options are used either.

Differentiation between development and production environments

To ensure efficient handling of the MongoDB connection in development and production environments, a distinction is made between the two environments.

A global variable _mongoClientPromise is used in the development environment to ensure that the client connection is maintained across module reloads by HMR (Hot Module Replacement). This improves the performance of the development process, as

```
if (process.env.NODE_ENV === “development”) {
        if (!global._mongoClientPromise) {
                client = new MongoClient (Uri, options);
                global._mongoClientPromise = client.connect();
        }
        clientPromise = global._mogoClientPromise;
}
```

Figure 2-22: Ensuring that a database connection is maintained.

the connection to the database is only established once, even if modules are loaded several times.

In the production environment, a global variable is not used due to the special features of Next.js and the production environment. Instead, the client connection is created directly, and the connection promise is exported. This ensures a reliable and efficient connection in production environments without the need for a global variable.

```
else {
        client = new MongoClient(uri, options);
        clientPromise = client.connect();
}
```

Figure 2-23: Filter if there is already a connection.

Export

Finally, the connection promise is exported so that it can be used in other modules. By using a promise, the MongoDB connection can be used asynchronously, which increases the scalability and performance of the application.

```
Export default clientPromise;
```

Figure 2-24: Export the MongoDB connection.

2.4.3 Streamlined database interaction with Mongoose: An ODM for MongoDB in Node.js applications.

This section explores the integration of Mongoose, an ODM (Object Data Mapping) for MongoDB, into Node.js applications to simplify and structure database interaction. Mongoose facilitates the use of MongoDB by providing an object-oriented interface to map MongoDB documents into JavaScript.

```
Import mongoose from "mongoose";
Export function mongooseConnect () {
        const uri = process.env.MONGODB_URI;
        if (mongoose.connection.readyState === 1) {
                return mongoose.connection.asPromise();
        } else {
                const uri = process.env.MOGODB_URI;
                return mongoose.connect(uri);
        }
}
```

Figure 2-25: Connect to mongoose.

Explanation of the code:

- The code includes the function mongooseConnect() which manages the database connection. The Mongoose module is imported to make MongoDB functions and methods available in the application. The function definition for mongooseConnect() is then provided and exported for use in other parts of the application.
- The connection URI for MongoDB is retrieved from the environment variables and is used to establish a connection to the database by calling mongooseConnect().
- Checking to connect is done by querying the Mongoose ready state.
- If there is already a connection, then a promise representing the existing connection will be returned. Otherwise, a new connection to the MongoDB database is established.

2.4.4 Modelling products in Node.js applications with Mongoose

Product modelling in Node.js applications using Mongoose is discussed in this section. Mongoose is a module for the mapping and management of MongoDB documents as JavaScript objects. The Mongoose product model is defined in models/Product.js to control how the application structures and validates product data.

Here is a detailed explanation of the code:

Importing modules: The required modules are imported, including mongoose for the use of Mongoose and Category for the category model.

```
import { Category } from "./Category";
const { Schema, model, models, default: mongoose } = require("mongoose");
```

Figure 2-27: Import Statement Code

```
const ProductSchema = new Schema({
   titel: { type: String, required: true },
   description: String,
   price: { type: Number, required: true },
   images: [{ type: String }],
   category: {type:mongoose.Schema.Types.ObjectId, ref:'Category'},
   properties: {type:Object},
},{
   timestamps: true
});
```

Figure 2-26: Product Schema Code

Schema declaration: A Mongoose schema (ProductSchema) is defined to control the structure and validation of the product information. It contains fields such as "title", "description", "price", "images", "category" and "properties".

Reference to category: The Category field is a reference to a category object in the database. It is a reference to a category object in the database and is a reference to a category object in the database. To refer to the corresponding category object, the Mongoose ObjectId type is used.

Timestamps option: The 'timestamps: true' option automatically adds timestamps for creating and updating products in Mongoose.

```
timestamps: true
```

Figure 2-28: Timestamp

Exporting the model: The Product model (Product) is created by either using existing models or by creating a new model with the schema. It is then exported for use in other parts of the application.

```
export const Product = models.Product || model('Product', ProductSchema);
```

Figure 2-29: Exporting product schema

This code enables the control of product data structure and validation in Node.js applications, as well as the storage and retrieval of products in the MongoDB database. Mongoose simplifies and standardises interaction with the database.

```
category: {type:mongoose.Schema.Types.ObjectId, ref:'Category'},
```

Figure 2-30: implement Catagories

2.5 Errors that occurred

This point thematises mistakes, that happed during development and what their solution was.

2.5.1 Forbidden error when uploading a photo

```
▼GET https://eroots-next-ecommerce.s3.amazonaws.com/1689263872977.jp 1689263872977.jpg:1
g 403 (Forbidden)
  Image (async)
  commitMount                      @ react-dom.development.js:11038
  commitLayoutEffectOnFiber        @ react-dom.development.js:23407
  commitLayoutMountEffects_complete @ react-dom.development.js:24688
  commitLayoutEffects_begin        @ react-dom.development.js:24674
  commitLayoutEffects              @ react-dom.development.js:24612
  commitRootImpl                   @ react-dom.development.js:26823
  commitRoot                       @ react-dom.development.js:26682
  finishConcurrentRender           @ react-dom.development.js:25981
  performConcurrentWorkOnRoot      @ react-dom.development.js:25809
  workLoop                         @ scheduler.development.js:266
```

Figure 2-31: Error Message

The error encountered indicates a CORS (Cross-Origin Resource Sharing) issue while trying to post an image from the external domain (eroots-next-ecommerce.s3.amazonaws.com) to the local development environment (localhost:3000). This error occurs because the AWS S3 bucket does not include the necessary CORS headers to allow requests from your localhost domain.

To resolve this issue it was needed to configure the CORS on the AWS S3 bucket to allow request from specific domains. Following these steps the error was solved.

Access the AWS S3 Management Console: Log in to the AWS Management Console and navigate to the S3 service.

Modified the bucket policy to allow public access to the objects stored in the bucket.

Bucket-Richtlinie Bearbeiten Löschen

Die in JSON geschriebene Bucket-Richtlinie bietet Zugriff auf die im Bucket gespeicherten Objekte. Bucket-Richtlinien gelten nicht für Objekte, die zu anderen Konten gehören. Weitere Informationen

Kopieren

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::eroots-next-ecommerce/*"
    }
  ]
}
```

Figure 2-32: Updated bucket request.

Add the following CORS Rules to allow requests from the listed domains:

Cross-Origin Resource Sharing (CORS) Bearbeiten

Die CORS-Konfiguration, geschrieben in JSON, bestimmt für Client-Webanwendungen, die in einer Domain geladen sind, eine Möglichkeit zur Interaktion mit Ressourcen in einer anderen Domain. Weitere Informationen

Kopieren

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD",
      "PUT",
      "POST",
      "DELETE"
    ],
    "AllowedOrigins": [
      "http://localhost:3000",
      "https://admin.eroots.store",
      "https://eroots.store"
    ],
    "ExposeHeaders": [],
    "MaxAgeSeconds": 3000
  }
```

Figure 2-33: Updated CORS requests.

2.5.2 Error while posting a new Product

While attempting to upload an image into a new product, an unexpected error occurred, disrupting the process. The error message, "Unhandled Runtime Error TypeError: props.list is not iterable," appeared, indicating a critical issue within the codebase.

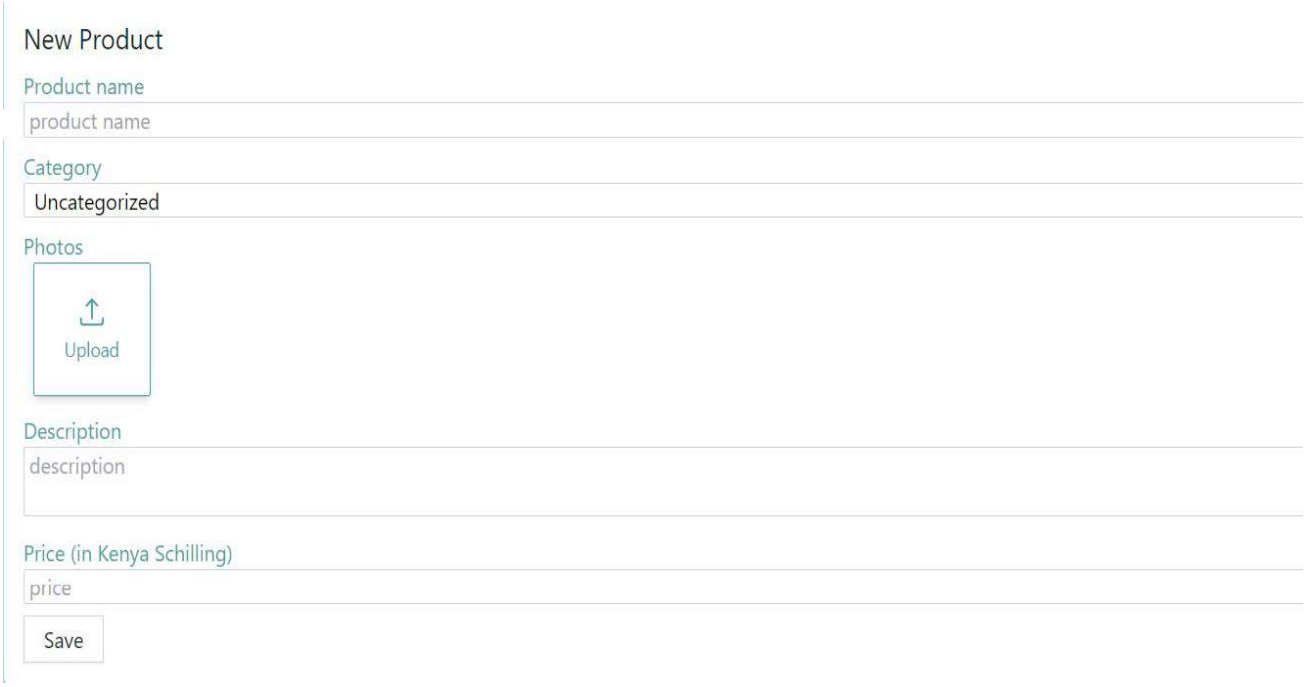


Figure 2-34: Save Produkt page

This bug affects the ReactSortable component, which is responsible for sorting and managing uploaded images in the interface. The component requires the 'list' property to be iterable in order to effectively manipulate and display images. However, when the code was examined, it was discovered that the 'images' variable, which was supposed to contain the list of images, was not iterable.
This happened in this line of code:

```
<ReactSortable list={images} className="flex flex-wrap gap-1" setList={uploadImagesOrder}>
```

Figure 2-35: Lists of Image

To resolve this issue and restore functionality, it was necessary to transform the 'images' variable into an iterable object before passing it to the ReactSortable component. This was accomplished by converting 'images' into an array, ensuring that it met the requirements for iterability.

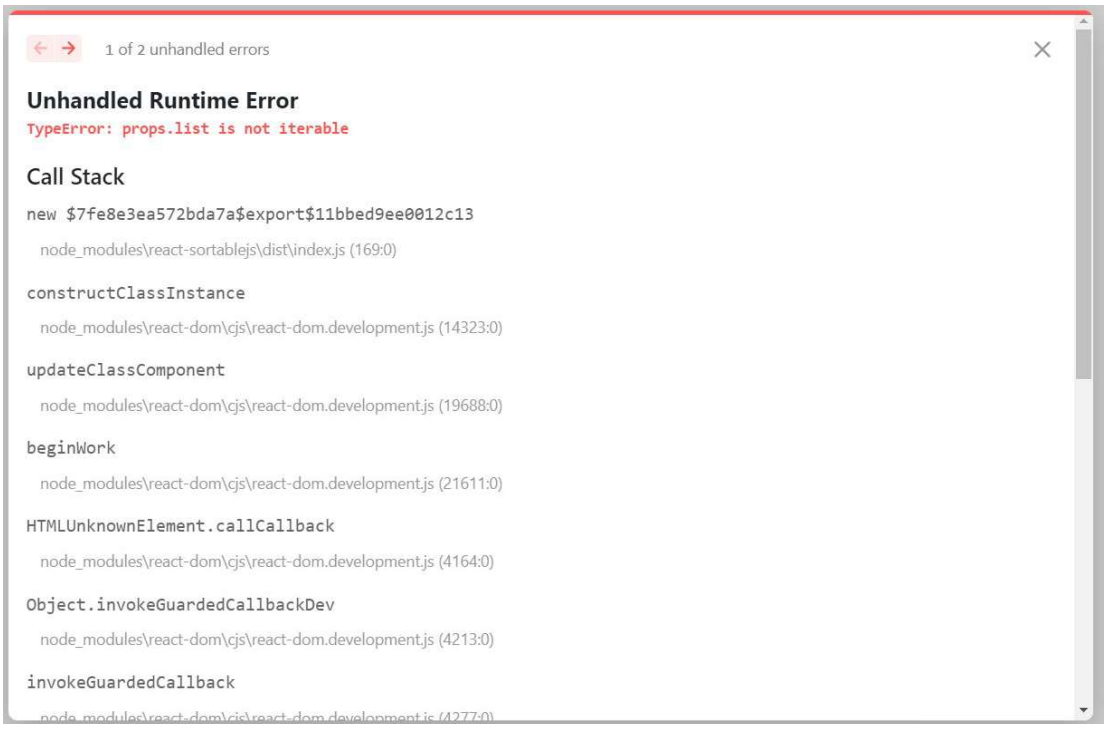


Figure 2-36: Error Message

2.5.3 Access blocked: The request of this app is invalid.

The error known as 'redirect_uri_mismatch' occurs when the redirect URI provided in the OAuth request does not match the one pre-registered with the authorization server. The redirect URI is a crucial link in the authentication process, allowing the authorization server to redirect the user back to the application after authentication.

Error 400 can be challenging, especially when accompanied by the 'redirect_uri_mismatch' message. To solve this issue, one had to adjust the redirection URL through diligent troubleshooting and iterative exploration.

Navigating the intricacies of the OAuth protocol and its error codes requires attention to detail and perseverance. Insights were gained with each attempt, leading to a better understanding of the underlying mechanics. Through experimentation and adaptation, the solution to this error gradually became clear.

By modifying the redirection URL to match the specifications required by the authorization server, the obstacle was overcome, allowing for smooth authentication and subsequent user redirection. This transformative process underscored the importance of flexibility, adaptability, and a systematic approach in navigating technical challenges within the realm of application development and integration.

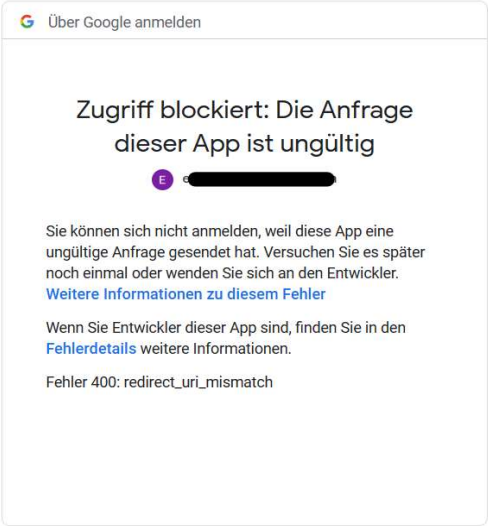


Figure 2-38: Access blocked error message

Figure 2-37: Screnshot of redirect uls in the google console.

2.6 Lessons Learned

Navigating development in a new environment, like when you're using Visual Studio Code and Next.js, was tough. It felt like trying to find your way through a maze—super confusing at first. But even in all the chaos, there were chances to learn and grow.

2.6.1 GitHub Challenges and Team Communication

One big challenge was figuring out how to work together using GitHub. It was like learning a new language, especially with everyone having their own way of doing things. There were definitely some communication mix-ups and even a few arguments early on. But over time, we got better at coordinating our efforts and working as a team.

Communication became super important for us to make progress together. Instead of getting mad or blaming each other when something went wrong, we talked it out and found solutions together. It wasn't easy at first, but the more we practiced, the smoother things got.

2.6.2 Flexibility and Adaptability in Software Development

I also learned the importance of being flexible and adaptable in the fast-paced world of software development. When we had a big problem that affected our whole project, we had to change our workflow to fix it. It took a lot of patience and perseverance, but we embraced the challenge and kept an open mind. And in the end, we came out stronger as a team.

While the project had its share of challenges, we celebrated significant milestones such as the admin log in via Google and saving products into the database and displaying them in the user site.

To be recognised for these achievements gave us a real boost. It made us feel good about what we were doing, and it gave us the motivation to do more. It also reminded us of the reason why we were working so hard - to create something great for our project sponsor. It encouraged us to keep aiming high and showed us that we were on the right track.

2.7 Accompanying protocol according to § 9 para. 2 PrO

Protocol by Everlyn Njeri with date, type of activity (in the case of a meeting, reference to meeting minutes if necessary), effort in hours (to the nearest 5 minutes) and aids used.

Date	Effort	Activity	Tast	Aids used
24.05.2023	1:30	Preliminary project phase	Project name and abbreviation, as well as icons reviewed	Google, Canva
25.05.2023	1:15	Preliminary project phase	Project application started to be filled out with the supervising teacher, thought about the structure and distribution of tasks and external server to host the website (Netlify, Railway.app)	Google
29.05.2023	0:25	Analysis	Researched on the internet via e-commerce websites	Google
29.05.2023	1:25	Analysis	Figured out how to add a MongoDB database and how to use the login with Google for the admin page	Google
29.05.2023	5:55	Implementation	Try to create a first version of the program and create a GitHub repository --> done. Admin dashboard created with MongoDB database connection Through Google, the admin can log in to the dashboard. All change already pushed to GitHub	Visual Code
30.05.2023	5:20	Implementation	Save products in database and display them in admin	Visual Code
31.05.2023	4:25	Preliminary project phase	Project meeting 1: What kind of tool do we use for the project?	Teams
31.05.2023	1:15	Preliminary project phase	Project meeting 2: Who does what? What do we do/need anyway?	Teams
01.06.2023	6:25	Implementation	Database access from my IP address to all IP addresses changed, colour palette found, colours changed and at table products, you can now see all Data of the product: title, description and price	Visual Code

15.06.2023	1:25	Implementation	Adjusted colours of admin page, fixed bugs where the program did not start	Visual Code
15.06.2023	3:10	Implementation	Microsoft Azur student account created. Setting up a server	Visual Code
28.06.2023	2:30	Implementation	Be able to edit products	Visual Code
07.07.2023	3:10	Project management	Project application completed with Sarah and went through with project client --> project proposal accepted by project commissioning	Word
08.07.2023	4:45	Implementation	Delete products and start uploading the images at Products	Visual Code
09.07.2023	3:25	Implementation	Upload product images, upload images but not yet Permanently stored	Visual Code
09.07.2023	0:50	Project management	Discussion about product image storage	
12.07.2023	4:35	Implementation	Upload product images continue, error with API upload, try error find failed	Visual Code
13.07.2023	4:25	Implementation	Trying to fix Api upload error --> didn't work Sarah tries to find it in the code --> bug fixed	Visual Code
13.07.2023	2:40	Implementation	Upload product images for the individual products, give images to the front with drag and drop and when an image is uploaded, a spinner appears	Visual Code
13.07.2023	0:20	Project management	Meeting with Sarah about other changes that have yet to be made Errors found in Add Product	Teams
13.07.2023	0:35	Implementation	Fixed a bug with add product and formatted all files with Shift+Alt+F	Visual Code
14.07.2023	0:35	Project management	informed the project client about progress and that what has already been programmed has been shown	WhatsApp
14.07.2023	0:20	Project management	Discussion with Sarah about product data options: purchase volume in grams (kilograms)	WhatsApp

16.07.2023	1:10	Design	Take product pictures and discuss with the product seller how the products should be presented	Word, WhatsApp
20.07.2023	0:25	Project management	Meeting with Sarah about the progress of the project	WhatsApp
22.08.2023	1:10	Project management	Meeting with team about progress in the project	Teams
07.09.2023	2:40	School	Diploma Thesis Upload Declaration	Word
13.09.2023	1:50	School	Thesis corrected in the database; declaration uploaded again	Visual Code, MongoDB Atlas
14.09.2023	2:40	School	Database for the thesis corrected for spelling mistakes, project proposal goals Smart defined	Visual Code, MongoDB Atlas
19.09.2023	0:50	School	Domain purchased and connected to server (?), domains compared: Easyname and Ionios.	Ionos, Vercel, Visual Code
19.09.2023	0:30	School	Meeting with Professor Schwarz	Word
24.09.2023	1:15	Implementation	ESLint error in admin dashboard improved, attempt to deploy website to Vercel. Front works, back still needs to be deployed --> build error on admin dashboard in Vercel	Visual Code
26.09.2023	0:30	Project management	Discussion with Sarah about publishing on Vercel instead of Amplify	Teams, WhatsApp
26.09.2023	1:10	Implementation	User interface online with the domain eroots.store	Visual Code
02.10.2023	1:40	School	Work on the preliminary study	Word
02.10.2023	0:35	Implementation	Merge frontend and backend again --> Error 500 when publishing on Vercel --> Error improved	Visual Code
03.10.2023	0:50	Preliminary project phase	Discussion with Professor Schwarz about the citation method for the thesis	Word
04.10.2023	1:00	School	Preliminary study Personal Topic 5, PHB discussed and corrected	Word, Zotero

09.10.2023	0:40	School	Preliminary study	Word, Zotero
10.10.2023	0:50	School	System architecture discussed with Professor Pirker, changes according to Eroots-4 meeting minutes v1	Word
11.10.2023	1:50	School	Preliminary study	Word, Zotero
11.10.2023	0:45	Project management	Preliminary study database models inserted	Word, Zotero
13.10.2023	2:30	Project management	Preliminary study of the database management systems, analysing the benefits, In the preliminary study database management system elaborated	Word, Zotero
16.10.2023	0:50	Project management	Pre-study presentation further developed, benefit analysis translated, currency changed from $ to KES	Word, Zotero
17.10.2023	0:50	Project management	Preliminary study continued	Word, Zotero
17.10.2023	0:20	Project management	Discussion with Professor Schwarz and team	Word
17.10.2023	3:05	Project management	Preliminary study continued, preliminary study presentation continued, Time recording repaired	Word, Excel
19.10.2023	2:40	School	Work on the preliminary study presentation and preliminary study	Word, PPT
19.10.2023	0:35	Project management	Preliminary study Solutions Variants for databases finalised.	Word, Zotero
21.10.2023	0:40	Preliminary project phase	Pre-study presentation corrected with PTM	Word, Zotero
22.10.2023	1:05	Preliminary project phase	Preliminary study finalised	Word, Zotero
23.10.2023	1:20	Preliminary project phase	Pre-study presentation 2.0 created --> new utility value analysis for equality	Word, Excel, Zotero
25.10.2023	0:50	Analyse	Requirement Specifications: 2.1angefangen	Word, Zotero

27.10.2023	1:00	Analyse	Requirement Specifications: 2.1weitergeführt	Word, Zotero
02.11.2023	1:00	Analyse	Requirement Specifications: 2.16	Word, Zotero
06.11.2023	1:00	Analyse	Requirement Specifications: 2.1: 2.1	Word, Zotero
06.11.2023	1:40	School	Requirement Specifications: 5 + PHB	Word, Zotero
06.11.2023	1:05	Analyse	Requirement Specification Inserted Use Case	Word, Zotero
10.11.2023	0:20	Implementation	Boris added as a product, order table extended in admin	Visual Code
13.11.2023	2:15	Project management	Project Manual Completed	Word
19.11.2023	0:45	Implementation	Specifications 3.1.1, 3.1.2	Word
20.11.2023	1:40	Implementation	Specifications: 3.4.1, 3.4.2	Word
20.11.2023	3:15	Implementation	Specifications 3.4.1, 3.4.2, 3.4.1, 3.4.1.1, 3.4.1.2, 3.4.1.3, 3.4.1.4, 3.4.1.5, 3.4.2, 3.4.3 added and bugs corrected	Word
23.11.2023	1:40	Implementation	Subdomain created (admin.eroots.store) for admin dashboard and published to Vercel (according to Arbeitsbericht_23_11_2023_Njeri)	Visual Code, Ionos, Vercel
28.11.2023	0:30	Project management	Meeting with Professor Schwarz	Word
28.11.2023	3:05	Implementation	Improve parts of code and error messages in the code. Cart pop-up for message that all data needs to be entered.	Visual Code
29.11.2023	1:50	Implementation	Correcting the functional specification	Word
30.11.2023	1:40	Implementation	Project work	Visual Code
06.12.2023	1:50	School	Status report as of 06.12.2023 in the classroom	Word
11.12.2023	0:05	Implementation	merge Pull Request	GitHub
12.12.2023	3:40	Implementation	worked on the branch categories and try to filter by categories	Visual Code
13.12.2023	2:45	Implementation	Categories completed and filter function added	Visual Code

13.12.2023	0:35	School	Categorie branch commits everything and creates a pull request	Visual Code
15.12.2023	00:40	Implementation	Settings branch, tries to create the setting for themes on the Settings page	Visual Code
19.12.2023	1:20	Implementation	Errors in the themes are not made via the whole admin interface	Visual Code
29.12.2023	0:30	Project management	Project team meeting	Teams
06.01.2024	2:25	Implementation	Categorie page adjusted to fit the screen	Visual Code
07.01.2024	1:30	Implementation	Tries to correct Google Login	Visual Code
07.01.2024	1:35	Implementation	Went through the Nextauth website and made the change according to the instructions Login via Localhost:3000 and try also via the domain	Visual Code
07.01.2024	0:35	Implementation	Oauth works in the auth branch and has now been pulled to the main branch	Visual Code
07.01.2024	01:30	Project management	Project meeting via Admin Login + Other	Teams, Word
08.01.2024	02:15	Implementation	User Login AUTH Branch	Visual Code
21.01.2024	0:55	Project management	Project Status Report 2 improved	Word
24.01.2024	01:50	Implementation	Dropped whole auth changes and started view orders from scratch	Visual Code
25.01.2024	03:20	Implementation	Dropped whole auth changes and started view orders from scratch	Visual Code
26.01.2024	02:50	Implementation	The user_name and user_email associated with the order	Visual Code
29.01.2024	03:40	Implementation	Try to save the User_id to the order as well	Visual Code
30.01.2024	02:40	Implementation	Implement History View	Visual Code
07.02.2024	0:55	Implementation	Diploma project worked and redeployed	Visual Code
07.02.2024	1:15	Thesis	Unstructured Data, 1.1.1	Word, Zotero

08.02.2024	01:10	Implementation	Error on a pull request handled Settings Branch Adopted	Visual Code
09.02.2024	02:35	Implementation	Userorders error continues, but under development it does not work in deployment	Visual Code
09.02.2024	01:30	Thesis	Management summary, project goals added, errors in photos fixed, 1.3.1 Relational databases, 1.3 Database Models, 1.2.1 Characteristics of a database´, 1.2 What is a Database?	Word, Zotero
11.02.2024	03:05	Implementation	Userorders error continues, but under development it doesn't work in deployment	Visual Code
12.02.2024	02:30	Project management	Test cases defined and reviewed with Sarah Paukner	Word, Teams
13.02.2024	00:20	Project management	New Test Cases Added	Word
14.02.2024	2:20	Implementation	User History View tries to correct in deployment	Visual Code
15.02.2024	3:45	Thesis	1.6.1 OAuth, 1.6 WHAT KIND OF AUTHENTICATION METHODS COULD BE USED, 1.5 What is Authentication? 1.4.2 NoSQL database management systems, 1.4.1 Relational databases management system, 1.4 Database Management Systems, 1.3.2 Object oriented databases	Word, Zotero
16.02.2024	3:15	Thesis	1.7.3 Mongoose, 1.7.2 How does MongoDB work with Next.js? 1.7.1 MongoDB, 1.7 Solution approaches,	Word, Zotero
17.02.2024	1:30	Thesis	Solution approaches: MongoDB, how does MongoDB work with Nextjs	Word, Zotero
18.02.2024	1:15	Implementation	User orders tried to change in deployment	Visual Code
19.02.2024	0:35	Project management	Edit Status Presentation	PPT
20.02.2024	2:40	Implementation	User orders connected via session	Visual Code
21.02.2024	3:50	Implementation	User orders are now also displayed in the deployment	Visual Code

22.02.2024	0:50	Project management	Diploma thesis discussion	WhatsApp
22.02.2024	3:00	Thesis	Authentication and authorization in Next.js with NextAuth.js and MongoDB	Word, Zotero
25.02.2024	03:30	Thesis	Mongoose	Word, Zotero
27.02.2024	04:00	Implementation	Fixed upload photos	Visual Code
28.02.2024	03:50	Thesis	Pros and cons different. More Project-related written	Word, Zotero
01.03.2024	02:30	Thesis	Improving code in text boxes	Word, Zotero
03.03.2024	04:50	Thesis	1.1 Authentication and Security Measures	Word, Zotero
05.03.2024	02:20	Thesis	1.5.3 Access blocked: The request of this app is invalid., 1.5.2 Error while posting a new Product	Word, Zotero
09.03.2024	00:50	Thesis	Errors that occurred, 1.1.1 Email	Word, Zotero
10.03.2024	00:40	Thesis	Lessons Leaned	Word, Zotero
11.03.2024	00:40	Thesis	Correction of quotations	Word, Zotero
Total			182:55	

*Calculated total excludes time spent at school and on the preliminary study

3 Task “Security & User Experience” by Sarah Jessica Paukner

This section of the thesis explores the relationship between security and user experience. It examines the theoretical foundations and tangible implementations of these concepts and their interplay within the specific context of the project.

This examination focuses on analysing the dimensions of security and user experience. It aims to uncover the mechanisms and methodologies that underpin security measures while enhancing the user experience. The goal is to show how these elements can harmonise to create a cohesive digital environment.

In addition, this exploration goes beyond theory and includes practical applications to show how integrating security and user experience considerations can have tangible effects. By examining good practices and mistakes, this section aims to provide practical insights and best practices that can help develop and improve digital systems.

3.1 Security Fundamentals

Securing a web application is critical to the project sponsor because it ensures a secure and trusted connection to the application, authenticates their identity, and protects the data of future users. Therefore, this project prioritises implementing comprehensive security measures, such as encryption, user authentication, and authorization controls, to protect sensitive data and address vulnerabilities.

3.1.1 Encryption

Encryption is important for sensitive data, like user passwords. However, in the case of the ERoots web application, there is no need for separate or self-implemented encryption. This project uses Google OAuth, which allows users to log in using their Google account, meaning that their passwords are not stored in a database or on the internet. Therefore, the implementation of encryption is no longer a requirement.

3.1.2 User Authentication

Users can log in to the web application using their Google Account without the need for a password, making it easier for them to create and access their accounts on the store's website. This method is convenient as it uses an already verified email address.

3.1.3 Domain Safety

Domain security is critical to ensuring the security of customer and user data. Therefore, in order to encrypt the transmission of data between the web browser and the server and to confirm the identity of the website, it is necessary to obtain an SSL certificate from the domain provider. However, an SSL certificate accepted by browsers is not permanent and must be renewed every four months, which is done automatically by the domain provider.

3.2 User Experience Fundamentals

Given the intended use of the web application by the Kenyan population, it is crucial to ensure that it is responsive and functional on both smartphones and desktops.

3.2.1 Responsive Design

A responsive design is crucial for this web application as it is intended to be viewed on both smartphones and desktops. Therefore, a responsive layout must be implemented. This ensures that the application is clearly structured on both devices. The use of design tools such as Tailwind CSS and Bootstrap can achieve this.

3.2.2 E-Commerce Layout

As the web application is intended for the online sale of agricultural products, it uses a layout commonly found in e-commerce applications. This ensures that the goods for sale are presented in a clear and concise manner, providing a quick overview of the products available.

3.3 Security Approaches

How should one provide security for a web application?

3.3.1 HTTP Security

To ensure a secure web application, it is essential to use a secure HTTP protocol, which can be achieved by converting it to HTTPS (Hypertext Transfer Protocol Secure). This is especially crucial when users transmit sensitive data, such as when logging into the web application.

There are two options for implementing a HTTPS protocol into a web application: self-implementation or third-party options.

3.3.2 Self-Implementation

One option for implementing a HTTPS service is self-implementation. To do so, the necessary service must first be defined: (cf. Cherian, 2018; cf. MaxSped, 2016)

3.3.2.1 High-Level Scala API

A High-Level Scala API is a more generic and relatively simpler option, providing more functionality within one command statement than a Lower-Level API. A High-Level API in Scala may include domain-specific operations, custom views and structures, and often executes at superior speeds. In accordance with the given references.

3.3.2.2 Low-Level Scala API

The Low-Level Scala API provides more detailed control over the manipulation and implementation of functions. This API offers low-level functionality and control, allowing for more direct control over the internals of the language or the JVM. Conversely, while low-level APIs may require more code and be more complex, they also provide greater control and potential for optimization.

3.3.3 Third-Party Options

Another option for securing the web application with HTTPS is to explore third-party providers.

3.3.3.1 SSL-Certificates

Many third-party options offer the use of SSL certificates to secure the domain.

SSL (Secure Sockets Layer) certificates are encryption protocols in the TCP/IP protocol stack. They serve as a proof of identity and contain information that browsers and servers use to establish encryption. The certificate is a file that contains information such as the issuer's name, serial number, and fingerprint for encryption. Website owners must contact a certificate authority to obtain a certificate. These organizations are authorized to issue SSL certificates, but they typically charge a fee for their service. (cf. IONOS, 2021)

3.4 User Experience Approaches

To ensure a good user experience, it is important to carefully select the libraries and frameworks used during web application implementation.

3.4.1 Next.js

Next.js is an open-source web development framework created by the private company Vercel providing React-based web applications with server-side rendering and static website generation.

React documentation mentions Next.js among "Recommended Toolchains" advising it to developers when "building a server-rendered website with Node.js". Where traditional React apps can only render their content in the client-side browser, Next.js extends this functionality to include applications rendered on the server-side.

The copyright and trademarks for Next.js are owned by Vercel, which also maintains and leads its open-source development. (cf. 'Next.Js', 2024)

Next.js provides various styling options for web applications, including Global CSS and CSS Modules. It is important to carefully consider the trade-offs between these options when choosing a styling approach for a Next.js application.

1. **Global CSS**: is simple to use and familiar to those experienced with traditional CSS, but it can result in larger CSS bundles and difficulties managing styles as the application grows. (Next.js, 2024a)
2. **CSS Modules**: offer a more modular approach to styling, allowing for better organization and encapsulation of styles. To improve maintainability and avoid naming conflicts, it is recommended to create locally scoped CSS classes.
3. **Tailwind CSS**: is a utility-first CSS framework that enables the rapid creation of custom designs by composing utility classes.
4. **Sass**: can also be used for more advanced styling options. A widely used CSS preprocessor that expands CSS with features such as variables, nested rules, and mixins.
5. **CSS-in-JS**: allows for the embedding of CSS directly into JavaScript components, enabling dynamic and scoped styling.

3.4.2 TailwindCSS

User experience depends on the layout of the web application and its navigability. TailwindCSS is a framework that can assist with this.

TailwindCSS offers a range of resources, including components, templates, screencasts, and a playground, to facilitate the creation of customised web applications. (cf. TailwindCSS.com, 2020)

3.4.3 Bootstrap

Bootstrap is a frontend framework that enhances user experience and layout of web applications. It can be customized in various ways depending on the project and complexity of the build tools used. (cf. Bootstrap, 2022)

3.5 Security Results

In order to provide a secure web application for the client, the project uses an SSL Certificate that was provided without cost by the domain provider IONOS.

3.5.1 IONOS SSL Certificate

As the project uses a domain provided by IONOS, 'eroots.store', the web application utilizes a free SSL certificate provided by the provider.

To activate the SSL certificate, the first step is to interact with the 'Domains & SSL' option.

Figure 3-1: Ionos website for domain management.

After the interaction, the website redirects you to the purchased domains in your account, such as 'eroots.store'. To secure the domain, click on its name to be redirected to the domain's settings.

Figure 3-2: The domain management.

Within the domain settings, various options are available, including status and SSL certificate. To activate the free SSL certificate, select the 'Enable SSL encryption' option.

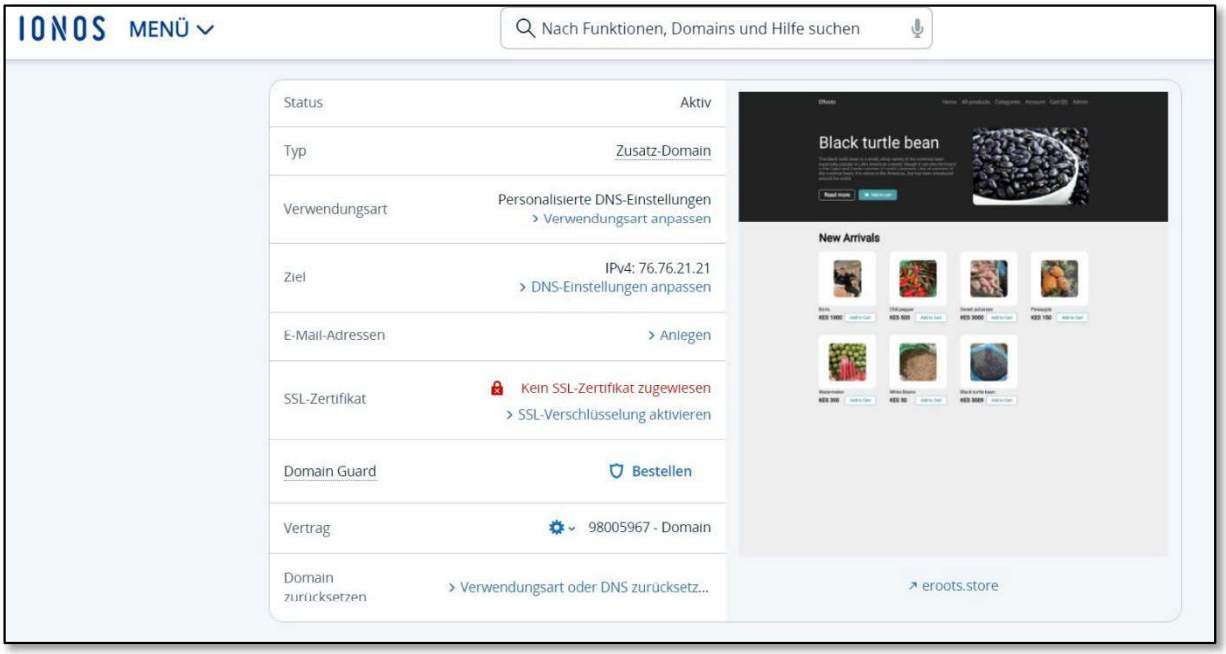


Figure 3-3: How the redirection to the domain looks like.

Once you have selected the site, you will be redirected to different certificate options. One of these options is free, while the other two may cost more. For this project, the free version is sufficient to secure the domain and the web application.

Figure 3-4: How to activate the SSL certificate in Ionos.

After selecting the appropriate SSL certificate version, you will need to configure the certificate for the domain by providing the domain name and intended purpose. Once completed, click the 'Set up SSL certificate' button.

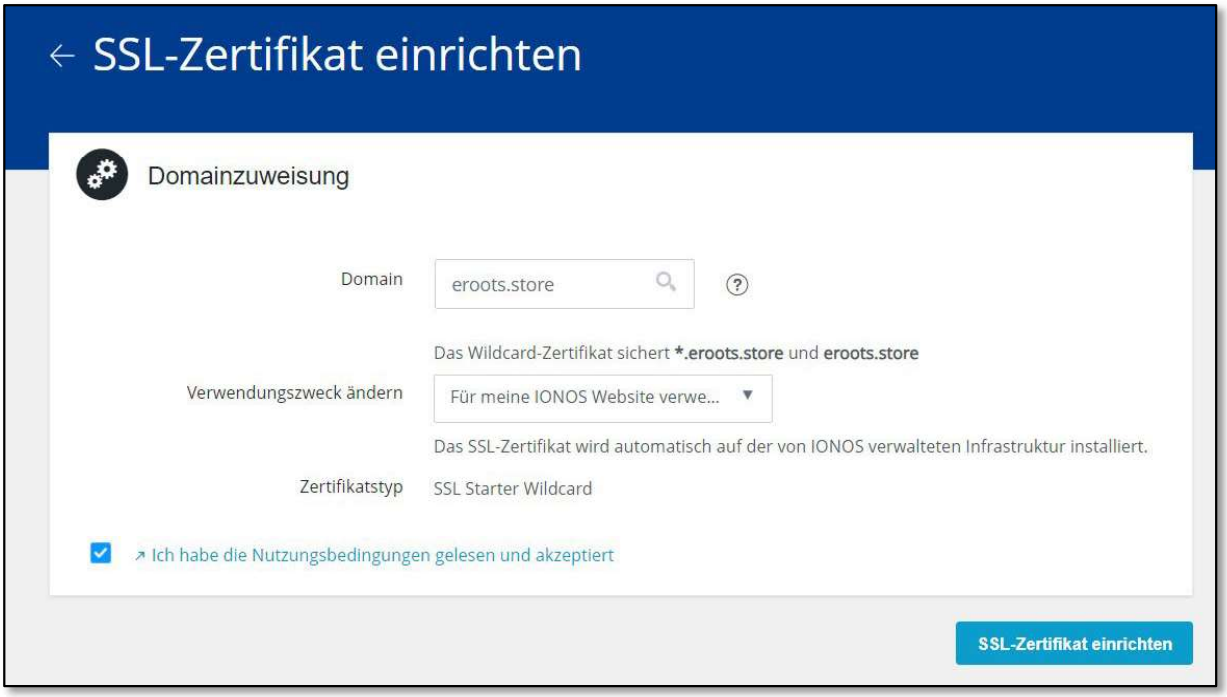


Figure 3-5: The SSL certificate assigned to the domain.

After completing the process, you will receive a notification confirming the successful changes. When you return to manage your SSL certificates, you can verify the certificate type, validation date, and its activation status.

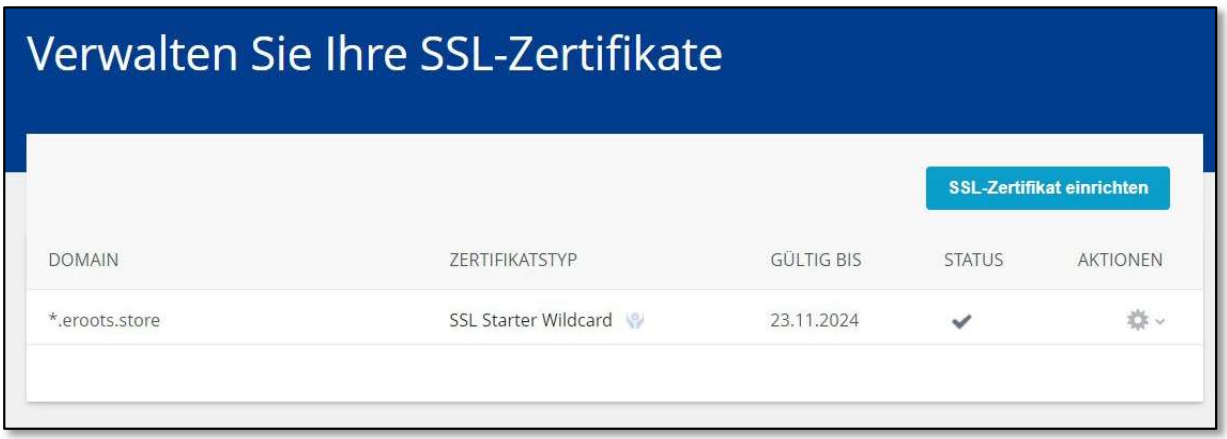


Figure 3-6: Completion of activation of the SSL certificate.

At the beginning, you will notice that the lock, which was previously red indicating an unsafe domain, has now turned blue, indicating that the domain is secured with an SSL certificate.

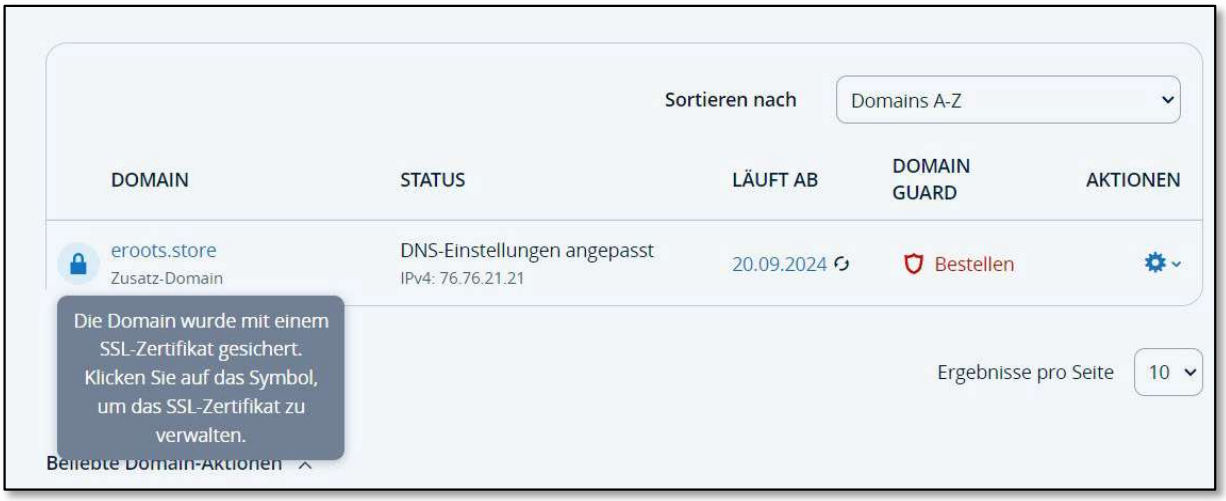


Figure 3-7: After completion the website is now secured.

3.6 User Experience Results

The user experience of web applications is enhanced through the use of TailwindCSS and Bootstrap.

3.6.1 Admin Layout

The administration page features a light blue background and a navigation bar located on the left-hand side of the browser. This provides a brief summary of the various administrative options available, including managing products, categories, and orders within the web application. In addition to the graph displaying the order history for the upcoming days.

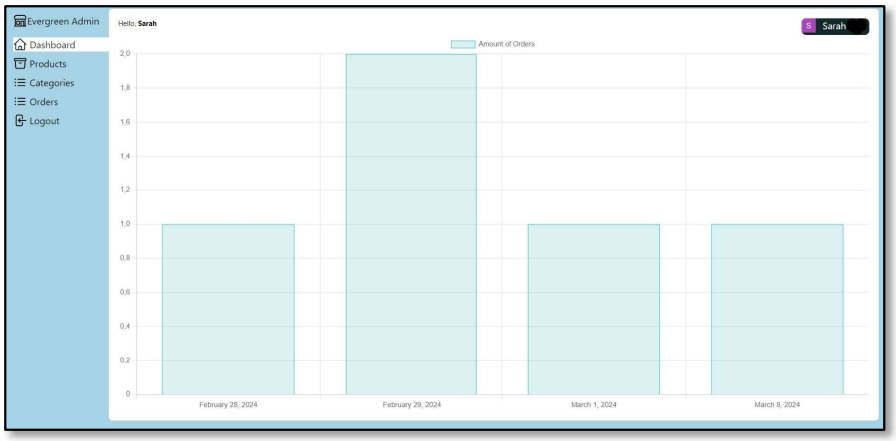


Figure 3-8: Admin dashboard.

3.6.1.1 Navigation Bar

To distinguish between the User Experience of the admin dashboard and the user interface, the navigation bar is positioned on the left side of the screen, providing a clear and structured overview of the available management options. In short, the position and animation of the navigation bar provides a better understanding of the management options and the administrator's current workspace.

3.6.1.2 Graph

The graph on the admin dashboard displays the number of daily reserved orders, providing the client with a better overview and understanding of sales and consumer reach. Its purpose is to aid in understanding the market and demand.

3.6.2 End User Layout

The end user page includes a 'featured product' that is highlighted by a black background, distinguishing it from the other products. The project's name is displayed on the left of the navigation bar at the top of the screen, with the search bar located between the navigation links to the Home-Page, All Products page, Categories page, Account page, and Cart.

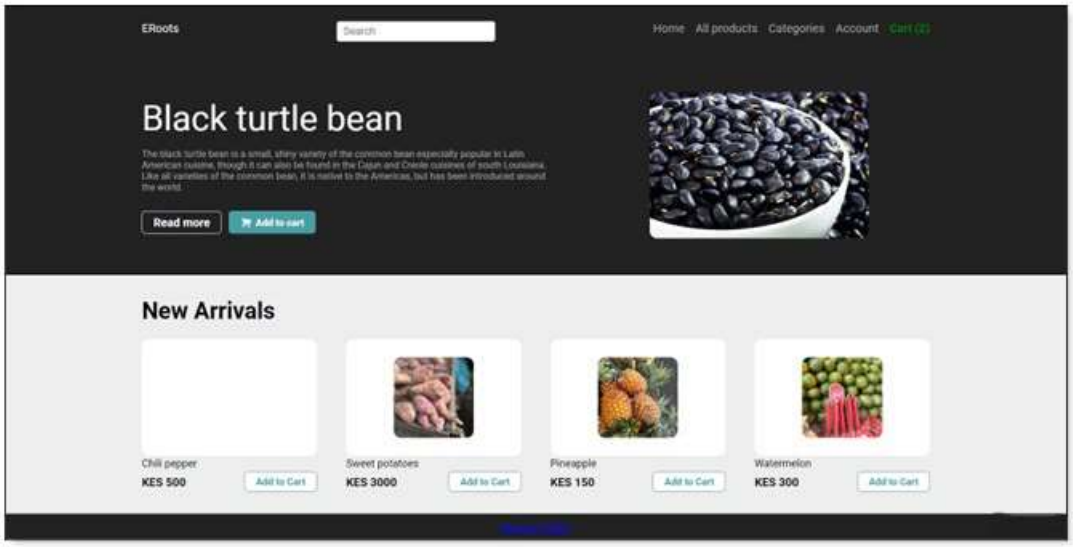


Figure 3-9: Client dashboard.

3.6.2.1 Navigation Bar

To differentiate between the User Experience of the admin dashboard and the user interface, the navigation bar is positioned at the top of the screen. This provides a wider feeling to the web application and allows for more options for the end user to select. As a result, the user is provided with a search bar and navigation links to various pages.

3.6.2.2 Featured Product Header

The purpose of the Featured Product on the homepage is to draw attention to a specific product that the project sponsor wants to sell. To achieve this, the Featured Product is the largest component on the website and extends the navigation bar in colour to give the page a well-rounded appearance.

3.6.2.3 Components

Following the Featured Product, the remaining products available for purchase are presented in separate containers to differentiate between them. Each product is accompanied by a photo and relevant information, including its name and price.

3.6.2.4 Footer

At the bottom of the page, you will find the footer which includes the privacy policy. This is an essential part of the web application's security.

3.7 Web Page „Categories“ Results

The web application enables the project sponsor to categorise their products. This is a feature that was carefully considered before it was implemented.

3.7.1 Use Case

As with any other project feature, a use case must be created to define the functionality and identify any knowledge required in relation to other functionalities.

Use Case 3	
Name	Manage Categories
Short description	Manage the product categories in the system.
Actors	Admin
Trigger	Admin’s desire to add, modify, or delete product categories.
Result(s)	The product categories are effectively managed in the system.
Preconditions	Admin is logged in and has access to the admin dashboard.
Incoming data	Category name, description, and optional parent category information.
Postconditions	The product categories are added, modified, or deleted in the database.
Process description (essential action steps)	• Admin selects the "Manage Product Categories" option. • The system displays a list of existing product categories. • Admin can add a new category or modify/delete existing categories. • The system validates the input and updates the database accordingly.
Error behaviour	Administrator attempts to delete a category with associated products.
Variations	Administrator assigns parent categories to subcategories

3.7.2 Create, Delete Or Edit Categories With Data Persistence

Once the functionality has been established, the next step is to implement it. In order for the admin to be able to create, save and delete a category, the functionality needs to be implemented in the categories.js file.

```
async function saveCategory(ev){
  ev.preventDefault();
  const data = {
    name,
    parentCategory,
    properties:properties.map(p => ({
      name:p.name,
      values:p.values.split(',')})),
  };
  if (editedCategory){
    data._id = editedCategory._id;
    await axios.put('/api/categories', data);
    setEditedCategory(null);
  } else {
    await axios.post('/api/categories', data);
  }
  setName('');
  setParentCategory('');
  setProperties([]);
  fetchCategories();}
async function editCategory(category){
  setEditedCategory(category);
  setName(category.name);
  setParentCategory(category.parent?._id);
  setProperties(category.properties.map(({name,values}) => ({
    name,
    values:values.join(',') }))
  ));
}
```

Figure 3-10: JavaScript Functions for Managing Categories

```
function deleteCategory(category){
    swal.fire({
      title: 'Are you sure?',
      text: `Do you want to delete ${category.name}?`,
      showCancelButton: true,
      cancelButtonText: 'Cancel',
      confirmButtonText: 'Yes, Delete!',
      confirmButtonColor: '#499FA4',
      reverseButtons: true,
    }).then(async result => {
      if(result.isConfirmed){
        const {_id} = category;
        await axios.delete('/api/categories?_id=' + _id);
        fetchCategories();}});}
  function addProperty() {
    setProperties(prev => {
      return [...prev, {name:'',values:''}]});}
  function handlePropertyNameChange(index,property,newName) {
    setProperties(prev => {
      const properties = [...prev];
      properties[index].name = newName;
      return properties;});}
  function handlePropertyValuesChange(index,property,newValues) {
    setProperties(prev => {
      const properties = [...prev];
      properties[index].values = newValues;
      return properties;}); }
  function removeProperty(indexToRemove) {
    setProperties(prev => {
      return [...prev].filter((p,pIndex) => {
        return pIndex !== indexToRemove;});
    });
```

Figure 3-11: JavaScript Functions for Managing Categories

3.7.3 Admin Layout

The admin can manage the categories to which the products belong through the implemented user interface.

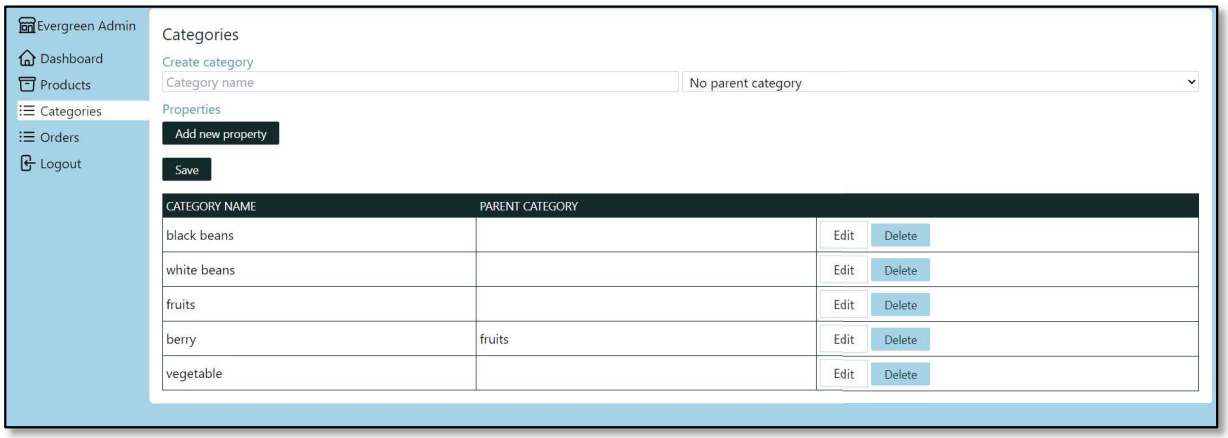


Figure 3-12: Admin category dashboard.

3.8 Web Page „Featured Product“ Results

The 'Featured Product' functionality was implemented later in the process to enhance the user experience. As a result, the web application now highlights a specific product, which can be changed according to the project sponsor's preferences.

3.8.1 Implementation

This part of the web application's layout was implemented in the Featured.js file as a component for the homepage.

```
<Bg>
        <Center>
          <ColumnsWrapper>
            <Column>
              <div>
                <Title>{product.titel}</Title>
                <Desc>{product.description}</Desc>

                <ButtonsWrapper>
                  <ButtonLink href={'/product/'+product._id} outline={1}
white={1}>Read more</ButtonLink>
                  <Button primary onClick={addFeaturedToCart}>
                  <CartIcon />
                    Add to cart</Button>
                </ButtonsWrapper>

              </div>
            </Column>
            <Column>
              <ImageHolder
src="https://th.bing.com/th/id/R.1ac6fc3384ff400a29dd318d2e48432f?rik=54xvackJQ
9kL4Q&riu=http%3a%2f%2fhealthcautions.com%2fwp-
content%2fuploads%2f2018%2f01%2fblack-beans-
1.jpg&ehk=8orUV7OSqRAI3dLol0VMVbI08k7PCZr%2bxi9J%2bHz389Q%3d&risl=&pid=
ImgRaw&r=0"/>
            </Column>
          </ColumnsWrapper>
        </Center>
      </Bg>
```

Figure 3-13: Product Card Component

3.8.2 Featured Product Layout

This is what the styled and finished implemented component looks like.

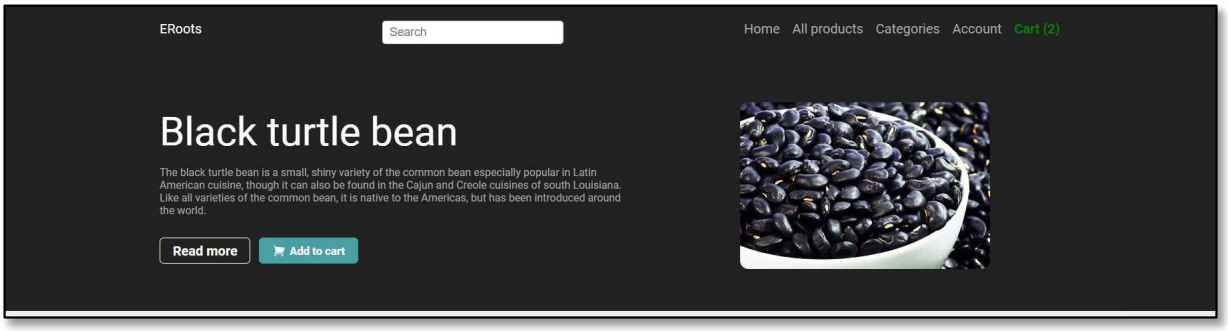


Figure 3-14: Featured Product Layout

3.9 Errors and their Solutions

A project wouldn't be a challenge if there weren't errors that occurred every once in a while. These errors included compile errors where modules were not found and had to be reinstalled.

3.9.1 Failed to compile

Often times an error such as 'Failed to compile' occurred. In this instance it was an error to compile the module 'react-sortablejs' as it was not found at the time were the project wanted to run.

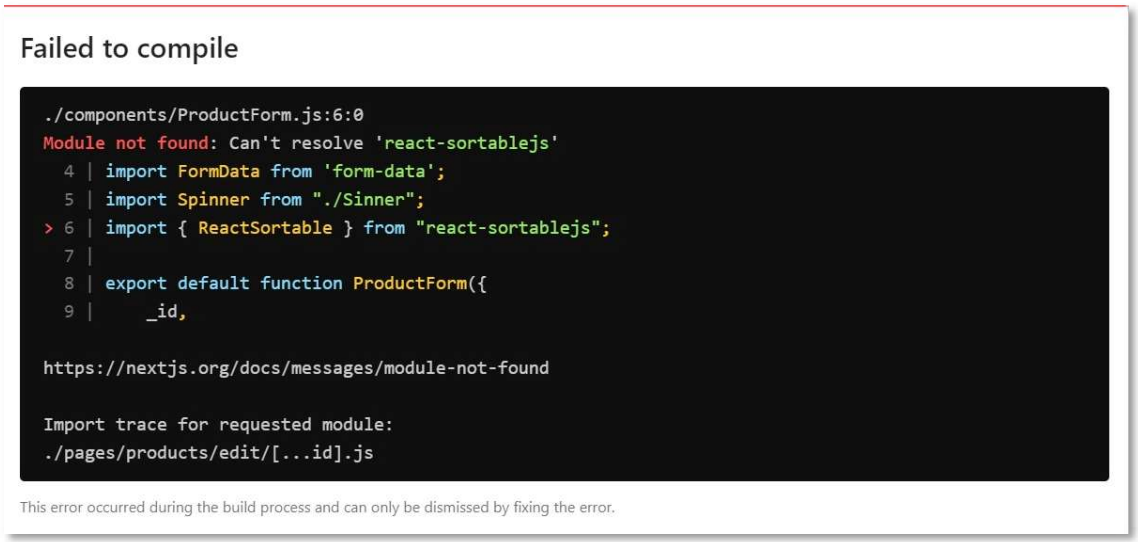


Figure 3-15 Error Message: Failed to Compile

A simple solution for such an error would be to reinstall the missing modules with the right command in the terminal: `npm install react-sortablejs`.

3.9.2 Create Next-App

To separate the admin dashboard from the client interface, two new applications had to be generated. However, an error occurred during the creation of the application and the application couldn't be created.

```
PS C:\Diplomarbeit\eroots-front> npm install --global yarn

changed 1 package in 326ms
PS C:\Diplomarbeit\eroots-front> yarn --version
yarn : Die Datei "C:\Users\sjpau\AppData\Roaming\npm\yarn.ps1" kann nicht geladen werden, da die Ausführung von
Skripts auf diesem System deaktiviert ist. Weitere Informationen finden Sie unter "about_Execution_Policies"
(https:/go.microsoft.com/fwlink/?LinkID=135170).
In Zeile:1 Zeichen:1
+ yarn --version
+ ~~~~
    + CategoryInfo          : Sicherheitsfehler: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Diplomarbeit\eroots-front> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
Set-ExecutionPolicy : Der Zugriff auf den Registrierungsschlüssel
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell" wurde verweigert. Starten Sie zum
Ändern der Ausführungsrichtlinie für den Standardbereich (LocalMachine) Windows PowerShell mit der Option "Als
Administrator ausführen". Führen Sie zum Ändern der Ausführungsrichtlinie für den aktuellen Benutzer
"Set-ExecutionPolicy -Scope CurrentUser" aus.
In Zeile:1 Zeichen:1
+ Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (:) [Set-ExecutionPolicy], UnauthorizedAccessException
    + FullyQualifiedErrorId : System.UnauthorizedAccessException,Microsoft.PowerShell.Commands.SetExecutionPolicyComma
   nd
```

Figure 3-1616 Error Message: Yarn cannot be Installed

To address this issue, the path needed to be redefined prior to developing the application with its required features.

```
PS C:\WINDOWS\system32> cd
PS C:\WINDOWS\system32> set-location C:\Diplomarbeit\eroots-front
```

Figure 3-1717 Define Path for Respository

Now the application can be created using the command in the terminal. Necessary features can be selected during creation to ensure they are already implemented and

included in the application.

```
● PS C:\ERoots-Diplomarbeit\eroots-front> yarn create next-app .
  yarn create v1.22.19
  [1/4] Resolving packages...
  [2/4] Fetching packages...
  [3/4] Linking dependencies...
  [4/4] Building fresh packages...

  success Installed "create-next-app@13.4.10" with binaries:
        - create-next-app
  √ Would you like to use TypeScript? ... No / Yes
  √ Would you like to use ESLint? ... No / Yes
  √ Would you like to use Tailwind CSS? ... No / Yes
  √ Would you like to use `src/` directory? ... No / Yes
  √ Would you like to use App Router? (recommended) ... No / Yes
  √ Would you like to customize the default import alias? ... No / Yes
  √ What import alias would you like configured? ... pages/*
  Creating a new Next.js app in C:\ERoots-Diplomarbeit\eroots-front.

  Using yarn.

  Initializing project with template: default
```

Figure 3-1818 Create Next-App

After the successful creation of the application, work on implementing the user interface could begin.

```
Done in 23.17s.
Success! Created eroots-front at C:\ERoots-Diplomarbeit\eroots-front
```

Figure 3-1919 Success Message for creating 'eroots-front'

3.9.3 Server Error

This server error often occurs when the project is reinstalled from GitHub or when the .env file is deleted from the directory. The error is caused by the project's inability to

connect to the mongoose database due to missing security keys.

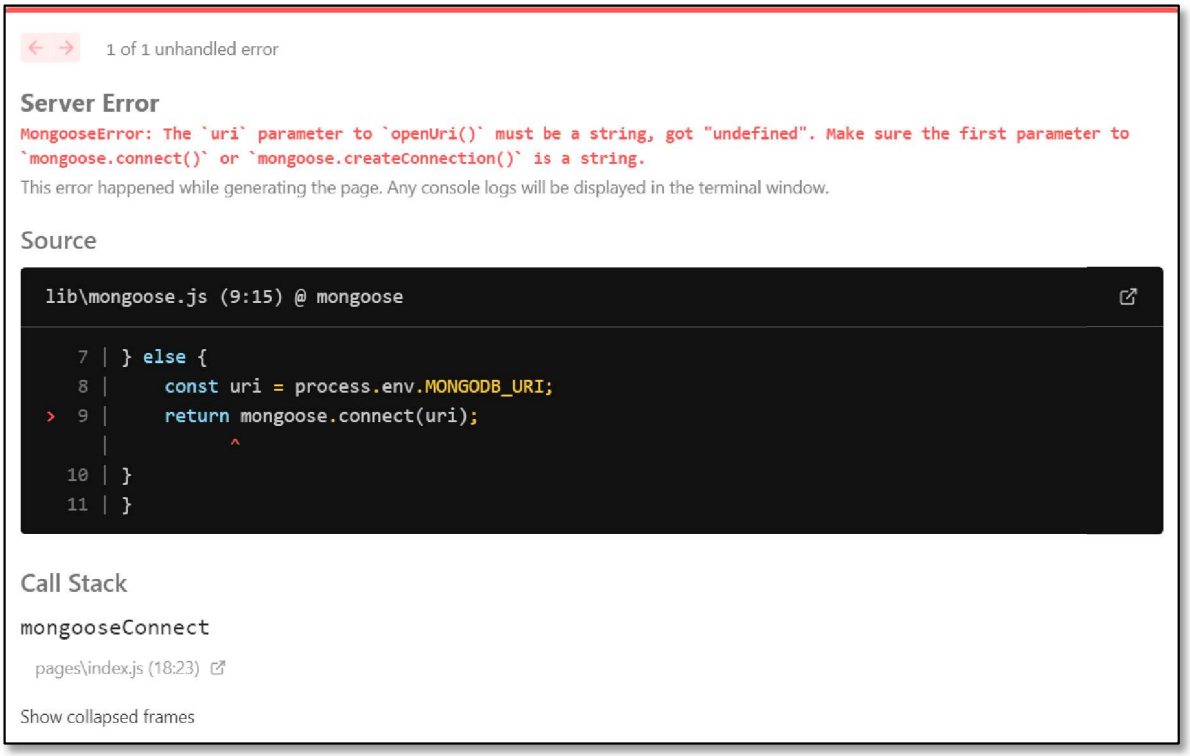


Figure 3-2020 Server Error

To resolve this issue, recreate the .env file in the project and include the security keys for the MongoDB database in the file.

3.10 Lessons Learned

3.10.1 Observation of Stress Escalation:

As the project neared completion, team members experienced a spike in stress levels due to the urgency of completing the web application. This was a stark reminder of the critical role of efficient time management in the project workflow.

3.10.2 Impact on Team Dynamics:

The project team experienced mounting stress levels, which resulted in increased tension. The need to meet deadlines sometimes overshadowed the collaborative effort, negatively affecting how the team bonded and communicated.

3.10.3 Insights on Time Management:

It became clear that starting the project early did not significantly reduce the stress experienced at the culmination of the project, even though the project was started well in advance. This highlights the importance of effective time management throughout the life cycle of a project.

3.10.4 GitHub Integration Issues:

In the later stages of the project, there were persistent issues with pushing and fetching from GitHub. This resulted in numerous merge conflicts. These technical issues consumed valuable time and resources, requiring multiple resets and reimplementations.

3.10.5 Handling Problems at Critical Junctures:

Dealing with GitHub integration issues provided valuable lessons in how to quickly identify and address issues at critical project stages. The team members improved their problem-solving skills and learned how to deal with unforeseen challenges in an agile and calm manner.

3.10.6 Challenges in Motivation:

Valuable lessons on how to quickly identify and resolve critical issues during the project phase were learned from dealing with GitHub integration issues. Team members improved their problem-solving skills and learned to be agile and calm in the face of unforeseen challenges.

3.10.7 Realization of Project Scope:

The scope of the project and the practical limitations of implementing the features led to a significant realisation. Efforts were made to include all the desired features. However, time constraints and technical feasibility forced difficult decisions. Attempting to include dark mode proved to be too difficult and time consuming, and it was left out of the final project due to persistent bugs and incomplete functionality.

3.11 Accompanying protocol according to § 9 para. 2 PrO

Protocol by Sarah Jessica Paukner with date, type of activity (in the case of a meeting, reference to meeting minutes if necessary), effort in hours (to the nearest 5 minutes) and aids used.

DATE	EFFORT	PHASE	ACTIVITY	AIDS USED
24.05.2023	7:35	Preliminary project phase	Folder created, functional specification, requirement specification, PHB, project application, project completion protocol created, Icons created, project name searched, project abbreviation, Project proposal translated into English and the layout made	Canva, MO365
25.05.2023	1:40	Preliminary project phase	Project application started to be filled out with the supervising teacher, thought about the structure and distribution of tasks and external server to host the website (Netlify, Railway.app)	MO365
31.05.2023	5:35	Preliminary project phase	Project meeting 1: What kind of tool do we use for the project? Project meeting 2: Who does what? What do we do/need anyway?	MO365
01.06.2023	1:40	Preliminary project phase	Entered the project request 'Individual tasks' and more, downloaded the program from Github and it didn't work because npm wasn't recognized. Node.js downloaded and ran the npm install command and added path to environment variables. Logged into the program with my email, worked, and added new product, also worked.	GitHub
05.06.2023	1:15	Preliminary project phase	Market Analysis, Checklist, Requirements/Questions PAG	MO365
12.06.2023	1:55	Preliminary project phase	Preliminary study created, started and received documents in English from Prof. Bierbamer	MO365

15.06.2023	1:50	Preliminary project phase	Project application online form submitted with Professor Schwarz and declaration signed	
07.07.2023	2:35	Project management	Project proposal completed with Everlyn	MO365
09.07.2023	0:45	Project management	Short meeting with Everlyn about saving the Product Images	WhatsApp
10.07.2023	3:50	Implementation	AWS Account Created, Bucket Created, User Created, Policy Created, Access Key Created and Pasted to .env, Log Created (Eroots AWS Protocol v1)	AWS, MO365
13.07.2023	3:00	Implementation	View API/.upload Internal Server Error; Solved: At Multiparty. Form() was missing the new before	
13.07.2023	0:35	Project management	Short meeting with Everlyn about more features and changes in the program	WhatsApp
14.07.2023	6:25	Implementation	Categories pages created and encoded: pages/categories.js, pages/api/categories.js, models/category.js + protocol created --> When trying to save the category, I get an Unhandled Runtime Error AxiosError: Request failed with status code 500; seems to have fixed itself again. Table for fetching the stored data from the database. Edit and Delete buttons created; Edit finished encoded. Delete Categories Finished Coded --> Categories Page finished; inserted into Products Page Categories Page and Product Page completed; Edit, Delete, Save.	
14.07.2023	0:35	Project management	Discussion with Everlyn about product data options: Purchase volume in grams (kilograms)	WhatsApp
15.07.2023	4:05	Implementation	add properties in categories for products (gram count), handlePropertyNameChange + ValueChange Finishing Adding and Deleting Properties Show and	

			edit/save ParentCategories on product page --> admin - categories management finished	
16.07.2023	3:15	Implementation	admin - security: Function to authenticate stored email addresses encoded and stored in /api/... inserted; Email addresses set by the admins, error message made --> done; Project team members notified of progress	
17.07.2023	6:50	Implementation	making admin mobile responsive: function to display navigation button, adjust navigation on large vs medium displays; Fake products inserted, change layout/design a bit -- > done. e-commerce front - featured product: New project created with Yarn; Error messages and didn't work for a long time --> fixed, ? Missing/Faulty Directories Created?	
18.07.2023	6:05	Implementation	Created a new repository for the frontend of the project, created next-app and copied missing folders from the old one --> didn't work, so I deleted everything again and created a new next-app --> the log says how it went right now. Frontend Design: Header, Featured, Custom Styling Created Mongoose Database Connection Created --> featured product was ready . . . until it was deleted because of add GitHub repository --> restore files and add team members to the repository	MO365
19.07.2023	0:40	Project management	Meeting: PTM informed that work can continue.	WhatsApp
20.07.2023	0:45	Project management	Meeting with Everlyn about the progress of the project	WhatsApp
22.08.2023	1:15	Project management	Meeting with team about progress in the project	WhatsApp
26.09.2023	0:55	Project managament	Instead of publishing to AWS Amplify, I discussed publishing to Vercel with Everlyn. System architecture improved	MO365

			instead of AWS Amplify deployment on Vercel. New system architecture added to preliminary study and description changed	
26.09.2023	0:55	Preliminary project phase	Security Options for ReactJS Researched	Edge
03.10.2023	1:45	Preliminary project phase	Discussion with Professor Schwarz about the citation method for the diploma thesis	Zotero
17.10.2023	0:45	Project management	Meeting with Professor Schwarz and team	
21.10.2023	1:50	Preliminary project phase	Preliminary study presentation corrected with PTM, introductory text inserted into Word and handed in	MO365
22.10.2023	3:15	Preliminary project phase	Preliminary study Security completed; Preliminary study completed. Errors that occurred in the preliminary study were corrected and submitted.	MO365
25.10.2023	2:15	Analysis	Requirement Specifications started, 1.1.1, 1.1.4, 1.2.2, 1.2.3, 1.3, 1.3.1, 1.3.2	MO365
02.11.2023	1:00	Analysis	Requirement Specifications 6	MO356
06.11.2023	2:25	Analysis	Requirement Specifications: 1, Summary, 1.6, 1.6.1, 1.6.2, 1.6.3, 4, 5, 8	MO365
07.11.2023	0:45	Project management	PHB: Object Observation Plan	MO365
13.11.2023	3:25	Project management	PHB: 1.7 Object Viewing Plan, 1.8 WBS and 1.9 Project Work-Package Specification, 2.1 Status Report, 1.12 Resource Plan	MO365
18.11.2023	3:40	Implementation	System Specifications: 1.2 (10 min), 2.1 (10 min), 2.2 + 2.2.1 (40 min), 2.2.2 (30 min), 2.2.3 (30 min), 2.3 (40 min), 2.4 (40 min), 2.5 (20 min)	MO365
20.11.2023	7:35	Implementation	System Specifications: 3.3, 5, 5.1, 2.6 (30 min), 2.7 (20 min), 2.8 (20 min), 2.9 (10 min), 2.10 (10 min), 2.11 (20 min), 2.12 (20 min), 1.3 (20 min), 1.4 + 1.5 (40 min), 3.6 (30 min), updating worklog, Read through and correct	MO365

23.11.2023	4:55	Implementation	User login, SSL certificate for domain added --> http to https	MO365
28.11.2023	0:35	Project management	Meeting with Professor Schwarz	
29.11.2023	2:20	Implementation	System specification corrected	MO365
30.11.2023	2:45	Implementation	ERoots-7-Besprechungsprotokoll-30-11-2023-v1, User login	MO365
06.12.2023	3:25	Implementation	User Login	
29.12.2023	0:30	Project management	Project team meeting	
30.12.2023	1:45	Implementation	Oauth User Login Internal Server Error 500, Google Cloud --> Admin works again but only in Developement after verifying that the product is only the trial version	
31.12.2023	2:00	Project management	Thought about how to write the diploma thesis and which overarching topics: Security (incl. Oauth and Google Cloud), User Login, Categorien Management via Admin Interface, etc.	MO365
07.01.2024	2:15	Implementation	User-Login, Project does not run with "npm run dev", laptop restarted, after restart it worked again, Sign in still does not work	
07.01.2024	2:05	Project management	Diploma thesis Word template (V2) downloaded and partially started labeling + project meeting. Project meeting about Admin Login	MO365
08.01.2024	1:55	Implementation	Progress for user login has been lost, so I concentrate on the settings page, changing light/dark mode only works in the settings area directly, changes are saved but not changed	
09.01.2024	1:15	Implementation	Dark/Light Mode tried to install using next-theme (doesn't work) + implementation code Implementation of Dark Mode with CSS and/or Javascript	Edge
10.01.2024	1:45	Project management	Status report as of 10.01.2024	MO365

10.01.2024	2:15	Implementation	Dark Mode implementing	Edge
13.01.2024	5:10	Implementation	Light/Dark Mode	Edge
14.01.2024	3:00	Implementation	Light/Dark Mode - should work, I just can't find where this is overwritten on the website and how to correct it.	Edge
15.01.2024	1:05	Implementation	Tried to find the error - didn't work - back to the status of 14.01.2024	
18.01.2024	3:15	Implementation	Light/Dark Mode - changed the ToggleButton to the design of the button and it now changes the HomePage (specially implemented in the file)	
18.01.2024	1:05	Thesis	Divided up my diploma thesis and wrote template in a Word document	MO365
21.01.2024	2:00	Project management	Project Status Report 2 improved	MO365
24.01.2024	1:30	Implementation	Worked on Light/Dark Mode	
25.01.2024	1:45	Implementation	Light/Dark Mode pushed (for HomePage)	
27.01.2024	3:30	Implementation	Worked on Light/Dark Mode	
29.01.2024	2:15	Implementation	Responsive design is buggy and doesn't work on phone --> just reload (on phone go to other page like: All Products go and it reloads and then works)	
30.01.2024	2:40	Implementation	Implemented Dark Mode in Products Page, Categories Page and Login Page (not logged in) --> works, missing Cart	
07.02.2024	0:55	Implementation	Diploma project worked and zipped what works	
07.02.2024	1:05	Thesis	Writing a diploma thesis	MO365
12.02.2024	2:30	Project management	Test cases defined and gone through with Everlyn Njeri	MO365
13.02.2024	2:10	Project management	Test cases defined in project completion protocol	MO365

15.02.2024	1:50	Implementation	Fix merge conflicts and errors (GitHub)	GitHub
17.02.2024	6:40	Implementation	Dashboards on github deleted and reinserted -- deployment works again. New branch for darkmode and history view created 'darkmode_historyview' and worked on improving darkmode and the implemented history view to find out why it doesn't work in the deployment.	
17.02.2024	0:20	Project management	Status Report for 20.02.2024	MO365
18.02.2024	2:15	Thesis	Writing a diploma thesis	MO365
19.02.2024	0:50	Project management	Status presentation KW8 created/copied from the first state presentation. Status presentation KW8 completed.	MO365
19.02.2024	2:10	Implementation	Dark Mode encoding commented out and tried to create using NextUI, DarkMode works and is now pinned to the footer, the footer also has a distance to the page so that it doesn't cover anything anymore, but the code for cart doesn't work anymore, the images, i.e. the products in the cart are no longer displayed...	
20.02.2024	3:55	Implementation	copied changed classes into a word document, noticed that the main branch was used --> switched to darkmode branch to see what was missing from the new cart. Why does an error occur?	
20.02.2024	2,45	Thesis	Diploma thesis part formatted and divided into headings: Results, Solutions, etc.	MO365
21.02.2024	0:25	Project management	Status presentation corrected	MO365
22.02.2024	2:55	Implementation	Dark Mode on branch DarkMode edited --> error with cart pictures, images on admin fixed in ProductPage	

22.02.2024	0:50	Project management	Diploma thesis discussion	
23.02.2024	3:30	Implementation	DarkMode implemented	
24.02.2024	5:20	Implementation	DarkMode implemented	
25.02.2024	6:15	Thesis	Frameworks and Libraries Fundamentals, Security Approaches, User Experience Approaches, Security Result, User Experience Results, Admin Layout, EndUser Layout, Dark/Light Mode, Web Page 'Categories' Results, Web Page 'Featured Product' Results, references and bibliography, Accompanying Protocol	DeepL Write, ChatGPT, MO365
27.02.2024	0:50	Implementation	Created a new branch after the old one had conflicts all the time when trying to merge the main branch.	
28.02.2024	4:10	Implementation	On main branch tries to start the project, it doesn't work to start the frontend in the terminal - import 'bootstrap/dist/css/bootstrap.min.css'; in _app.js project won't compile. Dark Mode Button no longer covered by Footer - Dark Mode Button changed and tries to change the background with it..., next-themes version changed.	
29.02.2024	2:00	Implementation	DarkMode cleared and Bootstrap error corrected	
05.03.2024	0:20	Project management	Diploma thesis discussed with Schwarz	
07.03.2024	1:35	Thesis	Diploma thesis corrected	MO365
10.03.2024	4:30	Thesis	Security Fundamentals, Encryption, User Authentication and Domain Safety & User Experience Fundamentals, Responsive Design, E-Commerce Layout & Web applicaton Categories Results, UseCse, Create, Delete Or Edit Categories With Data Persistence, Admin Layout & Web Page "Featured Product" Results, Implementation, Featured Product Layout	MO365, DeepL Write
11.03.2024	4:30	Thesis	Lessons Learned	MO365, DeepL Write

22.03.2024	4:15	Thesis	Diploma thesis improved: Errors and their Solutions added, UserExperience Results: The Navigation Bar, The Graph. EndUser Layout: The Navigation Bar, The Featured Product Header, The Components, The Footer. Time tracking.	MO365, DeepL Write
TOTAL	180:20			

* Calculated total excludes time spent at school and on the preliminary study

4 Aufgabenstellung: Wie kann eine Webapplikation implementiert und entwickelt werden, die HTTP-Endpunkte bereitstellt und Ereignisse empfängt von Valerian Braun.

Dieser Abschnitt der Diplomarbeit beschäftigt sich mit der Fragestellung, wie eine Webapplikation implementiert werden kann, die HTTP-Endpunkte bereitstellt und Ereignisse empfängt. Außerdem wird die Entwicklung dieses Systems beschrieben und als integraler Bestandteil der Diplomarbeit implementiert.

Die Analyse konzentriert sich auf die Planung, Umsetzung und Weiterentwicklung einer Webapplikation. Diese soll HTTP-Endpunkte bereitstellen und eingehende Ereignisse effizient verarbeiten. Dabei werden nicht nur technische Aspekte, sondern auch zugrunde liegende Konzepte und Methodologien beleuchtet, die für ein erfolgreiches Entwicklungsprojekt von entscheidender Bedeutung sind.

Ein zentraler Schwerpunkt liegt auf der Untersuchung verschiedener Ansätze, Techniken und bewährter Verfahren. Diese ermöglichen es, eine robuste und skalierbare Webapplikation zu entwerfen und zu realisieren. Durch eine umfassende Analyse der Herausforderungen und Möglichkeiten im Entwicklungsprozess sollen fundierte Erkenntnisse generiert werden. Diese können als Leitfaden für Entwickler dienen.

Eine detaillierte Betrachtung des Entwicklungszyklus soll ein umfassendes Verständnis für die Komplexität und Anforderungen dieses Prozesses vermitteln. Dabei werden die Planungs-, Implementierungs- und Testphasen berücksichtigt. Auch Aspekte wie Skalierbarkeit, Wartbarkeit und Sicherheit des entwickelten Systems werden beachtet, um eine Lösung zu gewährleisten.

4.1 Grundlagen

Um eine Webanwendung zu implementieren, die HTTP-Endpunkte bereitstellt und Ereignisse empfängt, müssen verschiedene Schritte durchgeführt werden. Ebenso muss ein passendes Framework ausgewählt werden. Es ist wichtig, eine Programmiersprache auszuwählen, die alle erforderlichen Tools zur Erstellung einer Webanwendung bietet.

Nachdem eine Sprache ausgewählt wurde, erfolgt die Evaluierung eines geeigneten Frameworks, wie zum Beispiel Spring für Java, Flask für Python oder andere. Um das gewählte Framework optimal nutzen zu können, müssen alle erforderlichen Bibliotheken installiert werden. Bei der Erstellung der Anwendung sollte auf die Einhaltung bewährter Praktiken und die Verwendung geeigneter Architekturmuster geachtet werden. Schließlich muss der HTTP-Endpunkt erstellt werden.

Um Anfragen von Clientanwendungen entgegenzunehmen, müssen HTTP-Endpunkte erstellt werden. Die Anfrageverarbeitung wird implementiert, indem die Anwendung so konfiguriert wird, dass sie eingehende Anfragen verarbeiten kann. Hierbei werden die entsprechenden Endpunkte aufgerufen und die erforderlichen Operationen ausgeführt. Abschließend wird eine Antwort gesendet. Nach der Verarbeitung der Anfrage sendet die Anwendung eine geeignete Antwort an den Client. Die Antwort gibt Auskunft über den Status der Anfrage oder die Ergebnisse der durchgeführten Operationen.

Eine robuste Webanwendung kann durch die Umsetzung dieser Schritte erstellt werden, welche HTTP-Endpunkte bereitstellt und in der Lage ist, Ereignisse von Clientanwendungen zu empfangen und entsprechend zu verarbeiten.

4.1.1 Was ist eine Web-Applikation

Eine Web-Applikation ist eine Softwareanwendung, die über das Internet oder ein privates Netzwerk zugänglich ist und über einen Webbrowser bedient wird. Im Unterschied zu herkömmlichen Desktop-Applikationen, die lokal auf einem Computer installiert sind, läuft eine Web-Applikation auf einem entfernten Server und wird über das Web aufgerufen.

Web-Applikationen können verschiedene Funktionen haben, von einfachen interaktiven Websites bis hin zu komplexen Anwendungen wie Online-Shops, sozialen Netzwerken, E-Mail-Diensten, Projektmanagement-Tools und vielem mehr.

Web-Applikationen bieten zahlreiche Vorteile. Dazu zählen die plattformübergreifende Kompatibilität, da sie über einen Webbrowser auf verschiedenen Geräten zugänglich sind, einfache Aktualisierungen, da Änderungen zentral auf dem Server vorgenommen werden können, und die Möglichkeit, von überall auf der Welt darauf zuzugreifen, solange eine Internetverbindung besteht. (Togs et al., 2023)

4.1.2 Was sind http Endpoints

HTTP-Endpunkte sind spezifische URLs oder Pfade auf einem Webserver, die als Ziele für HTTP-Anfragen dienen. Sie stellen Schnittstellen oder Punkte dar, an denen Clients Anfragen senden können, um Ressourcen oder Dienste von einem Server zu erhalten oder mit ihm zu interagieren. HTTP-Endpunkte können typischerweise Daten abrufen (GET), aktualisieren (POST oder PUT), löschen (DELETE) oder andere vom Server unterstützte Aktionen ausführen.

Sie sind grundlegend für die Entwicklung von RESTful-APIs (Representational State Transfer) und werden oft verwendet, um Daten zwischen verschiedenen Anwendungen, Diensten oder Systemen auszutauschen. Sie stellen eine standardisierte Methode dar, um auf Ressourcen zuzugreifen und mit ihnen zu interagieren. Dabei definieren sie eine klare und eindeutige Schnittstelle, über die Client- und Serveranwendungen kommunizieren können. (*Http-Endpoint*, n.d.) (Baeldung, 2023; Boyce, 2020)

4.2 Lösungsansätze

In diesem Abschnitt werden die verschiedenen Optionen betrachtet, die eine Lösung bieten und dann darüber entschieden, welche am besten für das Projekt geeignet ist.

4.2.1 Javascript/Node/Next.js

JavaScript ist eine Programmiersprache, die in Webseiten verwendet und vom Browser des Besuchers ausgeführt wird. Ursprünglich wurde sie 1995 von Netscape als LiveScript entwickelt und diente hauptsächlich zur Überprüfung von Formulareingaben. Heutzutage wird JavaScript für eine Vielzahl von Zwecken eingesetzt, darunter für den dynamischen Aufbau von Webseiten, die Anpassung an Benutzerinteraktionen, komplexe Steuerabfragen und die Anbindung an Datenbanken.

Node.js ist eine plattformübergreifende JavaScript-Laufzeitumgebung, die für ihre geringe Latenz und hohe Durchsatzleistung bei der Bereitstellung von Webanfragen bekannt ist. Dies wird erreicht, indem ein nicht blockierender Ansatz zur Behandlung von Anfragen verwendet wird, bei dem keine Zeit oder Ressourcen für das Warten auf den Abschluss von Ein-/Ausgabeoperationen verschwendet werden. (Duvall, n.d.)

Next.js ist ein bekanntes Open-Source-Framework zur Entwicklung von Webanwendungen mit React.js. Es bietet eine Vielzahl von Funktionen und Werkzeugen, um die Entwicklung von serverseitig gerenderten (SSR) und/oder statischen React-Anwendungen zu erleichtern. (Next.js, 2024b)

4.2.1.1 Vor- und Nachteile von Javascript

Vorteile	Nachteile
• JavaScript wird im Webbrowser des Benutzers ausgeführt, so dass keine spezielle Serverkonfiguration erforderlich ist. Dadurch ist sie leicht zugänglich und ermöglicht die schnelle Entwicklung interaktiver Webanwendungen. • JavaScript ist im Vergleich zu anderen Programmiersprachen eine relativ einfache Sprache. Sie verwendet eine Syntax, die der von C und Java ähnelt, was die Einstiegshürde senkt. • JavaScript hat eine große Entwicklergemeinde und eine breite Palette an Bibliotheken und Frameworks wie React, Angular und Vue.js, die die Entwicklung beschleunigen und die Funktionalität erweitern. • JavaScript ermöglicht es, Webseiten dynamisch zu gestalten und mit den Nutzern zu interagieren, ohne dass eine erneute Anfrage an den Server erforderlich ist. Dies führt zu einem verbesserten Nutzererlebnis.	• Verschiedene Browser können JavaScript unterschiedlich interpretieren, was zu Inkompatibilitäten führen kann. Entwickler müssen oft spezielle Workarounds implementieren, um sicherzustellen, dass ihre Anwendungen auf verschiedenen Plattformen einheitlich funktionieren. • Die asynchrone Natur von JavaScript kann zu komplexem und schwer lesbarem Code führen, insbesondere wenn viele verschachtelte Rückruffunktionen verwendet werden.

Figure 4-1: Vor- und Nachteile von Javascript

(IONOS, 2023b)

4.2.1.2 Vor- und Nachteile von Node.Js

Vorteile	Nachteile
• Node.js wurde entwickelt, um hochskalierbare Anwendungen zu unterstützen, die eine große Anzahl gleichzeitiger Verbindungen verarbeiten müssen. Durch die Verwendung einer ereignisgesteuerten, nicht-blockierenden I/O-Modell-Architektur kann Node.js viele gleichzeitige Anfragen effizient verarbeiten. • Mit Node.js können Entwickler sowohl auf der Client- als auch auf der Serverseite mit JavaScript arbeiten. Dies erleichtert die Wiederverwendung von Code und fördert eine einheitliche Entwicklererfahrung. • Node.js verfügt über ein umfangreiches Ökosystem von Modulen und Paketen, die über den Node Package Manager (npm) zugänglich sind. Diese Pakete ermöglichen es Entwicklern, schnell auf eine Vielzahl von Funktionen zuzugreifen und diese in ihre Anwendungen zu integrieren. • Node.js hat eine große und aktive Community von Entwicklern, die kontinuierlich an der Verbesserung der Plattform arbeiten, Fehler beheben und Ressourcen für andere Entwickler bereitstellen.	• Obwohl Node.js gut für skalierbare Anwendungen geeignet ist, kann die effektive Skalierung von Node.js-Anwendungen eine gewisse Komplexität mit sich bringen. Entwickler müssen spezielle Techniken wie Load Balancing und Clustering implementieren, um die Performance bei steigenden Benutzerzahlen aufrechtzuerhalten. • Obwohl Node.js eine asynchrone, nicht-blockierende Architektur verwendet, ist es im Wesentlichen single-threaded. Das bedeutet, dass CPU-intensive Operationen das gesamte System beeinträchtigen können, wenn sie nicht richtig gehandhabt werden. • Da Node.js eine relativ neue Plattform ist und JavaScript auf der Serverseite verwendet, können Sicherheitsrisiken wie unzureichende Datenvalidierung und unsichere Abhängigkeiten zu potenziellen Schwachstellen führen, wenn sie nicht sorgfältig gehandhabt werden

Figure 4-2: Vor- und Nachteile von Node.Js

(Oamkuma, 2021)

4.2.1.3 Vor- und Nachteile von Next.js

Vorteile	Nachteile
• Next.js ermöglicht das Server-seitige Rendern von React-Komponenten. Dadurch wird die Website schneller initialisiert und die Suchmaschinenoptimierung verbessert • Next.js bietet eine einfache Konfiguration, die die Einrichtung eines Projekts erleichtert. • Next.js unterstützt automatisches Code-Splitting, wodurch nur benötigter Code geladen wird, wenn er benötigt wird. Dies führt zu einer verbesserten Leistung und Benutzererfahrung. • Next.js bietet native Unterstützung für CSS-in-JS-Lösungen wie styled-components oder emotion, was die Verwaltung von Styles in React-Anwendungen erleichtert.	• Für Entwickler, die neu in der React-Entwicklung sind, kann Next.js eine steile Lernkurve haben, insbesondere wenn sie nicht mit Konzepten wie SSR und Routing vertraut sind. • Bei größeren Anwendungen kann die Komplexität von Next.js zunehmen, insbesondere wenn sie eine Vielzahl von Funktionen wie Datenabfrage, Authentifizierung und Zustandsverwaltung umfassen. • Während SSR Vorteile in Bezug auf SEO und initiale Ladezeit bietet, kann es auch zu einem erhöhten Server-Overhead führen, insbesondere wenn viele Anfragen verarbeitet werden müssen. • Obwohl Next.js viele Funktionen zur Beschleunigung der Entwicklung bereitstellt, kann es auch Einschränkungen geben, die Entwickler davon abhalten, bestimmte benutzerdefinierte Anpassungen vorzunehmen, die außerhalb des Rahmens von Next.js liegen.

Figure 4-3: Vor- und Nachteile von NextJs

(Matt, 2021; Matt & Github, 2024; Next.js, 2024b)

4.2.2 Python

Python ist eine moderne Programmiersprache, die objektorientierte, funktionale und imperative Programmierstile unterstützt. Sie ist ideal für Anfänger aufgrund ihrer Lesbarkeit und Benutzerfreundlichkeit. Python ist in erster Linie eine Skriptsprache, kann jedoch in computerlesbaren Binärcode kompiliert werden. Der Vorteil all dessen ist, dass Sie Programme mit weniger Codezeilen schreiben können als ein entsprechendes C/C++- oder Java-Programm. (Kelly, 2016; Python, 2024)

4.2.2.1 Vor- und Nachteile von Python

Vorteile	Nachteile
• Python zeichnet sich durch eine klare und leicht zulernende Syntax aus, welches die Entwicklung einer Web-Applikation erleichtert, beschleunigt. • Python verfügt über eine breite Palette an Bibliotheken und Frameworks für die Webentwicklung wie Django, Flask und Pyramid. Diese Bibliotheken erleichtern die Entwicklung von Web-Applikationen, da sie viele vorgefertigte Funktionen und Module zur Verfügung stellen. • Python hat eine große und aktive Entwicklergemeinde, die ständig neue Tools, Bibliotheken und Ressourcen für die Webentwicklung zur Verfügung stellt. Dies bedeutet, dass es leicht ist Hilfe zu finden, wenn man auf Probleme stößt. • Python kann für verschiedene Arten von Webanwendungen verwendet werden, von einfachen APIs bis hin zu komplexen Webanwendungen und Microservices. Dies macht es zu einer vielseitigen Option für Entwickler.	• Im Vergleich zu anderen Programmiersprachen ist Python langsamer, was dazu führen kann, dass Zeit für die Entwicklung verloren geht. Obwohl dies bei kleineren Anwendungen nicht auffällig ist, wird es bei größeren Anwendungen spürbar. • Obwohl Python eine sichere und leicht zu erlernende Sprache ist, können Fehler auftreten, die jedoch erst zur Laufzeit bemerkt werden. Daher ist es wichtig, ausreichend Zeit für das Testen aufzuwenden. • Python hat wie andere Programmiersprachen auch eine Lernkurve, die für einige eine Herausforderung darstellen kann.

Figure 4-4: Vor- und Nachteile von Python

(IONOS, 2023a)

4.2.3 Java

Java ist eine Programmiersprache und eine Computing-Plattform, die seit 1995 existiert. Sie wird verwendet, um eine Vielzahl von digitalen Anwendungen zu erstellen. Java ist bekannt für seine Zuverlässigkeit und wird oft in Anwendungen auf Unternehmensebene eingesetzt. (Java, 2024)

4.2.3.1 Vor- und Nachteile von Java

Vorteile	Nachteile
• Java-Anwendungen können auf verschiedenen Betriebssystemen und Plattformen ausgeführt werden, was sie sehr flexibel macht. • Java hat, wie Python, eine große Entwicklergemeinde, die eine Vielzahl von Bibliotheken, Frameworks und Tools zur Verfügung stellt, die die Entwicklung beschleunigen und verbessern können. • Java-Anwendungen können aufgrund ihrer Effizienz und der Verfügbarkeit leistungsfähiger Frameworks wie Spring Boot und Hibernate sehr leistungsfähig und skalierbar sein. • Java bietet eine robuste Sicherheitsarchitektur und zahlreiche Mechanismen zur Vermeidung von Sicherheitslücken und Angriffen. • Die native Unterstützung von Multithreading in Java ermöglicht die Entwicklung hochperformanter Webanwendungen, die viele Anfragen gleichzeitig verarbeiten können.	• Java-Anwendungen haben im Vergleich zu einigen anderen Sprachen und Plattformen einen höheren Speicher- und Ressourcenverbrauch. • Java ist eine umfangreiche Sprache mit vielen Konzepten, was eine steile Lernkurve für Entwickler bedeuten kann, insbesondere für Anfänger. • Der Entwicklungsprozess in Java kann im Vergleich zu einigen anderen Sprachen und Plattformen länger dauern, da Java eine strenge Typisierung und eine umfangreiche Codebasis erfordert. • Frameworkkomplexität: Einige Java-Frameworks, insbesondere die umfangreicheren wie Java EE oder Jakarta EE, können sehr komplex sein und eine gewisse Einarbeitungszeit erfordern.

Figure 4-5: Vor- und Nachteile von Java

(IONOS, 2023a)

4.2.4 Ruby

Ruby on Rails ist ein beliebtes Webanwendungs-Framework, das HTTP-Anfragen und Ereignisse effektiv verarbeiten kann. Es ermöglicht eine einfache Erstellung von Webanwendungen aufgrund der vielen Funktionen, die es ermöglichen, schneller zu arbeiten. Es verfügt über eine saubere Syntax, die es ermöglicht, den Code effektiver zu schreiben und zu verstehen. (Flanagan & Matsumoto, 2008; Ruby, 2023)

4.2.4.1 Vor- und Nachteile von Ruby

Vorteile	Nachteile
• Ruby ermöglicht eine schnelle Entwicklung von Anwendungen aufgrund seiner Syntax und zahlreichen Frameworks wie Ruby on Rails. • Die klare und prägnante Syntax von Ruby fördert die Schreibweise von hochwertigem, gut lesbarem Code, der leicht zu warten und zu erweitern ist.	• Im Vergleich zu einigen anderen Sprachen kann Ruby in Bezug auf die Ausführungsgeschwindigkeit etwas langsamer sein, insbesondere bei rechenintensiven oder ressourcenintensiven Aufgaben. • Für sehr große Anwendungen oder solche, die eine enorme Skalierbarkeit erfordern, kann Ruby manchmal Schwierigkeiten haben, mit dem Wachstum Schritt zu halten, insbesondere im Vergleich zu Sprachen und Technologien, die für ihre Skalierbarkeit bekannt sind. • Für Anfänger kann die Lernkurve von Ruby etwas steil sein, insbesondere wenn sie mit der dynamischen Natur der Sprache und den Konzepten von Ruby on Rails als Framework konfrontiert werden.

Figure 4-6: Vor- und Nachteile von Ruby

4.2.5 PHP

PHP ist eine Programmiersprache, die speziell für die Entwicklung von Webanwendungen konzipiert ist. Sie wird nicht nur für die Verwaltung von Datenbanken verwendet, sondern auch für die Ausführung verschiedener serverseitiger Aufgaben. PHP wird in HTML-Code eingebettet, um dynamische und interaktive Webanwendungen zu erstellen. (Benmoussa et al., 2019; PHP, 2001)

4.2.5.1 Vor- und Nachteile von PHP

Vorteile	Nachteile
• PHP bietet eine breite Palette von Funktionen zur Unterstützung der Interaktion mit verschiedenen Datenbanken, was die Entwicklung von webbasierten Anwendungen erleichtert. • PHP kann nahtlos in verschiedene Webserver integriert werden, was eine einfache Bereitstellung von Webanwendungen ermöglicht. • PHP ist flexibel einsetzbar und eignet sich sowohl für kleine Skripte als auch für komplexe, umfangreiche Webanwendungen.	• PHP hat in der Vergangenheit Sicherheitsprobleme aufgewiesen und erfordert eine sorgfältige Programmierung und Konfiguration, um Sicherheitslücken zu vermeiden. • Die Leistung von PHP kann je nach Serverkonfiguration und Art der Anwendung variieren, was zu Inkonsistenzen führen kann. • Für sehr große und komplexe Anwendungen können andere Sprachen und Frameworks möglicherweise besser geeignet sein als PHP. • PHP hat möglicherweise nicht immer die neuesten Funktionen oder Sprachkonstrukte im Vergleich zu anderen Programmiersprachen, was die Entwicklung moderner Anwendungen einschränken kann.

Figure 4-7: Vor- und Nachteile von PHP

(IONOS, 2023c)

4.2.6 Go (Golang)

Go ist eine Open-Source-Programmiersprache, die von Google im Jahr 2007 entwickelt wurde. Sie zeichnet sich durch ihre Einfachheit, Zuverlässigkeit und Effizienz aus und wird hauptsächlich für den Aufbau von verteilten Systemen, Webanwendungen und Netzwerkservern verwendet. (Oberm & BSc, October, 17)

4.2.6.1 Vor- und Nachteile von Go

Vorteile	Nachteile
• Go wurde mit dem Ziel entworfen, einfach zu sein, was bedeutet, dass Entwickler schnell produktiv sein können und weniger Zeit mit komplexen Syntaxregeln verbringen. • Go wurde für die Erstellung zuverlässiger Software entwickelt, wobei besonderer Wert auf robuste Fehlerbehandlung und einfache Debugging-Möglichkeiten gelegt wurde. • Go ist bekannt für seine Effizienz bei der Ausführung von Code und seiner Fähigkeit, parallele und verteilte Systeme effektiv zu verwalten.	• Obwohl Go auf Einfachheit ausgelegt ist, kann die Umstellung auf Go für Entwickler mit wenig Erfahrung in dieser Sprache eine steilere Lernkurve bedeuten, insbesondere wenn sie mit Konzepten wie der Nebenläufigkeit und dem Umgang mit der Go-Runtime umgehen müssen. • Go ist nicht primär für die Entwicklung von Benutzeroberflächen oder grafischen Benutzeroberflächen (GUIs) gedacht und bietet daher keine native Unterstützung für GUI-Anwendungen. Entwickler müssen auf externe Bibliotheken oder Frameworks zurückgreifen, um GUIs in Go zu erstellen.

Figure 4-8: Vor- und Nachteile von Go

4.2.7 C#

C# ist eine von Microsoft im Jahr 2000 entwickelte Programmiersprache. Sie wurde entwickelt, um Windows-Anwendungen und Webanwendungen auf eine einfachere Weise zu erstellen. (C#, 2023)

4.2.7.1 Vor- und Nachteile von C#

Vorteile	Nachteile
• C# ist eine robuste und leistungsstarke Sprache, die eine effiziente Verarbeitung und Ausführung von Code ermöglicht. • Zudem ist C# typsicher, was bedeutet, dass viele Fehler bereits zur Kompilierzeit erkannt werden. Dadurch wird die Entwicklungszeit verkürzt und die Stabilität der Anwendung erhöht. • C# wird häufig in Verbindung mit dem ASP.NET-Framework genutzt. Dieses Framework stellt umfangreiche Werkzeuge für die Webentwicklung bereit, einschließlich Datenbankzugriff, Authentifizierung und Web-API-Integration. Die Integration in die Microsoft-Plattform ermöglicht die Nutzung von Vorteilen wie Azure Cloud Services und Visual Studio.	• Für Entwickler, die bisher keine Erfahrung mit C# und dem .NET-Framework haben, kann es eine gewisse Lernkurve bedeuten, um sich in die Sprache und die zugehörigen Technologien einzuarbeiten. • Einige der fortschrittlichen Tools und Dienste von Microsoft, die mit C# und ASP.NET verbunden sind, können kostenpflichtig sein, insbesondere wenn man sich für die Nutzung von Azure-Diensten entscheidet. • Obwohl C# eine leistungsstarke Sprache ist, kann die Ausführungsgeschwindigkeit im Vergleich zu anderen Sprachen wie beispielsweise C++ oder Go etwas langsamer sein.

Figure 4-9: Vor- und Nachteile von C#

(Informatik-Verstehen, 2023)

4.2.8 Ergebnis

Nach der Bewertung der verschiedenen Programmiersprachen wurde entschieden, dass JavaScript in Verbindung mit Node/Next.js die geeignetste Lösung für das Projekt ist. JavaScript ist eine weit verbreitete Sprache für die Webentwicklung und verfügt über ein reichhaltiges Ökosystem an Bibliotheken, Frameworks und Tools, die den Entwicklungsprozess vereinfachen. Node.js ist eine serverseitige Laufzeitumgebung für JavaScript, die eine nahtlose Kommunikation zwischen Frontend und Backend ermöglicht und somit die Erstellung effizienter Webanwendungen erleichtert. Next.js ist ein React-Framework, das eine solide Basis für den Bau moderner Webanwendungen bietet. Es verfügt über Funktionen wie serverseitiges Rendern, statische Seitengenerierung und effizientes Routing.

4.3 Ergebnisse

In diesem Abschnitt wird die Umsetzung, sowie auch die Ergebnisse gezeigt, der jeweiligen Ziele des Projektes gezeigt.

4.3.1 Create product page with all and one product

Dieser Abschnitt befasst sich mit der Entwicklung der „All product page“ und der „Single Product Page“

4.3.1.1 All Product Page

```
export async function getServerSideProps(){
  await mongooseConnect();
  const product = await Product.find({}, null, {sort:{'_id':-1}})
  return {props:{
    products: JSON.parse(JSON.stringify(product))
  }}
}
```

Figure 4-10: Funktion zur Verbindung der Datenbank

Diese Funktion beschreibt, wie die Verbindung zur Datenbank hergestellt wird, um alle Produkte zu bekommen.

Next.js stellt diese Funktion bereit, um Daten zur Laufzeit zu laden. Hier wird eine Verbindung zur MongoDB über Mongoose hergestellt und alle Produkte nach dem _id-Feld absteigend sortiert aus der Datenbank abgerufen. Anschließend werden die Produkte als Prop zurückgegeben.

```
export default function ProductsPage({products}){
  const { resolvedTheme } = useTheme();
  const pageStyles = {
   backgroundColor: resolvedTheme === 'light' ? '#eee' : '#333',
   color: resolvedTheme === 'light' ? '#000' : '#fff',
  };
  return(
    <>
    <div style={pageStyles}>
      <Header />
      <Center>
        <Title>All product</Title>
        <ProductsGrid products={products}/>
      </Center>
      <Footer/>
    </div>
    </>
  );
}
```

Figure 4-11 Funktion für ProductsPage

Diese Funktion definiert ProductsPage, welche eine Seite zur Anzeige von Produkten darstellt. Sie erhält ein Prop-Objekt 'products', welches eine Liste von Produkten enthält, die von der Datenbank geholt wird.

Ergebnis:

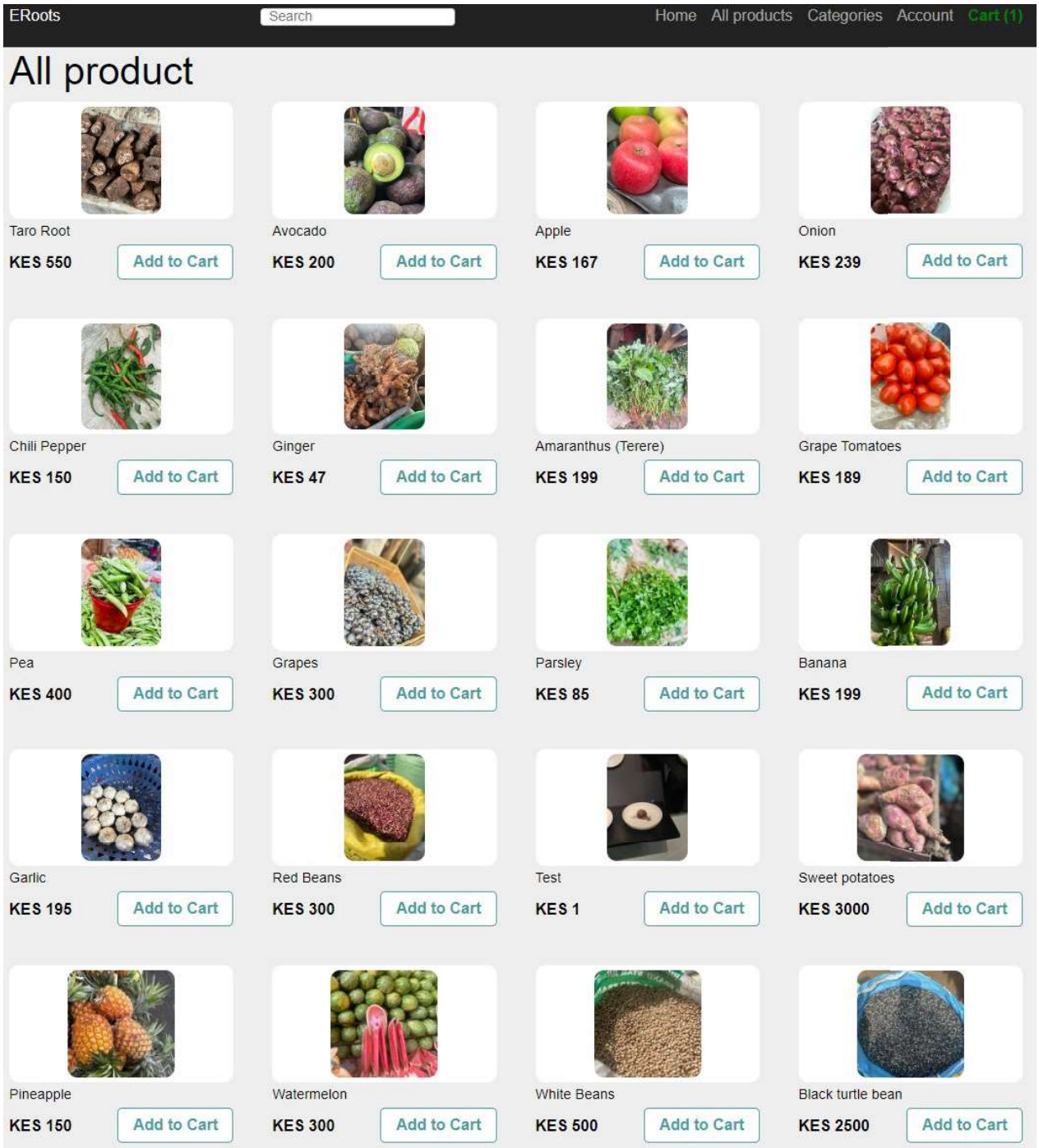


Figure 4-12: Screenshot der All-Product Page

4.3.1.2 Single Product Page

```
export async function getServerSideProps(context) {
  try {
    await mongooseConnect();
    const { id } = context.query;
    const product = await Product.findById(id);
    const allProducts = await Product.find({}, null, { sort: { _id: -1 } });
    return {
      props: {
        allProducts: JSON.parse(JSON.stringify(allProducts)),
        product: JSON.parse(JSON.stringify(product)),
        initialRandomProducts: [],
      },
    };
  } catch (error) {
    console.error("Error in getServerSideProps:", error);
    return {
      props: {
        allProducts: [],
        product: {},
        initialRandomProducts: [],
      },
    };
  }
}
```

Figure 4-13: Funktion zur Verbindung der SingelProductPage

Es wird eine Verbindung zur MongoDB über Mongoose hergestellt, indem mongooseConnect aufgerufen wird. Diese Funktion stellt sicher, dass eine Verbindung zur Datenbank besteht, bevor weitere Operationen durchgeführt werden. Der context.query-Parameter wird verwendet, um die ID des Produkts aus der URL-Abfrage zu extrahieren. Dies wird genutzt, um das spezifische Produkt zu finden, das auf der Seite angezeigt werden soll.

Product.findById(id) wird verwendet, um das gesuchte Produkt aus der Datenbank zu finden. Ebenso werden alle Produkte aus der Datenbank abgerufen und nach dem „_id“

Feld in absteigender Reihenfolge sortiert. Das zurückgegebene Objekt enthält die Props für die Seite. Diese Props umfassen das spezifische Produkt (product), alle Produkt, sowie eine leere Liste für initialRandomProducts.

```
export default function ProductPage({
  product,
  allProducts,
  initialRandomProducts,
}) {
  const { addProduct } = useContext(CartContext);
  const [randomProducts, setRandomProducts] = useState(initialRandomProducts);
  useEffect(() => {
    if (initialRandomProducts.length === 0) {
      const newRandomProducts = [];
      const productIds = allProducts.map((product) => product._id); // Assuming product IDs are
stored in '_id'

      while (newRandomProducts.length < 4) {
        const randomIndex = Math.floor(Math.random() * allProducts.length);
        const randomProduct = allProducts[randomIndex];
        if (
          randomProduct._id !== product._id &&
          !newRandomProducts.includes(randomProduct)
        ) {
          newRandomProducts.push(randomProduct);
        }
      }
      setRandomProducts(newRandomProducts);
    }
  }, [allProducts, initialRandomProducts, product]);
```

Figure 4-14: Funktion für die SingleProductPage

Die ProductPage-Komponente erhält drei Props: product, allProducts und initialRandomProducts. Diese Props enthalten Informationen über das spezifische Produkt, alle Produkte und die initialen zufälligen Produkte. Innerhalb der Komponente wird der CartContext über den useContext-Hook verwendet, um auf Funktionen und

Daten im Warenkorb zuzugreifen. Hier wird speziell die addProduct-Funktion aus dem Kontext extrahiert, die verwendet werden kann, um Produkte zum Warenkorb hinzuzufügen.

Der Code nutzt das Paket styled-components, um vier Komponenten zu definieren: ColWrapper, Box, PriceRow und Price. ColWrapper ist ein Container mit einem Rasterlayout, das sich je nach Bildschirmgröße anpasst. Box ist ein stilisiertes Element mit weißem Hintergrund und abgerundeten Ecken. PriceRow dient als Container für eine horizontal ausgerichtete Preiszeile, und Price definiert den Stil für den Preis innerhalb dieser Zeile. Es wird eine benutzerdefinierte CSS-Animation namens 'wobbleAnimation' erstellt und auf einen vorhandenen Button angewendet. Dadurch entsteht ein Wackeleffekt, wenn der Button angeklickt wird.

```
return (
  <>
   <Header />
   <Center>
    <ColWrapper>
     <WhiteBox>
      <ProductImages images={product.images} />
     </WhiteBox>
     <div>
      <Title>{product.titel}</Title>
      <p>{product.description}</p>
      <PriceRow>
       <div>
        <Price>KES {product.price}</Price>
       </div>
       <div>
        <AnimatedButton primary onClick={() => addProduct(product._id)}>
         <CartIcon /> Add to cart
        </AnimatedButton>
       </div>
      </PriceRow>
     </div>
    </ColWrapper>
    <Box>
     <Title>Recommendations</Title>
     <ProductsGrid products={randomProducts}></ProductsGrid>
    </Box>
   </Center>
   <Footer/>
  </>
 );
}
```

Figure 4-15: HTML Code für die SingleProductPage

Dieser Code besteht aus verschiedenen strukturierten Elementen, welche das Layout und die Funktionalität für die Produktanzeige und -interaktion definieren. Ein Header-Element bildet den oberen Teil der Seite, während das Center-Element den Seiteninhalt horizontal zentriert. ColWrapper definiert eine Spaltenstruktur und beinhaltet eine WhiteBox, welche Produktbilder und Details wie Titel, Beschreibung und Preis anzeigt, sowie einen Button zum Hinzufügen zum Warenkorb. Der AnimatedButton ermöglicht das Hinzufügen von Produkten zum Warenkorb und enthält ein Warenkorb-Symbol. Der Footer bildet den Abschluss der Seite.

Ergebnis:

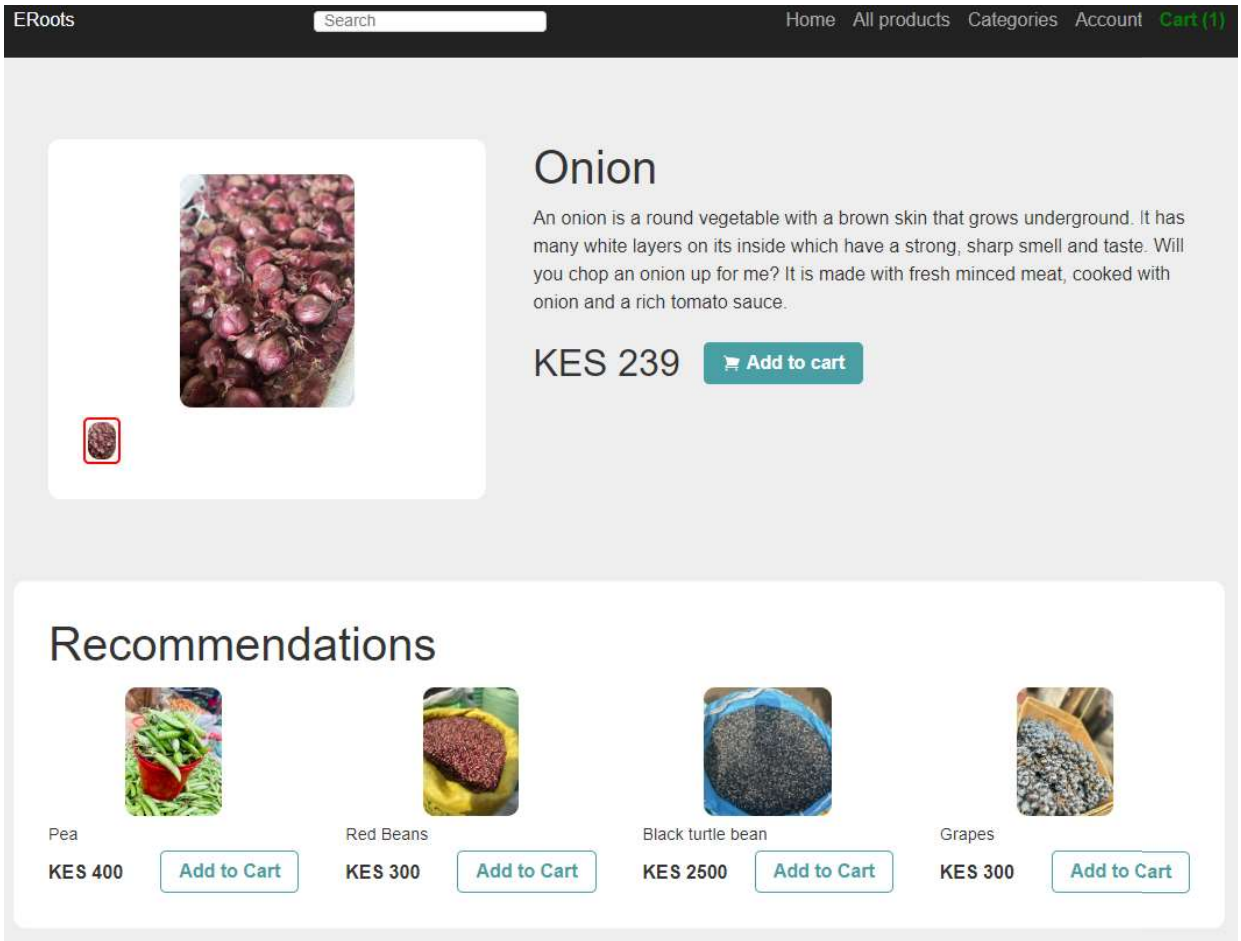


Figure 4-16: Screenshot der Single-Product Page

4.3.1.3 Whitebox und ProductsGrid

Dieser Abschnitt befasst sich mit dem Design und der Gliederung der Seite

Die WhiteBox ist ein div-Element mit weißem Hintergrund, abgerundeten Ecken und einem Innenabstand von 30 Pixeln. Außerdem werden die abgerundeten Ecken aller Bilder innerhalb der WhiteBox auf 10 Pixel gesetzt.

```
import styled from "styled-components";
import ProductBox from "./ProductBox";
const StyledProductsGrid = styled.div`
  display: grid;
  grid-template-columns: 1fr;
  gap: 40px;
  @media screen and (min-width: 768px){
    grid-template-columns: repeat(2, 1fr); /* Auf größeren Bildschirmen 2 Spalten */
  }
  @media screen and (min-width: 1024px){
    grid-template-columns: repeat(4, 1fr); /* Auf noch größeren Bildschirmen 4 Spalten */
  }
`;
export default function ProductsGrid({ products }) {
  return (
    <StyledProductsGrid>
      {products?.length > 0 && products.map(product => (
        <ProductBox key={product.id} {...product} />
      ))}
    </StyledProductsGrid>
  )
}
```

Figure 4-17: Funktion zur ProductsGrid

Der Code definiert ProductsGrid, die eine Rasteransicht von Produkten darstellt. Die StyledProductsGrid ist ein div-Element mit benutzerdefinierten Stilen, die ein Rasterlayout mit einer Spalte und einem Abstand von 40 Pixeln zwischen den Elementen festlegen. Media-Queries passen das Rasterlayout für verschiedene Bildschirmgrößen an, mit zwei Spalten für Breiten ab 768 Pixel und vier Spalten ab 1024 Pixeln. ProductsGrid rendert StyledProductsGrid und fügt für jedes Produkt in der Liste

'products' eine ProductBox-Komponente hinzu. Die Produktinformationen werden als props übergeben.

4.3.2 Graph

Dieser Abschnitt befasst sich mit den Zusatzzielen, die kein Hauptziel des Projekts sind, aber dennoch hinzugefügt wurden, um die Funktionalität des Projekts zu verbessern.

```
export function Graph() {
  const chartRef = useRef(null); // Ref für das Chart.js-Objekt
  const [ordersData, setOrdersData] = useState([]);
  useEffect(() => {
    fetchData(); // Daten beim ersten Rendern abrufen
    const interval = setInterval(fetchData, 60000);
    return () => clearInterval(interval);
  }, []);
  const fetchData = () => {
    const currentDate = new Date();
    axios.get('/api/orders')
      .then(response => {
        // Filtere die Bestellungen nach Zukunftdatum
        const ordersFiltered = response.data.filter(order => {
          if (order && order.line_items && order.line_items[0] && order.line_items[0].date) {
            const orderDate = new Date(order.line_items[0].date);
            return orderDate > currentDate;
          }
          return false;
        });
        setOrdersData(ordersFiltered);
      })
      .catch(error => {
        console.error('Error fetching orders:', error);
      });
  };
```

```
  useEffect(() => {
    renderChart();
  }, [ordersData]);
  const renderChart = () => {
    const ordersPerDay = {};
    const daysOfWeek = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday'];
    ordersData.sort((a, b) => {
      const dateA = new Date(a.line_items[0].date);
      const dateB = new Date(b.line_items[0].date);
      return dateA - dateB;
    });
    ordersData.forEach(order => {
      const orderDate = new Date(order.line_items[0].date); // Datum der Bestellung
      const options = {
        year: 'numeric', // Jahr
        month: 'long',   // Monat
        day: 'numeric'   // Tag im Monat
      };
      const orderDetails = orderDate.toLocaleDateString('en-US', options);
      ordersPerDay[orderDetails] = (ordersPerDay[orderDetails] || 0) + 1;
    });
    console.log('Orders per day:', ordersPerDay);
    const labels = Object.keys(ordersPerDay);
    const data = labels.map(label => ordersPerDay[label]);
    console.log(data);
```

```
    const chartData = {
      labels: labels,
      datasets: [{
        label: 'Amount of Orders',
        data: data,
        backgroundColor: 'rgba(75, 192, 192, 0.2)',
        borderColor: 'rgba(75, 192, 192, 1)',
        borderWidth: 1
      }]
    };
    const options = {
      scales: {
        y: {
          beginAtZero: true,
          precision: 0 // Nur ganze Zahlen auf der y-Achse anzeigen
        }
      }
    };
    const ctx = document.getElementById('exampleChart').getContext('2d');
    if (chartRef.current !== null) {
      chartRef.current.destroy();
    }
    // Neues Chart-Objekt erstellen
    chartRef.current = new Chart(ctx, {
      type: 'bar',
      data: chartData,
      options: options
    });
  };
  return <canvas id="exampleChart" width="400" height="200"></canvas>;
}
```

Figure 4-18: Funktion für den Graphen

Der Code verwendet React Hooks wie useRef, useState und useEffect sowie Axios, um Daten von einem Server abzurufen und ein Chart.js-Diagramm zu erstellen. Hooks sind Funktionen in React, die es ermöglichen, Zustand und andere React-Funktionen zu nutzen, ohne Klassen schreiben zu müssen. Zunächst wird ein ref-Objekt für das Chart.js-Element erstellt, um darauf zugreifen zu können. Anschließend wird ein Zustand für die Auftragsdaten definiert, der mit useState verwaltet wird. Der useEffect-Hook wird verwendet, um die Daten beim ersten Rendern der Komponente abzurufen und das Diagramm alle 60 Sekunden zu aktualisieren.

Die fetchData-Funktion nutzt Axios, um Daten vom Server abzurufen und im Zustand zu speichern. Sie extrahiert das aktuelle Datum und filtert die Daten, um nur zukünftige Bestellungen zu erhalten. Die renderChart-Funktion bereitet die Daten für das Balkendiagramm auf, indem sie die Anzahl der Bestellungen pro Tag berechnet und ein Datenobjekt für das Diagramm erstellt. Das Chart.js-Diagramm wird dann auf einem Canvas-Element gerendert, das im DOM mit der ID 'exampleChart' identifiziert wird.

Ergebnis:

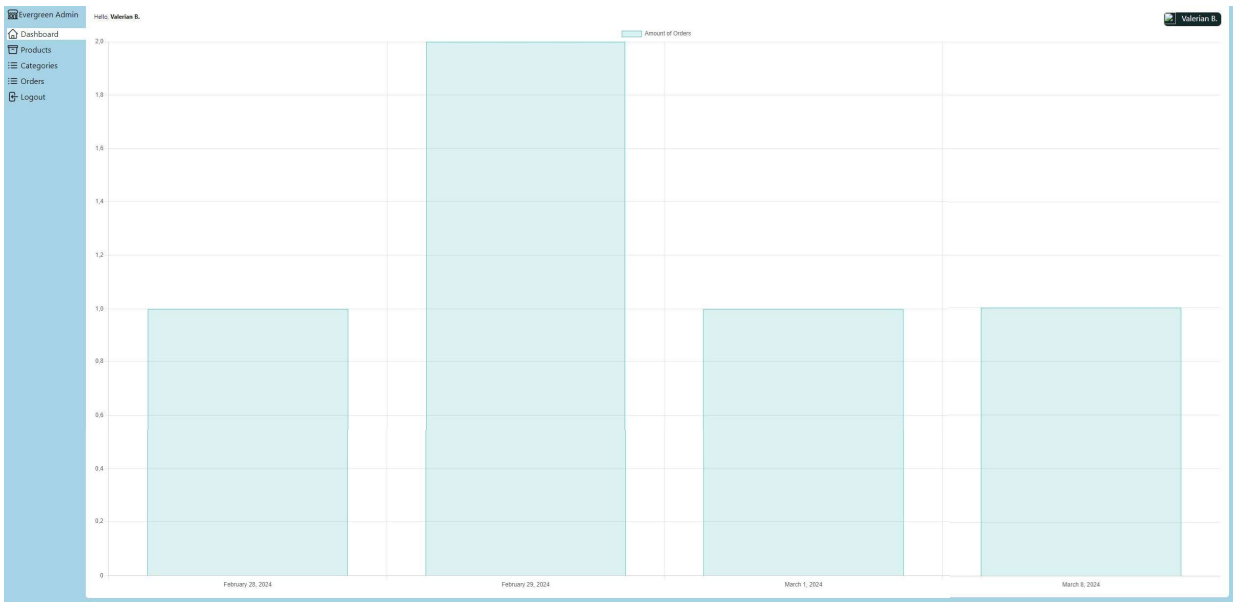


Figure 4-19: Bild vom Graphen

(*Graph*, n.d.)

4.3.3 Privacy Policy/ Impressum und Footer

Dieser Abschnitt befasst sich mit dem Impressum, sowie den Footer, wo dann das Impressum zu finden ist.

4.3.3.1 Impressum

Für die Erstellung des Impressums wurde ein Text mithilfe des 'Privacy Policy Generators' generiert. Dabei wurden sechs Schritte durchgeführt, in denen der Name und der Link der Webseite, der Geschäftstyp sowie der Standort angegeben wurden. (Termsfeed, 2023)

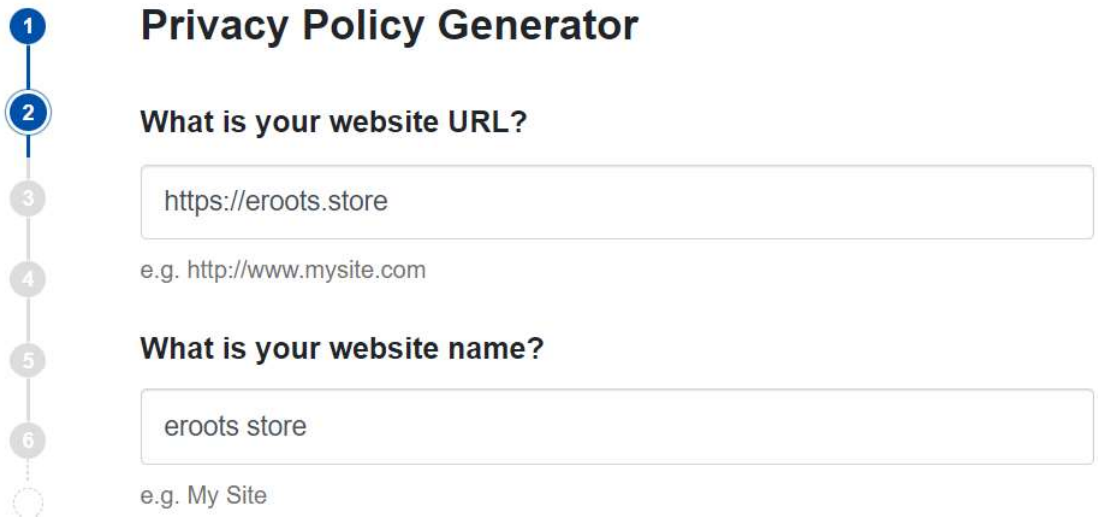


Figure 4-20: Screenshot Privacy Policy Generator Teil 1

Hier wurde der Name der Webseite, sowie der Link angegeben.

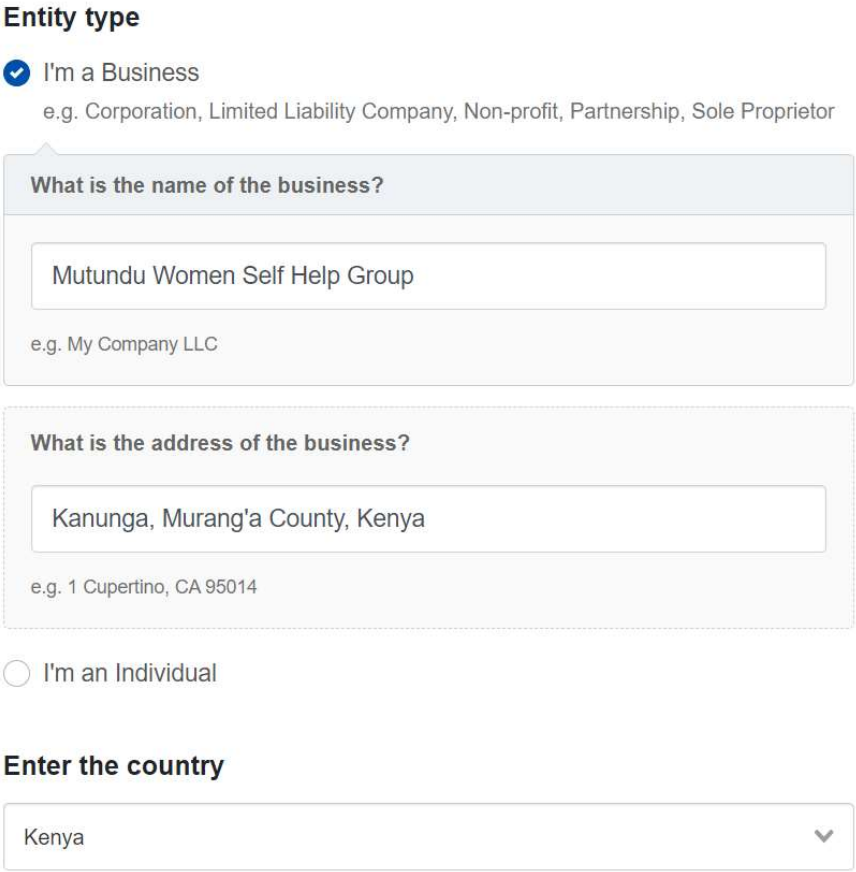


Figure 4-21: Screenshot Privacy Policy Generator Teil 2

Hier wurde der Typ des Business angegeben sowie dessen Standort.

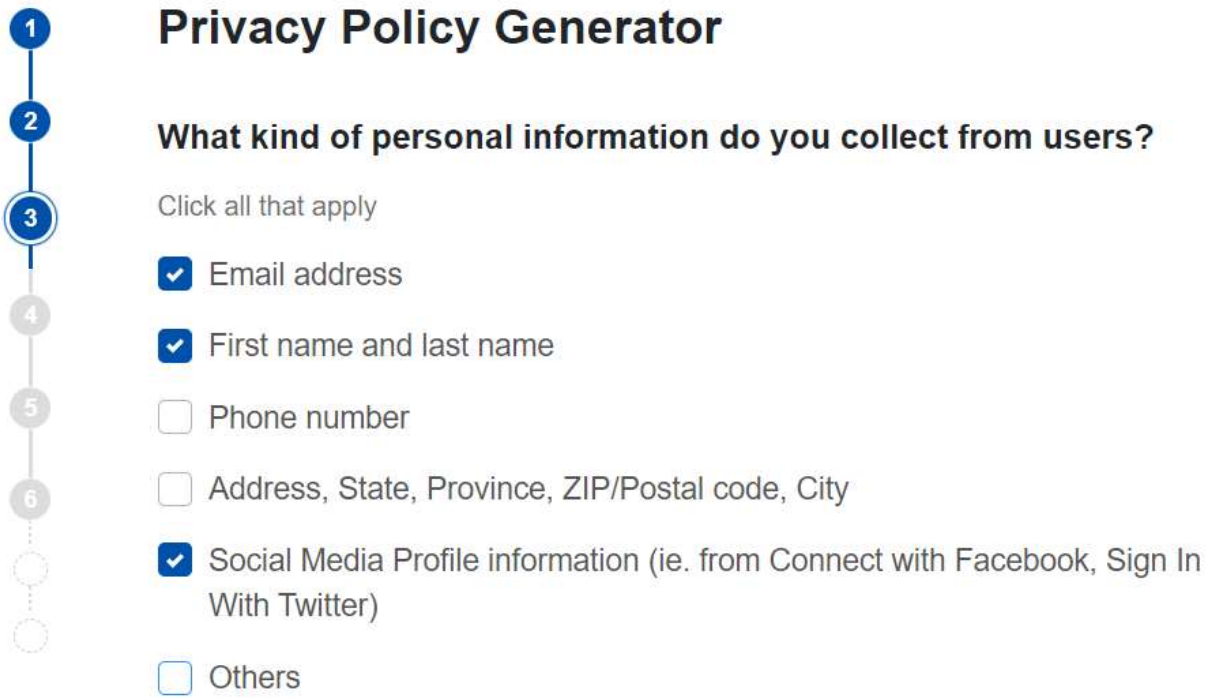


Figure 4-22: Screenshot Privacy Policy Generator Teil 3

Des Weiteren wurde erfragt, welche Daten gesammelt werden. Da die Registrierung über Google erfolgt, sammelt Google E-Mail-Adresse, Name und weitere Benutzerinformationen.

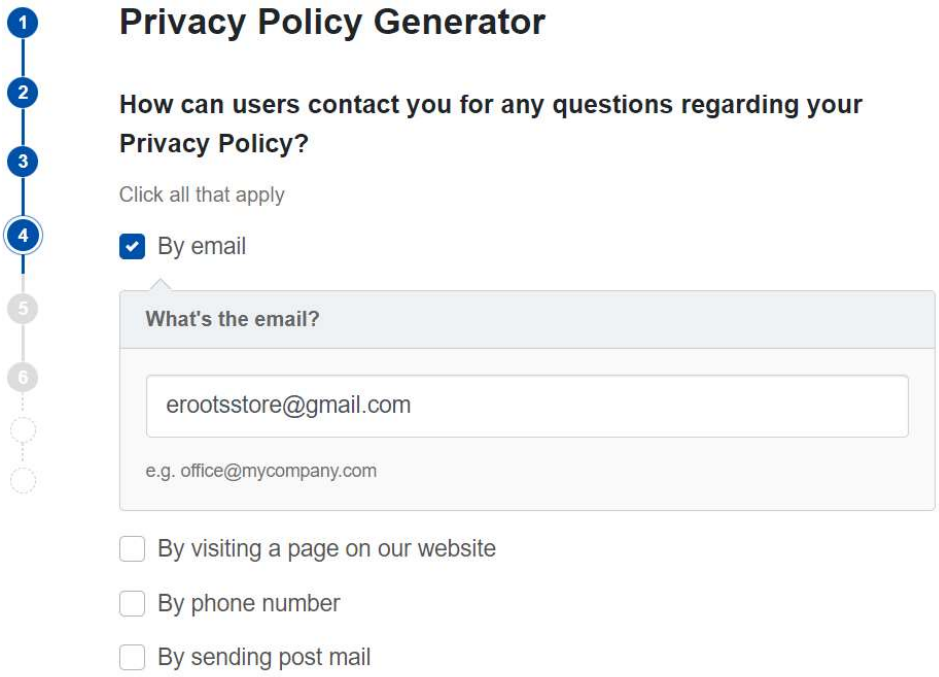
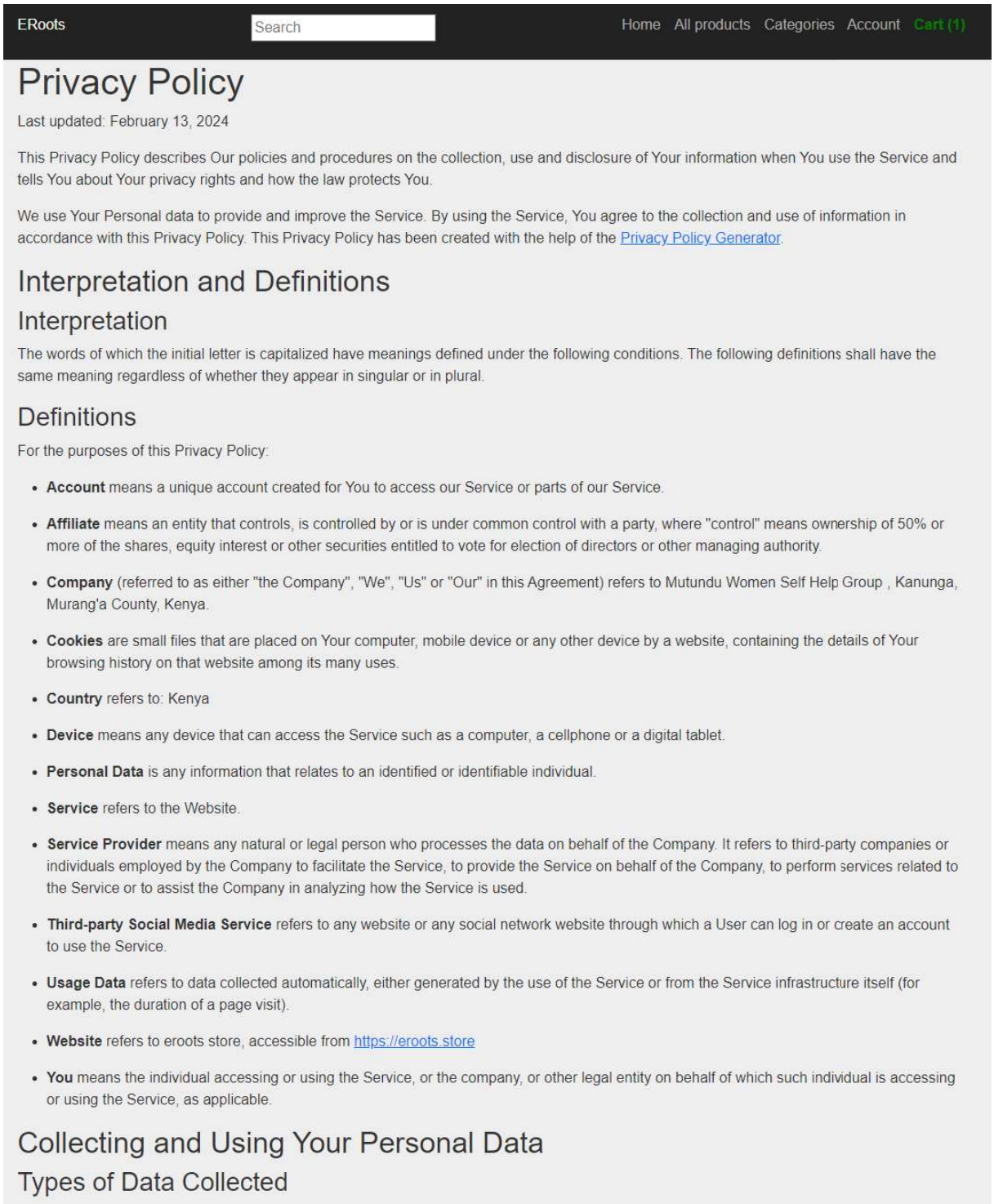


Figure 4-23: Screenshot Privacy Policy Generator Teil 4

Es wird angegeben, wo man sich bei Fragen melden kann.

Schritte 5 und 6 sind optional. (Termsfeed, 2012)

Teil des Ergebnisses:



ERoots | Search | Home All products Categories Account Cart (1)

Privacy Policy

Last updated: February 13, 2024

This Privacy Policy describes Our policies and procedures on the collection, use and disclosure of Your information when You use the Service and tells You about Your privacy rights and how the law protects You.

We use Your Personal data to provide and improve the Service. By using the Service, You agree to the collection and use of information in accordance with this Privacy Policy. This Privacy Policy has been created with the help of the Privacy Policy Generator.

Interpretation and Definitions

Interpretation

The words of which the initial letter is capitalized have meanings defined under the following conditions. The following definitions shall have the same meaning regardless of whether they appear in singular or in plural.

Definitions

For the purposes of this Privacy Policy:

- **Account** means a unique account created for You to access our Service or parts of our Service.
- **Affiliate** means an entity that controls, is controlled by or is under common control with a party, where "control" means ownership of 50% or more of the shares, equity interest or other securities entitled to vote for election of directors or other managing authority.
- **Company** (referred to as either "the Company", "We", "Us" or "Our" in this Agreement) refers to Mutundu Women Self Help Group , Kanunga, Murang'a County, Kenya.
- **Cookies** are small files that are placed on Your computer, mobile device or any other device by a website, containing the details of Your browsing history on that website among its many uses.
- **Country** refers to: Kenya
- **Device** means any device that can access the Service such as a computer, a cellphone or a digital tablet.
- **Personal Data** is any information that relates to an identified or identifiable individual.
- **Service** refers to the Website.
- **Service Provider** means any natural or legal person who processes the data on behalf of the Company. It refers to third-party companies or individuals employed by the Company to facilitate the Service, to provide the Service on behalf of the Company, to perform services related to the Service or to assist the Company in analyzing how the Service is used.
- **Third-party Social Media Service** refers to any website or any social network website through which a User can log in or create an account to use the Service.
- **Usage Data** refers to data collected automatically, either generated by the use of the Service or from the Service infrastructure itself (for example, the duration of a page visit).
- **Website** refers to eroots store, accessible from https://eroots.store
- **You** means the individual accessing or using the Service, or the company, or other legal entity on behalf of which such individual is accessing or using the Service, as applicable.

Collecting and Using Your Personal Data

Types of Data Collected

Figure 4-24: Eingefügtes Impressum in der Web-Applikation

4.3.3.2 Footer

```
const Footer = () => {
  const [isVisible, setIsVisible] = useState(true);
  useEffect(() => {
    const handleScroll = () => {
      const windowHeight =
        window.innerHeight || document.documentElement.clientHeight;
      const bodyHeight =
        document.body.scrollHeight ||
        document.documentElement.scrollHeight;
      const footer = document.getElementById("footer");
      const footerOffset = footer ? footer.offsetHeight : 0; // Höhe des Footers berücksichtigen
      const isFooterVisible =
        windowHeight + window.scrollY >= bodyHeight - footerOffset;
      setIsVisible(isFooterVisible);
    };
    window.addEventListener("scroll", handleScroll);
    return () => window.removeEventListener("scroll", handleScroll);
  }, []);
  return (
    <StyledFooter id="footer" isVisible={isVisible}>
        <StyledNav>
          {/* Link zu Legal.js */}
          <NavLink href="/legal">Privacy Policy</NavLink>
        </StyledNav>
    </StyledFooter>
  );
};
```

Figure 4-25: Funktion für den Footer

Der Code definiert einen Footer für die Webseite. Der Footer ist mit styled-components gestylt und hat eine feste Positionierung. Die Sichtbarkeit des Footers wird durch einen Zustand (isVisible) gesteuert, der standardmäßig auf true gesetzt ist. Ein useEffect-Hook überwacht das Scrollen der Seite, um zu bestimmen, ob der Footer im sichtbaren

Bereich des Viewports liegt. Die Funktion handleScroll berechnet dies basierend auf der Fenster- und Dokumentenhöhe sowie der Höhe des Footers. Der Zustand isVisible wird entsprechend aktualisiert, um den Footer einzublenden oder auszublenden.

Der Footer wird in einem StyledFooter-Container gerendert, der die Positionierung und Animationen definiert. Darüber hinaus gibt es ein Navigationsmenü (StyledNav) mit einem einzigen Link zur Datenschutzrichtlinie. Die Elemente im Navigationsmenü sind zentriert, um sie in der Mitte des Footers auszurichten. Insgesamt ermöglicht diese Implementierung eine dynamische Anpassung der Sichtbarkeit des Footers basierend auf dem Scrollverhalten der Seite.

4.3.4 Aufgetretene Fehler

Dieser Abschnitt behandelt die während des Projekts aufgetretenen Fehler.

Auf der All-Product-Page wurden Produkte nicht geladen und Bilder von den Produkten nicht angezeigt. Außerdem waren einige Produkte außerhalb des Grids und somit nicht auf dem Bildschirm sichtbar. Dieser Fehler wurde behoben, indem festgestellt wurde, dass die Verbindung zur Datenbank zwar vorhanden war, jedoch die Produkte und Bilder nicht im richtigen Format abgerufen wurden. Die Buttonanimation funktionierte nicht, wurde jedoch behoben, indem eine neue Funktion erstellt wurde, die die Buttonanimation und das Hinzufügen in den Einkaufswagen vereint. Auf der All-Product-Page wurden die Produkte nicht geladen und die Bilder der Produkte wurden nicht angezeigt. Außerdem waren einige Produkte außerhalb des Rasters und nicht auf dem Bildschirm sichtbar. Auch die Animation der Schaltfläche funktionierte nicht und die Schaltfläche zum Hinzufügen des Produkts zum Warenkorb war fehlerhaft. Diese Fehler wurden behoben, indem der HTML-Code neu strukturiert und das ProductGrid auf Fehler überprüft wurde.

Bei der Einzelproduktseite traten ähnliche Probleme auf, wie beispielsweise fehlende Bilder und ein fehlerhaftes Raster. Zudem erschien eine Fehlermeldung beim Auswählen des Produkts. Diese Fehler wurden behoben, indem überprüft wurde, ob die richtige ID abgefragt wurde.

Beim Graphen sind Fehler aufgetreten. Zunächst wurden die x- und y-Achsen falsch geladen. Dies wurde behoben, indem ein funktionierender Graph aus dem Internet genommen wurde. Anschließend wurden die Orders im Graphen nicht angezeigt und der Graph wurde nicht aktualisiert, nachdem eine neue Order gemacht wurde. Um die Fehler zu beheben, wurde überprüft, in welchem Format die Bestellungen aus der Datenbank abgerufen wurden. Es stellte sich heraus, dass sie als String und nicht als DateTime gespeichert waren. Außerdem wurde sichergestellt, dass das Datum der Bestellung und nicht das Datum der Erstellung der Bestellung abgerufen wurde. Um sicherzustellen, dass sich der Graph aktualisiert, wurde eine Zeitspanne eingebaut, die dafür sorgt, dass der Graph alle Minute aktualisiert wird. Es ist ein Fehler aufgetreten, der die y-Achse auf nur 10 Werte beschränkt hat. Dadurch wurden mehr als 10 vorhandene Orders nicht angezeigt. Dieser Fehler konnte behoben werden, indem ein Beispielgraph aus dem Internet verwendet wurde, der die x- und y-Werte anpassen kann. Wenn keine Daten vorhanden sind, wird der Graph nicht angezeigt. Dies wurde durch den Einbau des Graphen aus dem Internet mit den anpassenden Werten behoben. Ebenfalls wurde angezeigt, dass auch vergangene Orders angezeigt werden, was jedoch behoben wurde, indem nur die Orders von heute gefiltert wurden.

Der Footer hatte das Problem, dass er beim Scrollen sichtbar bleibt. Dieses Problem trat auf, da die Seite zu groß war, um den Footer von Anfang an anzuzeigen, aber zu klein war, um ihn aus dem Sichtfeld zu entfernen. Dadurch ist es passiert, dass der Footer beim Scrollen verschwunden ist, aber sobald man aufgehört hat zu scrollen, ist er wieder aufgetaucht. Dies wurde behoben, indem mehr Produkte eingeführt wurden. Dadurch wurde die Seite größer, was dazu führte, dass man mehr scrollen musste. Dadurch wurde der Footer ausgeblendet, solange man nicht bis ganz nach unten gescrollt hat.

All diese Probleme wurden mithilfe von Professoren, Mitschülern und dem Internet gelöst.

4.4 Lessons Learnd

4.4.1 Was lief gut?

Die Kommunikation und die Stimmung im Projekt waren durchgehend gut.

Obwohl wir uns erst mit Node/Next vertraut machen mussten und lernen mussten, wie wir am besten damit umgehen, haben wir meistens schnell Lösungen gefunden, um Fehler zu beheben.

Die Aufgaben wurden in der Regel zeitgerecht erledigt. Falls etwas nicht rechtzeitig fertig wurde, hatten wir dank unseres guten Projektmanagements immer noch genug Zeit, um Fehler zu verbessern und nicht in Stress zu geraten.

Obwohl es hin und wieder Probleme mit den einzelnen Funktionen gab, konnten wir sie dank unserer guten Zusammenarbeit immer wieder verbessern.

Die Kommunikation mit unserem Projektbetreuer war gut, da wir regelmäßig Besprechungen abhielten und er sich nach dem Fortschritt des Projekts regelmäßig erkundigte.

Die Erstellung eines Graphen, der sich immer an die Bestellungen anpasst, war eine schwierige Aufgabe, die jedoch nach langem Suchen, vielen Fragen und vielen Versuchen erfolgreich gelöst wurde.

4.4.2 Was lief schlecht?

In Tagen oder Wochen, in denen einige Teammitglieder krank waren, kam es gelegentlich zu Fehlkommunikation und Missverständnissen. Diese wurden jedoch immer schnell behoben.

Jede Person erhielt ihre eigene Aufgabenverteilung, die auch eingehalten wurde. Es gab jedoch Situationen, in denen eine Person eine Aufgabe hätte erledigen sollen, die dann von einer anderen Person übernommen wurde.

4.5 Begleitprotokoll gemäß § 9 Abs. 2 PrO

DATUM	STUNDEN	PROJEKTPHASE	BESCHREIBUNG	RESSOURCEN
31.05.2023	4:25	Vorprojektphase	Projektbesprechung 1: Was für Tool verwenden wir für das Projekt?	
31.05.2023	1:15	Vorprojektphase	Projektbesprechung 2: Wer macht was? Was machen/brauchen wir überhaupt alles?	
05.06.2023	0:10	Vorprojektphase	Marktanalyse	
22.08.2023	1:10	Projektmanagement	Besprechung mit Team über Fortschritt im Projekt	
07.09.2023	2:40	Projektmanagement	Vorbereitung der Dokumente	
11.09.2023	0:50	Schule	Vorstudie begonnen	
13.09.2023	1:50	Schule	Vorstudie weitergearbeitet und laut Vorlage verändert, Icons in die Dokumente eingefügt	
19.09.2023	0:30	Schule	Besprechung mit Herr Professor Schwarz	
01.10.2023	1:20	Projektmanagement	PHB 1.1, 1.6, 1.18 zum teil	
01.10.2023	1:45	Projektmanagement	PHB	
01.10.2023	2:50	Projektmanagement	PHB	
02.10.2023	1:40	Schule	Arbeit an den morgen abzugebenden Aufgaben	
03.10.2023	0:50	Vorprojektphase	Besprechung mit Professor Schwarz über die Zitierweise für die Diplomarbeit	
03.10.2023	1:25	Vorprojektphase	Die Vorstudie wurde weiter bearbeitet.	
04.10.2023	1:50	Schule	Die PHB wurde besprochen und korrigiert.	
06.10.2023	4:25	Analyse	Lastenheft	
08.10.2023	0:40	Vorprojektphase	Recherche nach HTTP	

10.10.2023	0:50	Schule	Die Architektur des Systems mit Professor Pirker besprochen. Die Änderungen wurden gemäß dem Eroots-4-Meeting-Protokoll v1 vorgenommen.
10.10.2023	3:50	Vorprojektphase	Vorstudie 11.2 verbessert und weiter gemacht aber noch nicht fertig
11.10.2023	1:50	Schule	Vorstudie, PHB 1.4
12.10.2023	1:00	Projektmanagement	PHB PSP
12.10.2023	1:30	Projektmanagement	PHB PSP verbesserung
14.10.2023	4:15	Implementierung	HTTPS Responses. Es wurde kein Commit durchgeführt, da dieser fehlgeschlagen ist 12, 12.1 Conclusion
15.10.2023	2:00	Projektmanagement	PHB
16.10.2023	0:50	Schule	Angefangen mit der Vorstudie
17.10.2023	0:20	Projektmanagement	Besprechung mit Professor Schwarz und Team
18.10.2023	2:15	Vorprojektphase	Recherche an den Programmiersprachen, Testung der Sprachen
19.10.2023	2:40	Schule	Arbeit an der Vorstudienpräsentation und der Vorstudie
19.10.2023	0:30	Vorprojektphase	Vorstudie weitergearbeitet
20.10.2023	1:30	Vorprojektphase	Vorstudie weitergearbeitet und Verbesserungen gemacht
21.10.2023	0:40	Vorprojektphase	Vorstudienpräsentation mit PTM korrigiert
22.10.2023	0:55	Vorprojektphase	Vorstudie fertiggestellt
23.10.2023	1:00	Vorprojektphase	Nutzwertanalyse verbessert und Bilder zitiert
25.10.2023	0:50	Schule	1.12 Betrachtungsobjekteplan
06.11.2023	1:40	Schule	Lastenheft UseCase

13.11.2023	2:00	Implementierung	Webhook erstellt	Visual Studio Code
13.11.2023	1:30	Projektmanagement	PHB: PSP Verbesserung, 1.9, 1.10	
13.11.2023	2:45	Projektmanagement	PHB: 1.9, 1.10, 1.3	
20.11.2023	3:20	Implementierung	Pflichtenheft	Visual Studio Code
23.11.2023	2:15	Implementierung	All Product Page erstellt	Visual Studio Code
25.11.2023	3:05	Implementierung	All Product Page gearbeitet	Visual Studio Code
26.11.2023	2:50	Implementierung	All Product Page gearbeitet	Visual Studio Code
29.11.2023	3:15	Implementierung	All Productspage fertig, bei One Product gearbeitet	Visual Studio Code
30.11.2023	3:10	Implementierung	Projektarbeit	Visual Studio Code
06.12.2023	1:50	Schule	Statusbericht vom Stand 06.12.2023 im Unterricht	
09.12.2023	5:30	Implementierung	front responsive	Visual Studio Code
10.12.2023	0:45	Implementierung	front responsive	Visual Studio Code
10.12.2023	2:45	Implementierung	front responsive fertig	Visual Studio Code
11.12.2023	0:10	Implementierung	Pull Request erstellt	Visual Studio Code
14.12.2023	01:55	Implementierung	Button animation hinzugefügt	Visual Studio Code
14.12.2023	1:50	Implementierung	Probiert eine Animation für das Cart hinzuzufügen	Visual Studio Code
29.12.2023	0:30	Projektmanagement	Projektteambesprechung	
07.01.2024	1:45	Projektmanagement	Projektbesprechung	
08.01.2023	04:40	Implementierung	Animationen fertiggestellt	Visual Studio Code
09.01.2023	04:50	Implementierung	Animationen neu hinzugefügt weil nicht gespeichert, Impressum angeschaut, Methoden für einen Graphen angeschaut/probiert	Visual Studio Code
10.01.2024	1:20	Implementierung	Button animation geändert	Visual Studio Code

11.01.2024	1:50	Implementierung	Footer angefangen (Probleme mit der Position wird bei zu großen Bildschirmen in der Mitte des Bildschirms angezeigt)	Visual Studio Code
12.01.2024	2:25	Implementierung	Footer Verbesserungen gemacht + Impressum angefangen	Visual Studio Code
13.01.2024	2:40	Implementierung	Footer fertig	Visual Studio Code
14.01.2024	1:30	Implementierung	Angefangen mit den Graphen aber noch viele Fehlermeldungen	Visual Studio Code
14.01.2024	1:00	Implementierung	Graph wird angezeigt aber noch nichts zu sehen	Visual Studio Code
14.01.2024	2:30	Implementierung	Graph wird nicht mehr angezeit wegen neuen Fehlermeldungen	Visual Studio Code
24.01.2024	1:50	Implementierung	Graph gearbeitet	Visual Studio Code
25.01.2024	2:10	Implementierung	Fehler beim Graphen versucht zu verbessern	Visual Studio Code
26.01.2024	04:00	Implementierung	Alte Vehler verbessert aber neue hinzugekommen (daten werden empfangen aber falsch ausgegeben	Visual Studio Code
26.01.2024	04:00	Implementierung	Graph wird komplett angezeigt aber falsche Daten (x-Achse falsch + bei keinen Daten wird der Graph nicht angezeigt)	Visual Studio Code
27.01.2024	03:30	Implementierung	Graph wird komplett angezeigt	Visual Studio Code
30.01.2024	02:40	Implementierung	Graph Implementierung	Visual Studio Code
30.01.2024	02:30	Implementierung	neue Fehler beim Graphen versucht zu verbessern	Visual Studio Code
31.01.2024	00:50	Implementierung	Graph bearbeitet	Visual Studio Code
31.01.2024	03:30	Implementierung	Graph bearbeitet	Visual Studio Code
05.02.2024	02:15	Implementierung	Graph Fehler verbessert/ fertiggestellt	Visual Studio Code
06.02.2024	00:05	Implementierung	Fehler beim Pull-Request	Visual Studio Code

12.02.2024	02:00	Implementierung	Fehler Verbessert	Visual Studio Code
13.02.2024	02:35	Implementierung	Weitere Fehler beim Graphen verbessert	Visual Studio Code
14.02.2024	03:00	Implementierung	Der Footer wurde überarbeitet, da er Fehler enthielt.	Visual Studio Code
15.02.2024	01:50	Implementierung	Impressum gemacht	Visual Studio Code
16.02.2024	01:00	Implementierung	Versucht Footer dynamisch zu machen	Visual Studio Code
17.02.2024	04:00	Implementierung	Footer dynamisch gemacht	Visual Studio Code
19.02.2024	01:45	Implementierung	Produkte gesucht und hinzugefügt.	Visual Studio Code
21.02.2024	04:00	Implementierung	Neu aufgetretener Fehler beim Graphen verbessert und Produkte hinzugefügt	Visual Studio Code
22.02.2024	00:50	Projektmanagement	Diplomarbeits besprechung	
23.02.2024	3:55	Implementierung	Produkte hinzugefügt	Visual Studio Code
24.02.2024	3:45	Diplomarbeit	Grundlagen geschrieben	
24.02.2024	4:45	Diplomarbeit	JavaScript/Node.Js + Python + Java + C# Vor und Nachteile	Word
25.02.2024	4:30	Implementierung	Fehler bei der Graphen-Darstellung korrigiert.	Visual Studio Code
25.02.2024	5:45	Diplomarbeit	Ergebnisse: All Product and Product Page + Graph + Footer	Word
26.02.2024	03:45	Diplomarbeit	Korrekturlesen + Zitieren eingefügt + Impressum fertig	Zotero, Word, Visual Studio Code
29.02.2024	4:00	Implementierung	Produkte hinzugefügt (Bilder, Beschreibung, …)	Visual Studio Code
01.03.2024	1:45	Implementierung	Aufgetretene Fehler verbessert und Schönheitsfehler verbessert	Visual Studio Code
05.03.2024	0:20	Projektmanagement	Diplomarbeit mit Schwarz besprochen	
09.03.2024	6:00	Diplomarbeit	Verbesserungen von der Diplomarbeit umgesetzt	Word
12.03.2024	2:50	Diplomarbeit	Vor und Nachteile vereinheitlich	Word

15.03.2024	2:10	Diplomarbeit	Begleitprotokoll eingefügt	Word
16.03.2024	2:30	Diplomarbeit	Code mit den anderen abgeglichen	Word
18.03.2024	2:20	Diplomarbeit	Zusammenlegung der Diplomarbeit	Word
SUMME	182:00			

*Berechnete Gesamtzahl ohne die in der Schule und im Vorstudie geleistete Arbeitszeit.

5 Task description “Planning and implementing/developing of deployment, testing and quality assurance” by Elias Andrei.

In software development, the deployment, testing, and quality assurance of web applications are essential for creating reliable digital solutions.

Deployment involves setting up infrastructure, automating code deployment, and ensuring smooth operation in the intended environment. From selecting hosting platforms to configuring servers and databases, each step is executed to ensure the availability and functionality of the deployed application.

Testing plays a critical role in verifying the correctness, functionality, and performance of software applications. Testing methodologies cover a range of levels, from unit tests that verify individual components to integration and system testing, which assess the application's behaviour as a cohesive whole. End-to-end testing simulates real user interactions to ensure a consistent and satisfactory user experience. Performance testing evaluates responsiveness, scalability, and stability under various conditions, while acceptance testing validates compliance with user requirements before deployment.

Quality assurance (QA) is the framework that ensures software development processes adhere to established standards and practices. It includes defining standards, conducting technical reviews, and overseeing change management processes to maintain the integrity and reliability of software products. Quality assurance (QA) is a crucial aspect of software development that involves identifying, assessing, and mitigating potential risks within the development process to minimize their impact on the final product.

Deployment, testing, and quality assurance are integral parts of the software development lifecycle. They ensure that applications are delivered with high quality, functionality, and reliability. By adhering to best practices in these areas, software development teams can deliver solutions that meet the evolving needs and expectations of users and stakeholders, driving innovation and success in the digital landscape.

This chapter explores the current state and best practices in the mentioned areas, proposes several solutions and outlines some of the implemented results of the chosen solution during the project.

5.1 Basics

5.1.1 Deployment

Deployment encompasses all the activities necessary for making the application available to end users. It involves setting up the infrastructure, automating the process of pushing code changes into production, and ensuring that the application runs smoothly in its intended environment.

This phase includes selecting an appropriate hosting platform, configuring servers, databases and other necessary components.

The result of the deployment phase should be a fully functional web application available on the web.

The following sections discuss important aspects of web application deployment.

5.1.1.1 Environment Setup

The initial phase involves selecting an appropriate hosting platform in line with the project requirements. Factors such as scalability, performance, cost and ease of use should be carefully considered. This is followed by server configuration, which may involve setting up virtual machines or containerised solutions such as Docker. In addition, selecting and configuring an appropriate database system, such as PostgreSQL or MongoDB, is essential to maintaining a robust foundation.

5.1.1.2 Continuous Integration/Continuous Deployment (CI/CD)

To facilitate streamlined development workflows, a robust CI/CD pipeline must be established. This involves integrating version control systems, such as Git, with a CI/CD tool to enable automated builds and deployments as code is committed.

Continuous integration and continuous deployment (CI/CD) pipelines are established to automate the build and release process and reduce manual intervention. Different deployment strategies, such as blue-green deployments or canary releases, are used to minimise downtime and mitigate risk during updates. In addition, robust monitoring and logging practices can be implemented to track the performance of the application and identify and address any issues in a timely manner. Scaling strategies are put in place to handle increased traffic and ensure a seamless user experience. (cf. GeekforGeeks, What is CI/CD?, 2021)

5.1.1.3 Monitoring and Logging

For effective deployment management, robust monitoring and logging mechanisms should be implemented. Tools such as Prometheus for metrics collection and Grafana for visualisation are invaluable for tracking critical performance metrics. Alerting configurations should be in place to promptly notify the team of anomalies or critical issues. In addition, centralised logging solutions such as the ELK stack or Splunk should be integrated to facilitate efficient troubleshooting and debugging. (cf. GeekforGeeks, Working with Monitoring and Logging Services, 2023)

5.1.1.4 Scaling

In anticipation of increased traffic, it is imperative that scalability measures are put in place. This includes implementing both horizontal scaling, achieved by adding more

servers or instances, and vertical scaling, achieved by upgrading individual servers with additional resources. Load balancing mechanisms should be employed to evenly distribute traffic across multiple servers to optimise performance.

5.1.1.5 Deployment Strategies

Deployment strategies are methods or approaches used to release software applications or updates into production environments. These strategies help to ensure that deployment processes are efficient, reliable and minimise disruption to users.

5.1.1.5.1 Blue-Green Deployment

This strategy maintains two identical production environments (Blue and Green). The current version of the application runs in one environment (e.g. blue) while the updated version is deployed to the other environment (e.g. green). Once the green environment is deemed stable, traffic is switched from blue to green, making it the new production environment.

5.1.1.5.2 Canary Deployment

Canary deployment involves releasing a new version of the software to a small subset of users or servers before rolling it out to the entire infrastructure. This allows the new version to be tested in the real world with minimal risk. If the canary deployment proves successful, the update is gradually rolled out to more users or servers.

5.1.1.5.3 Rolling Deployment

Rolling deployment is the gradual deployment of software updates across the infrastructure, one subset (e.g. server, cluster) at a time. This approach ensures that the application remains available throughout the deployment process as only part of the infrastructure is updated at any one time.

5.1.2 Testing

Testing is a important part of software development that aims to ensure the application functions correctly, meets its requirements, and remains robust in various scenarios. It involves multiple levels of examination, starting with unit tests that validate individual components. Integration testing focuses on interactions between different parts of the application, while end-to-end (E2E) testing simulates real user interactions.

The following sections discuss important aspects of web application testing.

5.1.2.1 Testing Principles

Meeting Customer Requirements

Ensuring that all tests align with customer requirements is essential to software quality assurance. This involves a thorough understanding of customer expectations, specifications, and intended use cases. By aligning testing activities with these requirements, the software can be validated against the criteria that truly matter to end-users and customers, promoting their satisfaction and product success.

Optimal Testing Based on Risk Assessment

Acknowledging the impossibility of exhaustive testing, Software Quality Assurance (SQA) focuses on performing an optimal amount of testing. This approach is guided by a thorough risk assessment of the application. By identifying and prioritising high-risk areas, the team can allocate testing resources efficiently, addressing critical

components that are more likely to contain defects, thus maximising the effectiveness of the testing process.

Plan all tests before implementation

This includes defining test objectives, outlining test cases, and specifying the criteria for test success. Test planning ensures that testing efforts are organized, systematic, and aligned with project goals, facilitating a structured and well-coordinated approach to quality assurance.

Pareto Rule in Testing

This is based on the Pareto rule, also known as the 80/20 rule, which recognises that 80% of errors can be traced back to a relatively small subset of program components (20%). Software Quality Assurance (SQA) activities are strategically focused on intensively testing critical program components. Addressing these critical components can have a substantial impact on overall software reliability and quality.

Incremental Testing from Small to Large Parts

The testing process follows an incremental approach, starting with examining small, isolated parts of the software and gradually extending to larger, integrated components. This progressive testing strategy enables early detection of defects in individual units, providing a solid foundation for ensuring that these components function correctly when integrated into the broader system. It facilitates an efficient debugging process and aids in identifying issues at an early stage, thereby reducing the cost and complexity of later-stage bug fixes.

5.1.2.2 Types of Testing

Software testing can be a very broad subject. From verifying individual units of code to assessing the overall performance and user experience, different types of testing serve distinct purposes in identifying errors and ensuring that software meets its intended requirements. Some common types of software testing include component testing, unit testing, integration testing, end-to-end (E2E) testing, performance testing, system testing and acceptance testing. Each type of testing brings its unique approach and focus, collectively contributing to the overall quality assurance process of software development. (cf. Luo, Software Testing Techniques, p. 2)

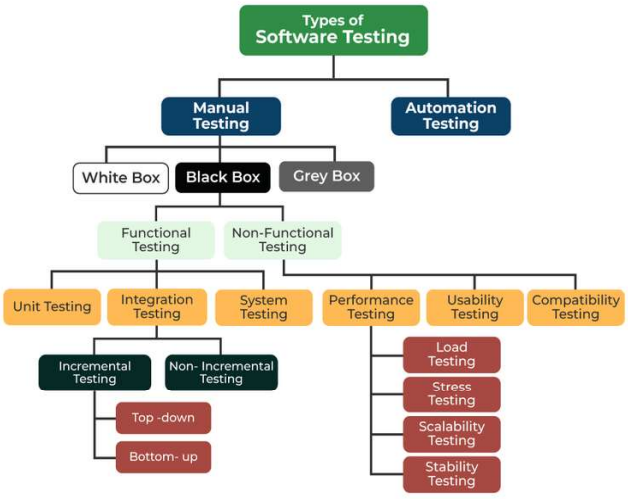


Figure 5-1: Types of Software Testing arranged

5.1.2.2.1 Black Box

The commonly used software testing techniques almost all fall into the category of 'black box' testing. This type of testing is a method of testing software where the tester does not know the internal workings or structure of the system being tested. Instead, the tester focuses solely on the inputs and outputs of the software, treating it as a 'black box' whose internal mechanisms are hidden from view.

Testers evaluate the functionality of the software based on its specifications and requirements, without any knowledge of its underlying code or implementation details. This approach enables testers to evaluate the software's performance of its intended functions and determine if it meets the expected behavior specified in the requirements documentation.

5.1.2.2.2 Component Testing

Component testing plays a pivotal role in ensuring the seamless integration and functionality of different modules or larger functional units within the application. Testing libraries such as Jest and Enzyme are employed to create comprehensive test suites for components, covering various use cases and edge scenarios. Component testing involves verifying not only the individual behavior of each component but also their interactions and integration with other components.

To enable component testing, mock objects or stubs are used to simulate the behavior of dependencies or external services. This ensures that each component can be tested in isolation without relying on external factors, enhancing test independence and enabling more effective debugging and troubleshooting.

5.1.2.2.3 Unit Testing

Unit testing ensures the correctness and functionality of individual components. To verify the rendering and behaviour of components under various scenarios, testing libraries such as Cypress and React Testing Library are utilized to construct test cases. In addition, the practice of mocking external dependencies or services is crucial for isolating unit tests and maintaining test independence. (cf. Olan, Unit testing: Test early, test often, 2003, p. 320)

5.1.2.2.4 Integration Testing

Integration testing, on the other hand, deals with the interaction between different components or modules within the application. By focusing on scenarios where multiple components collaborate, it ensures seamless integration and functionality. Test cases are created to cover scenarios such as form submissions, data fetching, and state management. Integration tests may use stubs or mocks to simulate interactions with external services or APIs.

5.1.2.2.5 End-to-End (E2E) Testing

E2E Testing tools like Cypress or Selenium, simulates user interactions with the application and validates critical user flows such as registration, product search, and the checkout process. These tests help identify discrepancies in behaviour, ensuring a consistent and satisfactory user experience.

5.1.2.2.6 Performance Testing

Performance testing evaluates the performance characteristics of the software, such as its responsiveness, scalability, and stability under various conditions like load and stress. Load and performance tests guarantee that the application can handle the anticipated load without compromising performance.

Scalability testing is a type of performance testing. The text is already well-written and meets the desired characteristics. No changes are necessary. The objective is to determine how well a software application can scale up or down as the workload changes, for example, an increase in the number of users, transactions, or data volume.

5.1.2.2.7 System Testing

System testing is a thorough testing phase that evaluates the entire integrated system as a whole. It verifies that all software components work together correctly to achieve the desired functionality outlined in the system requirements. Unlike unit testing and integration testing, which focus on individual modules or component interactions, system testing assesses the system's behaviour and performance in a real-world environment.

This testing phase encompasses a broad range of scenarios, such as normal operation, error handling, boundary conditions, and interoperability with external systems or dependencies. System testing guarantees that the software satisfies the specified requirements and is prepared for deployment to end-users or customers.

5.1.2.2.8 Acceptance Testing

Acceptance Testing, on the other hand, is the final phase of testing before the software is released to users or customers. It involves evaluating the software's compliance with business requirements and user expectations to determine whether it is acceptable for deployment. Acceptance testing can be performed by end-users, stakeholders, or quality assurance teams, depending on the project's requirements. This testing phase typically includes two main types: User Acceptance Testing (UAT), where end-users validate that the software meets their needs and operates as expected in their environment, and Business Acceptance Testing (BAT), where stakeholders verify that the software aligns with business objectives and delivers the intended value. Successful completion of acceptance testing indicates that the software is ready for production deployment.

5.1.2.2.9 Automated Testing

Testing automation can be achieved by connecting the testing software to the Continuous Integration/Continuous Deployment (CI/CD) pipeline from a version management software. Automating testing procedures enables quick identification and prevention of potential issues. Tests can be defined in this way to ensure that important parts of the project are functioning properly after every commit.

5.1.2.3 Levels of Testing/Testing during a Project

As described in Section 5.1.2.1 below the final point, *Incremental Testing from Small to Large Parts*, the testing process should begin with the testing of the smallest project components and gradually extend to larger elements. This methodical progression ensures thorough verification.

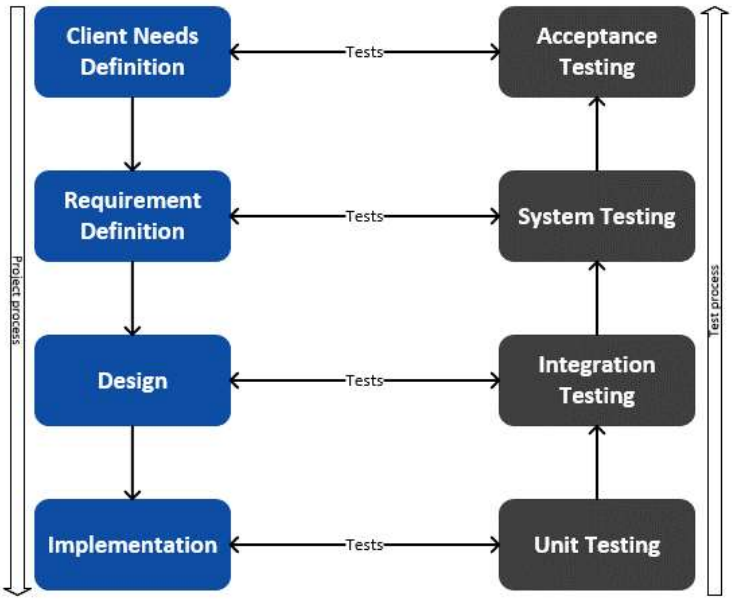


Figure 5-2: Levels of Testing

Beginning with unit testing at the implementation phase, where individual units or components are scrutinized for correctness. Subsequently, integration testing assesses the interactions between these units, followed by system testing to evaluate the functionality of the integrated system. Finally, acceptance testing verifies the compliance of the entire project with client demands and requirements gathered during the initial phases. This structured approach ensures comprehensive testing coverage and enhances confidence in the project's quality and readiness for deployment. It moves from the granularity of individual components to the entirety of the project. (cf. GeekforGeeks, Levels of Software Testing, 2020)

5.1.3 Quality Assurance in Software Development

Quality Assurance (QA) is an approach to ensuring that the application functions correctly and meets certain standards of quality in terms of code, security, accessibility, and usability. (cf. GeekforGeeks, Software Quality Assurance - Software Engineering, 2019)

As can be observed, testing and quality assurance share many similarities as both aim to achieve the highest possible software quality.

The following sections discuss important aspects of Quality Assurance in Software Development

5.1.3.1 Standarts, Reviews and Audits

Software Quality Assurance (SQA) ensures that software engineering standards set by organizations such as IEEE and ISO are followed diligently throughout all stages of development. This involves verifying that all work products, including documentation, code, and processes, conform to these established standards.

Technical reviews, conducted by software engineers, serve as a quality control mechanism aimed at identifying errors in software artifacts. Audits, led by SQA personnel, focus on ensuring consistent adherence to quality guidelines and standards throughout the software engineering process.

5.1.3.2 Testing (see 5.1.2)

SQA oversees the planning and execution of software testing activities to achieve the primary goal of error detection. This involves ensuring that testing strategies are properly formulated, test cases are comprehensive, tests cover a wide range of code, and testing processes are efficient in uncovering defects across various software components.

5.1.3.3 Change Management, Error/Defect Collection and Analysis

Ensuring that robust change management practices are established and followed within the software development lifecycle is important. This involves overseeing procedures for requesting, evaluating, approving, and implementing changes to software requirements, designs, and code while maintaining traceability and minimizing risks associated with changes.

SQA systematically collects and analyses error and defect data to gain insights into how errors manifest in software artifacts and what software engineering practices are most effective in eliminating them. This data-driven approach helps in refining development processes to mitigate future occurrences of similar issues.

5.1.3.4 Security and Risk Management

SQA has an important role in ensuring that suitable processes and technologies are used to effectively address software security concerns. This involves implementing security standards, conducting security assessments, and integrating security practices into the software development lifecycle to protect against potential vulnerabilities and threats.

In situations where software failure could pose safety risks, SQA may be responsible for evaluating the impact of such failures and taking necessary steps to reduce risks. This task may involve evaluating safety-critical systems, identifying failure modes, and implementing measures to enhance the reliability and resilience of the software under consideration.

Risk management is also overseen by SQA to identify, assess, and mitigate potential risks within the software development process. This involves ensuring that risk identification techniques are applied, conducting comprehensive risk analysis, and devising contingency plans to effectively address identified risks. This enhances the overall resilience of the software product.

5.2 Solution Approaches

This section aims to compare various options for deploying and testing web applications objectively. The goal is to identify the most suitable solutions for the project's needs. The goal is to narrow down the available options based on their alignment with project requirements.

5.2.1 Deployment

Starting with deployment options, including cloud-based platforms and traditional server setups, each option is compared against the following factors: Performance/Scalability, Ease of Use, Cost, Flexibility/Compliance, Availability and Content Delivery Network (CDN).

5.2.1.1 Deployment Services

5.2.1.1.1 Netlify

Netlify is a modern hosting and automation platform designed for web projects. It excels in continuous deployment, allowing automatic builds and deployments directly from a Git repository. Netlify also offers features such as serverless functionality and form handling.

Pros

- Offers automatic builds and deployments from Git repositories.
- Supports serverless architecture for backend logic.
- Provides a global content delivery network for optimal performance.

Cons

- Complex/High Pricing
- For more complex projects, additional build configuration may be needed.

5.2.1.1.2 S3/Cloudfront

Amazon S3 (Simple Storage Service) and CloudFront provide a powerful combination for hosting and delivering web content. S3 stores the static assets while CloudFront acts as the content delivery network, ensuring low latency and high availability.

Pros

- Provides high scalability and reliability for static content
- Cost-Efficient

Cons

- Complex Setup
- Primarily designed for static content, requires additional services for dynamic applications

5.2.1.1.3 Vercel

Vercel is a cloud platform designed to streamline the process of deploying, hosting, and managing web applications. Originally created to support Next.js, a popular React framework for building web applications, Vercel has evolved into a versatile platform that supports a wide range of frontend frameworks, backend technologies, and static site generators.

Pros

- Tailored specifically for frontend development and hosting, making it a good option for react-based web applications
- Provides automatic deployments from Git repositories with minimal configuration
- Global CDN: Ensures fast and reliable content delivery worldwide

Cons

- Limited Backend Functionality
- Limited Database Options

5.2.1.1.3.1 Deployment Process via Vercel

Vercel is a platform that enables the effortless deployment of web applications. This chapter details this process of installing a new web application with Vercel.

The initial stage of deploying a web application involves choosing a Git repository. Vercel enables automatic deployments for every branch push and merge onto the Production Branch (main) of GitHub.

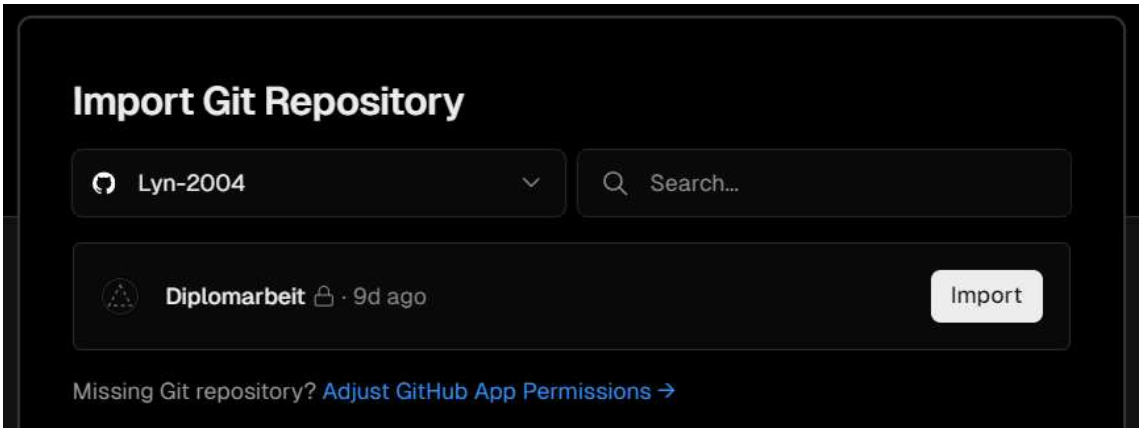


Figure 5-3: Import Git Repository

The 'Configure Project' window (Figure 5-4) will appear next. This project has to change the root directory of the deployment, as in the Git repository are both 'admin' and 'front' web applications (Figure 5-6).

After configuring and connecting the domain, your web application will be accessible on the internet. You can manage and access all deployments easily through the Vercel Dashboard. This process was repeated twice for this project, once for the admin web app and once for the actual site (Figure 5-5).

Furthermore, Vercel offers a range of analytics for each project, such as the number of requests made to each URL over a period of time (Figure 5-7).

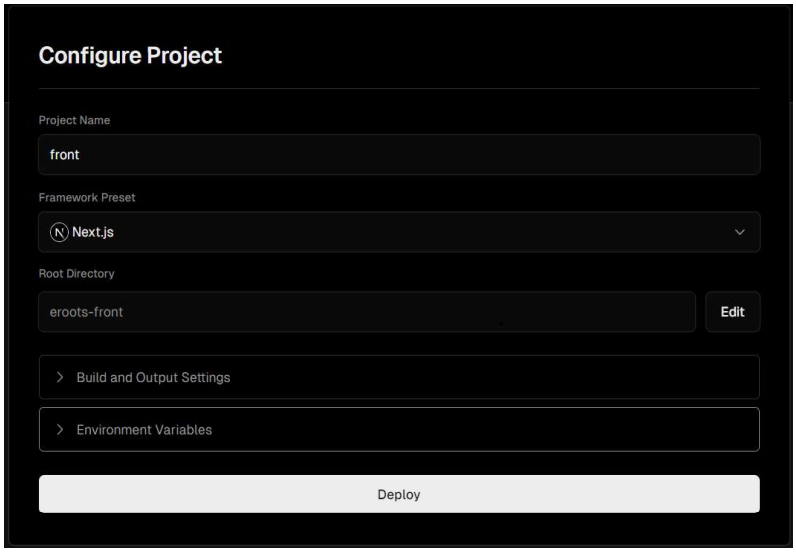


Figure 5-4: Configure Project window

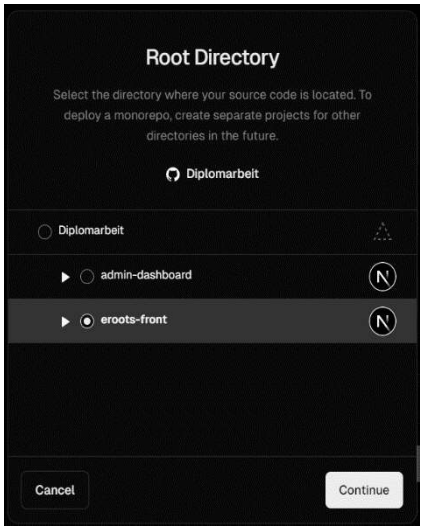


Figure 5-6: Select Root Directory

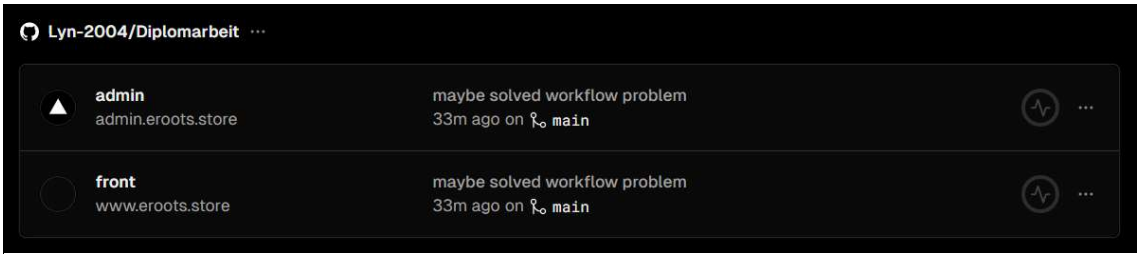


Figure 5-5: Vercel Dashboard

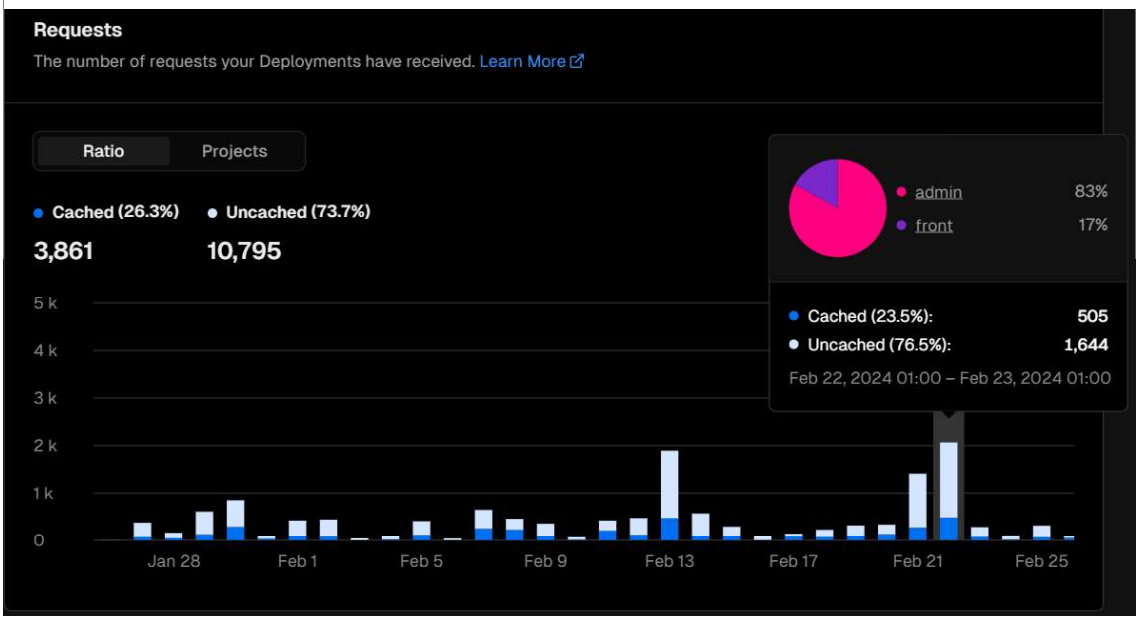


Figure 5-7: Vercel Analytics, here Request Amount

5.2.1.2 Evaluation of Deployment Solutions

To evaluate the most suitable deployment service for the project, a cost benefit analysis is conducted. The aim is to make an informed decision that maximises value and aligns with the project objectives.

Points will be distributed on a scale of 1 to 5, with higher scores indicating better performance. This holds true for the weights as well.

5.2.1.2.1 Criteria

Performance/Scalability

This criterion assesses the solution's capacity to manage growing workloads while maintaining optimal performance. A scalable and high-performance solution guarantees that the web application can handle expansion without compromising speed or reliability.

Ease of Use

Ease of use evaluates the user-friendliness and intuitiveness of deployment and testing solutions. A solution that is easy to use simplifies processes, reduces the learning curve, and increases productivity for developers and stakeholders.

Cost

Cost refers to the financial implications of implementing deployment solutions, including initial setup costs, ongoing maintenance expenses, and any associated licensing fees. A cost-effective solution provides the desired features and benefits without exceeding budgetary constraints.

Flexibility/Compliance

Flexibility assesses the adaptability of the solution to meet various project requirements and compliance standards. A flexible solution can accommodate changes in project scope or regulatory requirements while adhering to industry standards and regulations.

Availability

Availability assesses the reliability and uptime of the solution. A highly available solution minimises downtime and ensures uninterrupted access to the web application for users. This criterion considers factors such as redundancy, failover mechanisms, and disaster recovery capabilities.

Content Delivery Network (CDN)

A Content Delivery Network (CDN) is a network of servers distributed across various geographical locations. Its primary purpose is to deliver web content, such as images, videos, scripts, and other static or dynamic assets, to users more efficiently and reliably. The CDN criterion evaluates whether the solution integrates seamlessly with a CDN to optimize content delivery and improve website performance. A solution that enables CDN integration can improve the speed, reliability, and global reach of the web application by caching content closer to end-users.

5.2.1.2.2 Cost-Benefit Analysis

Criteria	Weight	Netlify	Points	Vercel	Points	S3/CloudFront	Points
Performance/Scalability	5	4	20	5	25	4	20
User-friendliness	3	4	12	5	15	2	6
Cost	4	2	8	4	16	5	20
Flexibility/Compliance	5	4	20	4	20	4	20
Availability	4	5	20	5	20	5	20
CDN	3	4	12	5	15	5	15
Total		**76,67%**		**93%**		**84,17%**	

Rating		*simplified
Weight	1 minor ↔ crucial 5	
Points	1 bad ↔ excellent 5	

Figure 5-8: Cost-Benefit Analysis for Deployment Services

After conducting the analysis, it was found that Vercel was the preferable deployment service among Netlify, Vercel and S3/CloudFront, due to the efficient Content Delivery Network and good Performance, it is able to function effectively for kenian users.

5.2.2 Testing

5.2.2.1 Software testing services

5.2.2.1.1 Protractor

Protractor is an end-to-end testing framework designed for testing AngularJS and Angular applications. It utilizes Selenium WebDriver to automate interactions with web browsers. Protractor offers a testing environment tailored to Angular applications, providing built-in support for Angular-specific features such as synchronization, locator strategies, and automatic waiting for asynchronous tasks to complete.

Protractor enables developers to write tests in JavaScript or TypeScript using familiar testing syntax, such as Jasmine or Mocha. This makes it easy to write and maintain test suites for Angular applications.

Protractor's comprehensive API and integration with Angular's testing utilities enable developers to write efficient and reliable end-to-end tests that validate the functionality, performance, and user experience of their Angular applications.

Pros

- Asynchronous Execution
- Built-in Waits
- Good available Documentation

Cons

- Angular-Centric
- Limited Browser Support

5.2.2.1.2 Jest

Jest is a widely used JavaScript testing framework known for its simplicity, speed, and powerful features. It is used for testing JavaScript code, including frontend applications,

backend services, and Node.js modules. It includes a test runner, assertion library, and mocking capabilities, making additional setup or configuration unnecessary.

Jest enables developers to concentrate on writing tests rather than managing testing infrastructure with its zero-configuration setup. Its intuitive API and built-in features, including snapshot testing, automatic mocking, and parallel test execution, streamline the testing process and make it simple to write and maintain tests. Furthermore, Jest provides comprehensive code coverage reporting, allowing developers to identify untested code and ensure adequate test coverage. Overall, Jest is a versatile and developer-friendly testing framework that enables developers to write efficient and reliable tests for their JavaScript applications.

Pros

- Performance
- Snapshot Testing helps in detecting unexpected changes in UI components.
- Good Documentation

Cons

- Good Compatibility with JavaScript
- Less Focus on Browser Interaction, more coding

5.2.2.1.3 Cypress

Cypress is a contemporary end-to-end testing framework that prioritises developer experience and fast, reliable tests. Unlike traditional testing frameworks, Cypress runs directly in the browser. This allows developers to observe tests in real-time, debug failures, and interact with the application under test using a built-in interactive test runner.

Cypress offers a wide range of APIs and utilities for test writing. These include commands for interacting with DOM elements, making network requests, and handling asynchronous tasks. The Service simplifies the process of writing and maintaining tests with its automatic waiting, intelligent retries, and built-in assertions. This results in faster test development and more resilient tests.

Furthermore, Cypress provides a range of features including a time-traveling debugger, snapshots, and video recording of test runs. These features facilitate the diagnosis and troubleshooting of test failures. In summary, Cypress enables developers to create effective and dependable tests that verify the functionality, performance, and user experience of their web applications.

Pros

- Performance
- Real-Time Browser Interaction
- Test Replays and Snapshots

Cons

- Limited Browser Support

5.2.2.1.3.1 Testing process using Cypress

5.2.2.1.3.1.1 Installation

To add Cypress as the testing software to the project, run

“npm install cypress --save-dev” in the command line, and it is ready to go.

To check if Cypress is installed, examine the 'package.json' file and verify that 'cypress' is listed under the 'DevDependencies' tab.

There are two ways to open the Cypress software:

- running “npx cypress open” directly
- adding the line ‘"cypress": "cypress open" ’ to the 'scripts' section of the 'package.json' file, and then executing “npm run cypress” in the command line.

he second option may be more convenient because it resembles the command to start the web application, which is “npm run dev”. However, for this project, the first option was chosen.

```
{
        "name": "eroots-front",
        "version": "0.1.0",
        "private":true,
        "scripts": {
        },
        "dependecies": {},
        "devDependencies":{
                "cypress": "^13.3.3"
        }
}
```

Figure 5-9: package.json including only Cypress as a Dependency

5.2.2.1.3.1.2 Structure

When Cypress is installed for the first time, a 'cypress' folder is automatically created in the root project folder with the following subfolders

- ‘/downloads’: files downloaded during a test are stored here
- ‘/fixtures’: contains fixture files
- ‘/screenshots’: this is where Cypress saves screenshots of failed tests or the “cy.screenshot()” command
- ‘/component’ and ‘/e2e’: subfolders holding the tests (specs) are
- ‘/support’: Contains customizable configuration files

Some later added, additional folders may be:

- ‘/videos’: Videos recorded during a cypress run are stored here
- ‘/integration’: ‘component’ and ’e2e’ subfolder are moved here to simplify Test automation

5.2.2.1.3.1.3 Configuration

Cypress Launcher

When Cypress is opened, the Lauchpad appears. Here it prompts you to choose whether you want to test e2e or component tests.

Once the preferred test type has been selected (here E2E Testing), it asks you which browser to use.

After selecting the preferred browser to use (here Chrome), Cypress opens an automated test browser where all the specs (tests) are displayed.

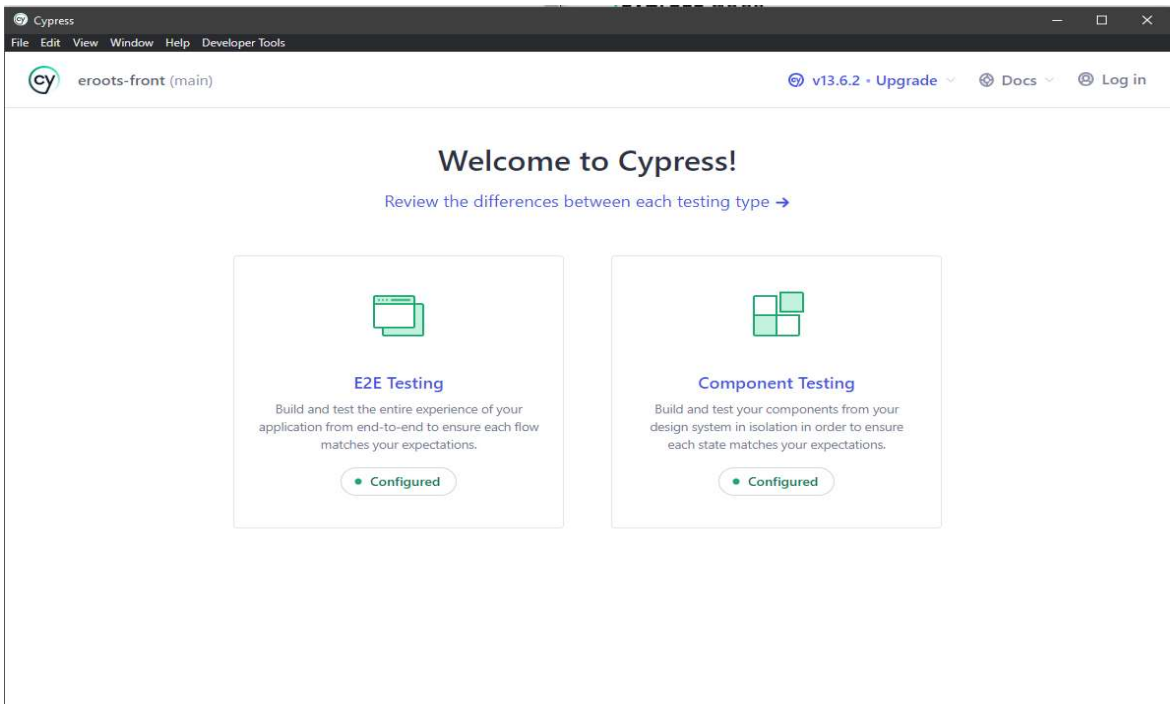


Figure 5-10: Cypress Launcher

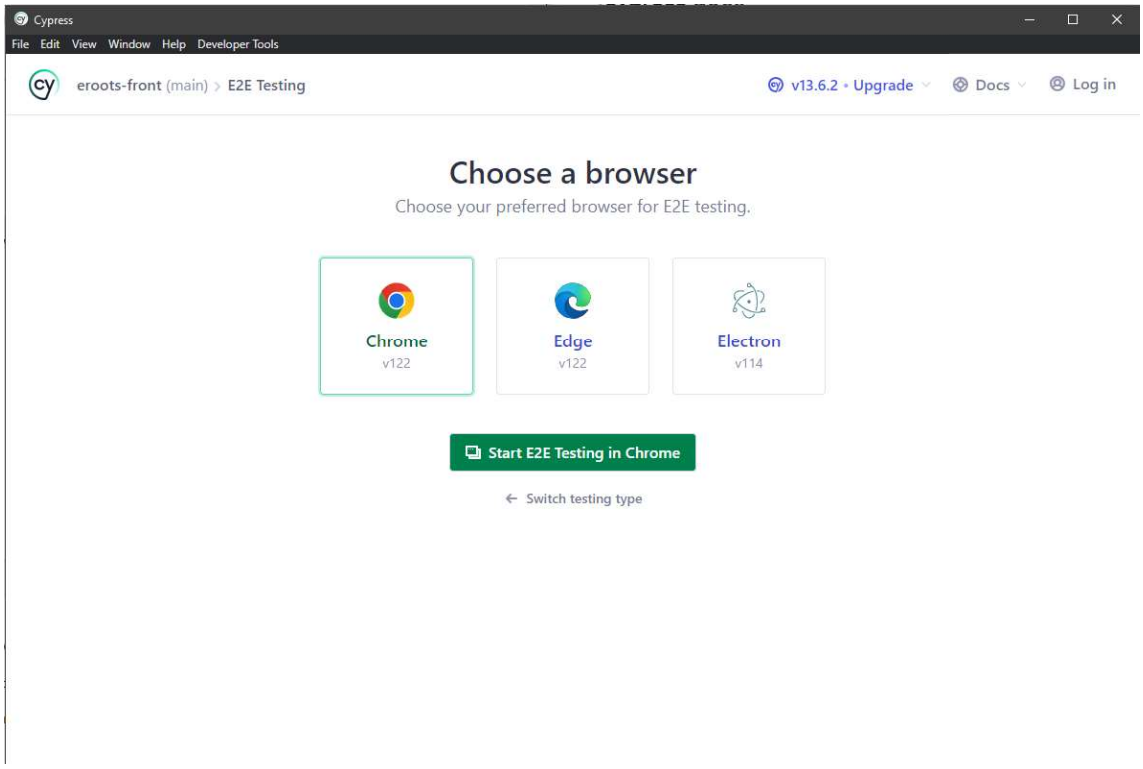


Figure 5-11: Cypress Launcher: Broswer Selection

Configuration file

S electing one of the two Test Types when neither has been implemented will automatically take you to the Configuration window. This will display the necessary configuration details for the selected type.

After configuring both E2E and Component testing, the configuration file ('cypress.config.js') should resemble the following (Figure 5-12):

```
module.exports = defineConfig({
  projectId: "2hvn4x",

  e2e: {
    setupNodeEvents(on, config) {
    },
    baseUrl: 'http://localhost:3000',
    specPattern: "cypress/integration/e2e/**/*.cy.{js,jsx,ts,tsx}",
  },

  component: {
    devServer: {
      framework: "next",
      bundler: "webpack",
    },
    specPattern: "cypress/integration/component/**/*.cy.{js,jsx,ts,tsx}",
  },
});
```

Figure 5-12: Cypress config file

The base url is defined in the E2E bracket. It specifies the basic url that the test software will call when no other has been provided.

The “specPattern” specifies the location of the tests (specs) inside the cypress folder and their specific filename ending and filetype.

5.2.2.1.3.1.4 E2E Testing

Cypress offers an extensive and user-friendly interface for coding and executing end-to-end tests. It provides a developer-friendly experience with an intuitive API and interactive test runner. It includes auto-wait, time-travel for debugging, built-in assertions, and screenshot/video capture.

The code snippet below provides an example of an end-to-end test written in Cypress. The test first visits the home page, then selects the search box in the header and enters the letter 'b'. This is followed by verifying that the results returned are correct. The program should identify all products whose names contain the letter 'b', that is, 'black turtle beans' and 'white beans'. After that it follows the link of the first result and checks if it leads the user to the right product page.

- “cy.visit()”: visits any specified link. As the default address is set to ‘http://localhost:3000’ no additional text is needed
- “cy.get()”: selects one or more DOM elemtens by their selector
- “.type()”: types provided String in the selected input element

- ".contains()": verifies the text is included in the selected element

```
describe ('Searcher Test', () => {
 it('tests the SearchBox function located in the Header', () => {
  cy.visit('/cart')

  cy.get('.sc-jcMgDF').type('b')
  cy.get('[href="/product/64b13f569917efbf4c3e0054"] > .sc-hLBatu')
        .contains('Black turtle bean')

  cy.get('[href="/product/64b5050bd72e4ace84a5e665"] > .sc-hLBatu')
        .contains('White Beans')

  cy.get('[href="/product/64b13f569917efbf4c3e0054"] > .sc-hLBatu').click()

  cy.get('.id-__ColWrapper-sc-fae1be2-1 > :nth-child(2) > .Title-sc-c9cac02-0')
        .contains('Black turtle bean')
 })
})
```

Figure 5-13: E2E test

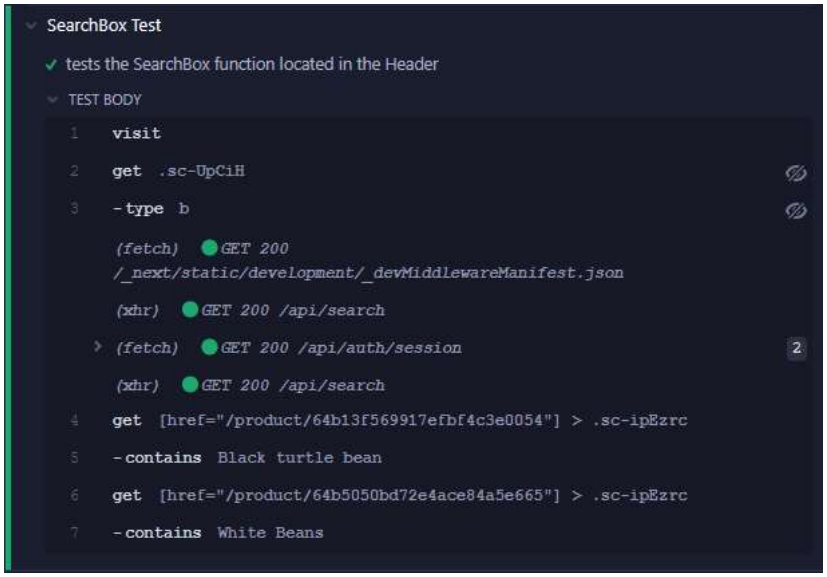


Figure 5-14: E2E Test log

Figure 5-15: Searchbar after test execution

Furthermore, while the test is being executed, Cypress generates a thorough report (Figure 5-15).

5.2.2.1.3.1.5 Component Testing

Cypress provides a lightweight and efficient solution for component testing. This enables developers to test individual components of their web applications in isolation, ensuring that they function correctly in different scenarios.

As a second assertion, the software verifies whether the API call returns the status code 200 OK.

```
describe('<SearchBox />', () => {
  it('renders', () => {
    cy.mount(<SearchBox />)

    cy.get('.sc-hLBatu').should('exist')
    cy.get('.sc-hLBatu').should('have.attr', 'placeholder', 'Search')

    cy.request("GET", "http://localhost:3000/api/search").then((response) => {
      expect(response.status).to.eq(200)
    })
  })
})
```

Figure 5-16: Component Test

- “cy.mount()”: generates the specified Component
- “.should()” & “expect()”: creates an assertion, althoug they have different syntaxes
- “cy.request()”: creates a request to a given url
- “.then()”: works with the subject selected/yielded from the previous command

Figure 5-17: Cypress log for component test

Cypress Cloud and Test runs

A Cypress run is a command that can be executed from either the terminal or the Cypress cloud. All available tests are run consecutively while logging important data. Test runs are typically used with continuous integration (CI).

In the Cypress Cloud, all data gathered during a test run, including information from external services such as GitHub Actions, is available in one central location. It records all executed tests and saves them for 30 days or longer, depending on the subscription.

5.2.2.2 Evaluation of Testing Solutions

5.2.2.2.1 Criteria

Performance/Speed

Performance and speed refer to the efficiency and responsiveness of the test service in executing tests. This criterion assesses how quickly tests can be run and completed, and the overall impact of performance on the development workflow. A test service that offers fast test execution times and minimal overhead contributes to faster feedback cycles and improved developer productivity.

Documentation

Documentation assesses the quality and completeness of the test service's documentation. This criterion evaluates whether the documentation provides clear, detailed instructions on how to use the test service, including setup instructions, API references, troubleshooting guides and best practices. Well-documented testing services enable developers to get up and running quickly and effectively use the service's features.

Flexibility/Compliance

Flexibility and compliance assesses the extent to which the testing service can adapt to different project requirements and comply with industry standards and regulations. This criterion considers factors such as support for different testing methodologies, customisation options. A flexible and compliant testing service meets diverse project needs and ensures that testing practices are in line with regulatory guidelines.

Tool Integration (CI, IDEs)

Tool integration evaluates the test service's compatibility and integration capabilities with other development tools and platforms, including continuous integration (CI) systems and integrated development environments (IDEs).

This point checks whether the testing service seamlessly integrates with popular CI tools such as Jenkins, Travis CI or GitHub Actions, and IDEs such as Visual Studio Code or IntelliJ IDEA. Strong tool integration allows developers to integrate testing into their existing workflows and toolchains, improving collaboration and efficiency.

Cost/Licensing

Cost and Licensing considers the financial implications of using the testing service, including upfront costs, ongoing costs and any associated licensing fees. This criterion evaluates whether the testing service offers affordable pricing plans or open source options that fit within the project's budget constraints. It also considers factors such as scalability and price transparency to ensure that the selected testing service offers good value for money.

Maintainability/Stability

Maintainability and Stability assesses the ease of maintaining and managing tests over time, as well as the overall stability and reliability of the testing service. This criterion evaluates factors such as test organisation, debugging capabilities and the frequency of updates and bug fixes. A maintainable and stable test service minimises the effort required to maintain tests and reduces the likelihood of test failures due to platform instability or compatibility issues.

5.2.2.2.2 Cost-Benefit Analysis

Criteria	Weight	Protractor	Points	Jest	Points	Cypress	Points
Performance/Speed	5	4	20	4	20	5	25
Documentation	5	4	20	4	20	5	25
Flexibility/Compliance	3	1	3	4	12	4	12
Tool Integration (CI, IDEs	4	4	16	4	16	4	16
Cost/Licensing	4	5	20	5	20	5	20
Maintainability/Stability	3	4	12	4	12	4	12
Total		**75,83%**		**83,33%**		**92%**	

Figure 5-18: Cost-Benefit Analysis for Testing Services

Based on performance, comprehensive documentation, flexibility, cost effectiveness and ease of use, Cypress emerged as the preferred choice of the three test services.

5.2.3 Quality Assurance

Due to the project's relatively small size, only code reviews were necessary as part of the Quality Assurance Process.

5.3 Results

5.3.1 Deployment

Upon completion of the deployment process, the site is available via the Internet at the following URL: https://www.eroots.store/, the same applies to the admin panel, which is available at: https://admin.eroots.store/. The process of deploying a webapp on vercel is discussed in chapter 1345.2.1.1.3.1 .

5.3.2 Testing

During the implementation of the tests for this project, a total of 9 were written, 3 component tests and 6 E2E tests. Additionally, a test automation system vis GitHub Actions has been added

At the top of every cypress test are two functions: “describe()” and it “it()”. The latter function contains the test code and it is inside the former one. As the names would suggest, these functions describe the basic goals of the test. “describe()” explains the whole test file, and “it()” describes only the following test code inside itself. In this way, more test cases can be written in the same file. However, these tests will not appear sperately in the cypress dashboard, but they will run in order in which it is written.

5.3.2.1 Component Tests

This component Test simply mounts a component, Input, and checks if it is present.

```
import React from 'react'
import Input from '../../../components/Input'

describe('<Input Component Test', () => {
  it('renders', () => {
    cy.mount(<Input />)
    cy.get('.Input__StyledInput-sc-3f4a8800-0').should('exist');
  });
})
```

Figure 5-19: Input Component Test

5.3.2.2 E2E Tests

The 2 following tests will show two very different functionalities of cypress E2E tests demonstrating the wide range of functions available to the tester.

5.3.2.2.1 Header Test

The test visits the homepage and sends a request for each link found in the header. The pass is only given if all links return the status code 200.

This test does not directly click the buttons in the header. Instead, it requests the specific URLs from the buttons, which reduces traffic and test time.

```
describe('Tests Home Page', () => {
  it('Tests if all the Buttons in the Head function correctly', () => {
    cy.request({ url: '/products', followRedirect: false, }).then((resp) => {
      expect(resp.status).to.eq(200)})

    cy.request({ url: '/categories', followRedirect: false, }).then((resp) => {
      expect(resp.status).to.eq(200)})

    cy.request({ url: '/cart', followRedirect: false, }).then((resp) => {
      expect(resp.status).to.eq(200)})

    cy.request({ url: '/dashboard', followRedirect: false, }).then((resp) => {
      expect(resp.status).to.eq(200)})
  })
})
```

Figure 5-20: Header E2E Test

5.3.2.2.2 Add-to-Cart function Test

The second end-to-end (E2E) test shown here visits the homepage, adds three products to the cart (the featured product, the same featured product via the product list, and a third product), and then navigates to the cart page to verify the presence of all products.

```
describe ('AddToCart', () => {
  it('Adds 3 products from the Home page to cart (2x BlackBeans, 1x
Watermelon), '+
  'goes to Cart page and checks if the products are in the cart with the
respective quantity', () => {
    cy.visit('/cart')
    cy.contains('Home').click()

    cy.get(':nth-child(4) > .sc-dJGLgI > .sc-hIPCWT > .sc-guDMob').click()
    cy.get(':nth-child(6) > .sc-dJGLgI > .sc-hIPCWT > .sc-guDMob').click()
    cy.get('.sc-hABBGs > .sc-guDMob').click()

    cy.contains('Cart (3)').click()
    cy.contains('Watermelon:')
    cy.contains('Black Beans:')

    cy.get(':nth-child(1) > :nth-child(2) > .sc-iBdoyZ > .sc-gLLuHO').contains(2)
    cy.get(':nth-child(2) > :nth-child(2) > .sc-iBdoyZ > .sc-gLLuHO').contains(1)
  })
})
```

Figure 5-21: E2E Test

This second example test runs the same way as if a human were to run through it, clicking all the buttons. It is instead an automated test.

5.3.2.3 Test Automation

For the test automation, GitHub Actions will be used, as this Service is the preferred tool for test automation due to its seamless integration with repositories, flexibility in defining workflows, extensive set of pre-built actions, scalability, visibility within the GitHub interface, automatic triggering based on events such as pull requests, and cost effectiveness.

To begin, it is necessary to create a new folder named '.github/workflows' and include a YAML file, such as 'main.yml', in the root directory. After that the Github Action is ready and will be initiated on every push.

```yaml
name: Cypress Tests

on: [push]

defaults:
  run:
    working-directory: ./eroots-front
jobs:
  install:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Cypress run
        uses: cypress-io/github-action@v4
        with:
          project: ./site
          browser: chrome
          build: yarn build
          start: npm run dev
          wait-on: "http://localhost:3000"
```

Figure 5-22: Github Actions .yml file

5.3.3 Additional Results

In the following chapters, two additional implemented functions will be presented. The Search Bar in the Header and the random Recommendations on every product page.

5.3.3.1 Search Bar

The Search Bar, also known as the SearchBox, provides a search function in the header, enabling users to easily find their preferred product from the available options.

To achieve this, the “/api/search” endpoint was created, which returns the ids and names of all products. The ids are used for linking the product page and the names for the search function itself.

This component is self-contained and requires no additional code or function when added to a page. In addition, the search bar is not case-sensitive.

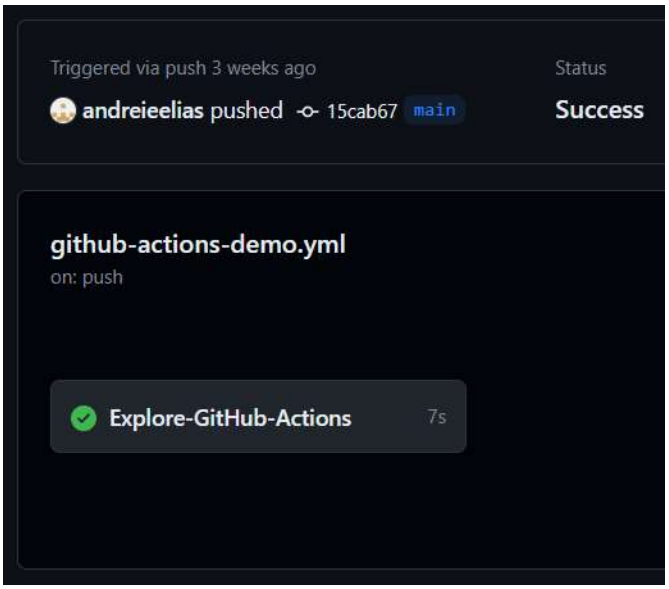


Figure 5-23: example for working Action in GitHub

Firstly, it requests important information from the previously mentioned endpoint. Then, on every change in the input (coded as 'Searcher'), it calls the 'filter' function within the 'productNames' array. The output of this function is an array of filtered names saved as the 'filteredProductNames' array.

If any results are available, the 'SearchResultHolder' component appears below the search bar, displaying a list of every applicable result with the corresponding link.

```
export const SearchBox = () => {
  const [searchQuery, setSearchQuery] = useState("");
  const [productNames, setProductNames] = useState([]);

  useEffect(() => {
    const fetchProductNames = async () => {
    const response = await axios.get("/api/search");
    setProductNames(response.data);
    };
    fetchProductNames();
  }, []);

  const filteredProductNames = productNames.filter(
    productName =>
      productName.titel &&
      productName.titel.toLowerCase().includes(searchQuery.toLowerCase())
  );

  const handleInputChange = e => {
    setSearchQuery(e.target.value);
  };

  return (
    <div style={{ position: "relative" }}>
      <Searcher
        type="text"
        placeholder="Search"

        value={searchQuery}
        onChange={handleInputChange}
      />
      {searchQuery && (
        <SearchResultHolder>
          <SearchResultList>
            {filteredProductNames.map(product => (
              <Link key={product._id}
                      href={"/product/" + product._id}>
                <SearchResult key={product._id}>
                      {product.titel}
                </SearchResult>
              </Link>
            ))}
          </SearchResultList>
        </SearchResultHolder>
      )}
    </div>
  );
};
```

Figure 5-24: SearchBox Component (simplified)

5.3.3.2 Random Recommendations

In contrast to the previously discussed function, the random recommendations panel is not a component but is rather coded into the product page itself.

To achieve this, all the products are retrieved at the initialization of the page using the "getServerSideProps" function (Figure 5-25), which connects to the database and collects the required elements.

```
export async function getServerSideProps(context) {
  await mongooseConnect();

  const { id } = context.query;
  const product = await Product.findById(id);
  const allProducts = await Product.find({}, null, { sort: { _id: -1 } });

  return {
    props: {
      allProducts: JSON.parse(JSON.stringify(allProducts)),
      product: JSON.parse(JSON.stringify(product)),
      initialRandomProducts: [],
    },
  }
}
```

Figure 5-25: getServerSideProps function (simplified)

If more than four products are available, a set of four products will be randomly selected and displayed in the 'ProductsGrid' panel located at the bottom of the page.

```
const newRandomProducts = [];
while (newRandomProducts.length < 4) {
      const randomIndex = Math.floor(Math.random() * allProducts.length);
      const randomProduct = allProducts[randomIndex];

      if (
        randomProduct._id !== product._id &&
        !newRandomProducts.includes(randomProduct)
      ) {
        newRandomProducts.push(randomProduct);
      }
}
setRandomProducts(newRandomProducts);
```

Figure 5-26: Random product selection function (simplified)

```
<Box>
        <Title>Recommendations</Title>
        <ProductsGrid products={randomProducts}></ProductsGrid>
</Box>
```

Figure 5-27: Recommendation panel implementation

The 'ProductsGrid' is a separate component that displays a number of products (see

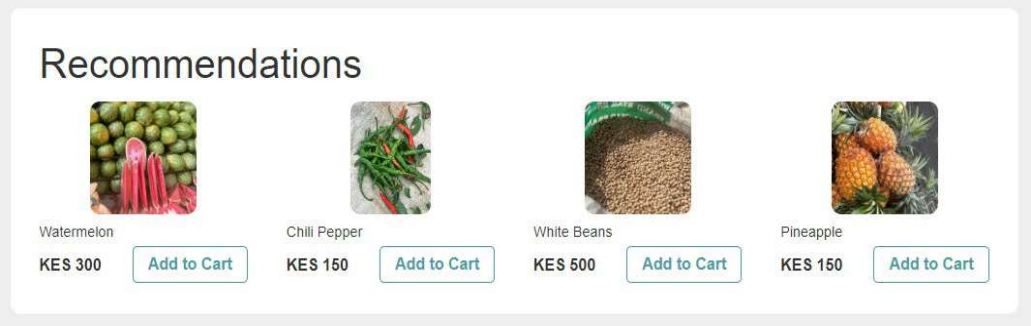


Figure 5-28: Recommendations

Figure 5-28).

Additionally, it is seen that, none of the selected products are the currently displayed product on the page.

5.3.4 Lessons Learned and Difficulties

Development of an React Web Application with using MongoDB as the Database

During the development of our React Web Application, we experienced several key lessons. Previously, there was no experience with MongoDB in any project, especially as it is a non-relational database.

Testing using Cypress

Testing the reliability and performance of our application using Cypress. Implementing tests that function well involves understanding the different types and functions that Cypress can provide.

A problem arose with the E2E tests due to the dynamic styled components used in this project, which changes its label or tag after almost every update. It was necessary to update these tests after every push to ensure their correct functionality.

Using GitHub in a long-term project and automating processes via GitHub Actions

However, the most significant challenge was encountered with GitHub Actions. Although it promised to automate project processes, configuring workflows proved to be complex. Overcoming these obstacles required extensive troubleshooting and a deeper understanding of GitHub Actions.

Due to the lack of documentation on the combination of Cypress and GitHub Actions, automating the tests proved to be a challenge. As our GitHub repository contains both the admin and front page projects, the 'package.json' file was not located in the root directory. This prevented GitHub from finding it and starting the test run. Very little of the online documentation and troubleshooting regarding this matter was useful, so it took a long time of experimenting to get it to work. The solution to this matter can be seen in Figure 5-22, where the “working-directory” tag is in the “defaults” bracket.

Furthermore, testing the workflow requires pushing the code onto GitHub, which can result in lengthy testing times even for minor changes.

5.4 Accompanying protocol according to § 9 para. 2 PrO

Protocol by Andrei Elias with date, type of activity (in the case of a meeting, reference to meeting minutes if necessary), effort in hours (to the nearest 5 minutes), aids used

DATE	EFFORT (h)	PHASE	TYPE OF ACTIVITY	AIDS USED
25.05.2023	01:15	Preliminary phase	Project application started to be filled out with the supervising teacher, considerations made regarding the structure and distribution of tasks, and external server for hosting the website (Netlify, Railway.app).	
31.05.2023	04:25	Preliminary phase	Project meeting 1: Which tool do we use for the project?	
31.05.2023	01:15	Preliminary phase	Project meeting 2: Who does what? What do we need/make at all?	
19.07.2023	00:20	Project management	Meeting: PTM notified that work can continue	
25.07.2023	01:20	Implementation	Frontend - latest products	VS Code
25.07.2023	01:05	Implementation	Frontend - latest products	VS Code
26.07.2023	02:15	Implementation	Frontend - latest products	VS Code
22.08.2023	01:10	Project management	Meeting with team about progress in the project	
06.09.2023	02:00	Implementation	Frontend - latest products, Product Grid, minor Error fixes	VS Code
07.09.2023	04:30	Implementation	finished latest products, minor Error fixes	VS Code
13.09.2023	01:50	School	Pre-study market analysis expanded	
19.09.2023	00:30	School	Meeting with Professor Schwarz	
23.09.2023	02:45	Implementation	Shopping Cart	VS Code
30.09.2023	01:45	Implementation	Bug fixes, finished Cart page and functions	VS Code
02.10.2023	01:40	School	Work on tasks due tomorrow	

03.10.2023	00:50	Preliminary phase	Meeting with Professor Schwarz about citation style for the thesis	
04.10.2023	03:50	Preliminary phase	PHB discussed and corrected	
10.10.2023	00:50	School	System architecture discussed with Professor Pirker, changes according to Eroots-4-Meeting Protocol-v1	
12.10.2023	03:45	Implementation	Checkout, Order Procedure	VS Code
14.10.2023	04:00	Project management	Pre-study, Cost-Benefit Analysis (10.1.1)	
15.10.2023	04:15	Testen	Created Cypress Test for Cart	
16.10.2023	05:15	Implementation	Implemented automatic tests with Cypress, pre-study	VS Code, Cypress Docs
17.10.2023	00:50	Project management	Continued working on pre-study	
17.10.2023	00:20	Project management	Meeting with Professor Schwarz and team	
19.10.2023	02:40	Preliminary phase	Work on pre-study presentation and pre-study	
21.10.2023	00:40	Preliminary phase	Pre-study presentation corrected with PTM	
25.10.2023	01:45	Testing	setting up CI	VS Code
30.10.2023	04:45	Implementation	Order Logic im Frontend, Succesful Order page	VS Code
31.10.2023	04:15	Implementation	Display all Orders in Admin Panel	VS Code
06.11.2023	01:40	School	Requirements specification + PHB	
10.11.2023	04:35	Implementation	general bug fixes and similar stuff	VS Code
13.11.2023	01:30	Project management	PHB: 1.9	
19.11.2023	02:30	Project management	PHB: 3.1, 3.5, 3.7, 4	
20.11.2023	02:30	Implementation	Requirements specification	

23.11.2023	02:15	Implementation	Deployment from the Admin page	VS Code
26.11.2023	01:50	Implementation	Research on Deployment	
28.11.2023	0:30	Project management	Meeting with Professor Schwarz	
29.11.2023	01:50	Implementation	Correcting requirements specification	
06.12.2023	1:50	School	Status report from 06.12.2023 in class	
07.12.2023	02:45	Implementation	Time and Date of Appointment in the Admin-Panel, bug fixes, Recommendations in seperate Product Pages	VS Code
08.12.2023	05:00	Implementation	Time and Date of Appointment in the Admin-Panel, bug fixes, Recommendations in seperate Product Pages	VS Code
14.12.2023	04:35	Implementation	Recommendations in seperate Product Pages	VS Code
29.12.2023	0:30	Project management	Project team meeting	
02.01.2024	09:00	Implementation	Random Product Recommendations, Order Conformation Email	VS Code
03.01.2024	06:15	Implementation	Random Product Recommendations	VS Code
04.01.2024	03:40	Implementation	Random Product Recommentdations done	VS Code
06.01.2024	02:00	Implementation	Random Product Recommentdations improved	VS Code
07.01.2024	00:45	Project management	Project meeting	
25.01.2024	06:55	Implementation	Produkt Search	VS Code
26.01.2024	06:40	Implementation	Produkt Search	VS Code
26.01.2024	2:00	Testing	Component Testing configured	
27.01.2024	04:30	Implementation	Product Search finished	VS Code
29.01.2024	00:15	Implementation	Product Search bar improved optically	VS Code

13.02.2024	4:00	Testing	Cypress E2E Test Env re-configurated, old Tests updated	VS Code, Cypress Docs
15.02.2024	4:00	Testing	SearchBox.cy.js and Header.cy.js Component Tests, E2E Tests updated	VS Code, Cypress Docs
16.02.2024	1:30	Testing	Header.cy.js Component Test refined	VS Code
17.02.2024	03:00	Thesis	Introduction, 5.1.2 Testing	
21.02.2024	03:00	Thesis	General: 5.1.1 Deployment, 5.1.3 Quality Assurance	
22.02.2024	03:30	Implementation	General Bug Fixes, Recovering Tests	VS Code
22.02.2024	00:50	Project management	Thesis meeting	
22.02.2024	02:00	Thesis	Solution Approaches: 5.2.1 Deployment	
23.02.2024	04:00	Testing	Automated Tests new Setup and Config using GitHub Actions and Cypress Cloud	VS Code
24.02.2024	01:50	Testing	Automated Test workflow improved	VS Code
24.02.2024	02:20	Testing	Automated Test workflow improved	VS Code
25.02.2024	05:45	Testing	Tests updated, added new tests	VS Code
25.02.2024	08:00	Thesis	Solution Approaches: 5.2.1 Deployment revised, 5.2.2 Testing, 5.2.3 Quality Assurance Results: 5.3.1 Deployment, 5.3.2 Testing	
26.02.2024	04:00	Testing	even more Tests written and Updated	VS Code, Cypress Docs
26.02.2024	02:00	Thesis	Results: 5.3.1 Deployment revised, 5.3.2 Testing revised	
27.02.2024	03:00	Thesis	General Revisions	
05.03.2024	00:20	Project management	Diplomarbeit mit Schwarz besprochen	
09.03.2024	04:00	Thesis	Improved 5.3 Results, changed code to Text boxes	

09.03.2024	03:00	Thesis	General Revisions, Improved 5.3.2 Testing
10.03.2024	04:00	Thesis	5.3.3 Additional Results, 5.3.4 Lessons Learned and Difficulties
12.03.2024	01:00	Thesis	Expanded 5.3.4, Improved 5.3.2.3
Total:	183:45h		

*Calculated total excludes time spent at school and on the preliminary study

6 Index of Figures

FIGURE 1-1 LOGO 3
FIGURE 1-2: CLIENT WEBAPPLICATION 3
FIGURE 1-3: LOGO 5
FIGURE 1-4: CLIENT WEBAPPLICATION 5
FIGURE 2-1: PRO AND CONS OF MONGODB ..... **FEHLER! TEXTMARKE NICHT DEFINIERT.**
FIGURE 2-2: PRO AND CONS OF MONGOOSE 15
FIGURE 2-3: A POSSIBLE NEXTJS APPLICATION WITH AWS : (LIENDO, MICHAEL, 2022) 15
FIGURE 2-4: PRO AND CONS OF GOOGLE OAUTH 16
FIGURE 2-5: EXAMPLE OF A CONNECTION FILE: (CHATGPT, 2024) 18
FIGURE 2-6: HOW TO INSTALL MONGOOSE. 19
FIGURE 2-7: HOW TO IMPORT MONGOOSE 19
FIGURE 2-8: EXPORTING THE DATABASE CONNECTION. 19
FIGURE 2-9: IMPORTING THE CONNECTION TO THE DATABASE 19
FIGURE 2-10: A POSSIBLE SCHEMA. (HALL, 2022) 20
FIGURE 2-11: INSTALL AWS IN NEXTJS. 21
FIGURE 2-12: NEXTJS WITH GOOGLE OAUTH API 22
FIGURE 2-13: REDIRECT URL. 22
FIGURE 2-14: GOOGLE PROVIDER 23
FIGURE 2-15: ONLY ADMIN EMAILS CAN LOGIN. 24
FIGURE 2-16: NEXTAUTH SECRET 24
FIGURE 2-17: PROVIDER IN NEXTAUTH. 24
FIGURE 2-18: CONNECTION TO THE DATABASE 25
FIGURE 2-19: CALLBACK IMPLEMENTATION 25
FIGURE 2-20: FUNCTION FOR ONLY ADMIN LOGIN 25
FIGURE 2-21: CHECKING IF THERE IS A CONNECTION TO MONGODB. 26
FIGURE 2-22: ENSURING THAT A DATABASE CONNECTION IS MAINTAINED. 26
FIGURE 2-23: FILTER IF THERE IS ALREADY A CONNECTION. 27
FIGURE 2-24: EXPORT THE MONGODB CONNECTION. 27
FIGURE 2-25: CONNECT TO MONGOOSE. 28
FIGURE 2-26: PRODUCT SCHEMA CODE 29
FIGURE 2-27: IMPORT STATEMENT CODE 29
FIGURE 2-28: TIMESTAMP 29
FIGURE 2-29: EXPORTING PRODUCT SCHEMA 29
FIGURE 2-30: IMPLEMENT CATAGORIES 30
FIGURE 2-31: ERROR MESSAGE 31
FIGURE 2-32: UPDATED BUCKET REQUEST. 32
FIGURE 2-33: UPDATED CORS REQUESTS. 32
FIGURE 2-34: SAVE PRODUKT PAGE 33
FIGURE 2-35: LISTS OF IMAGE 33
FIGURE 2-36: ERROR MESSAGE 34
FIGURE 2-37: SCRENNSHOT OF REDIRECT ULS IN THE GOOGLE CONSOLE. 35
FIGURE 2-38: ACCESS BLOCKED ERROR MESSAGE 35
FIGURE 3-1: IONOS WEBSITE FOR DOMAIN MANAGEMENT. 52
FIGURE 3-2: THE DOMAIN MANAGEMENT. 52
FIGURE 3-3: HOW THE REDIRECTION TO THE DOMAIN LOOKS LIKE. 53
FIGURE 3-4: HOW TO ACTIVATE THE SSL CERTIFICATE IN IONOS. 53
FIGURE 3-5: THE SSL CERTIFICATE ASSIGNED TO THE DOMAIN. 54
FIGURE 3-6: COMPLETION OF ACTIVATION OF THE SSL CERTIFICATE. 54
FIGURE 3-7: AFTER COMPLETION THE WEBSITE IS NOW SECURED. 55
FIGURE 3-8: ADMIN DASHBOARD. 56

FIGURE 3-9: CLIENT DASHBOARD. 57
FIGURE 3-10: JAVASCRIPT FUNCTIONS FOR MANAGING CATEGORIES 60
FIGURE 3-11: JAVASCRIPT FUNCTIONS FOR MANAGING CATEGORIES 61
FIGURE 3-12: ADMIN CATEGORY DASHBOARD. 62
FIGURE 3-13: PRODUCT CARD COMPONENT 64
FIGURE 3-14: FEATURED PRODUCT LAYOUT 65
FIGURE 3-15 ERROR MESSAGE: FAILED TO COMPILE 66
FIGURE 3-16 ERROR MESSAGE: YARN CANNOT BE INSTALLED 67
FIGURE 3-17 DEFINE PATH FOR RESPOSITORY 67
FIGURE 3-18 CREATE NEXT-APP 68
FIGURE 3-19 SUCCESS MESSAGE FOR CREATING 'EROOTS-FRONT' 68
FIGURE 3-20 SERVER ERROR 69
FIGURE 4-1: VOR- UND NACHTEILE VON JAVASCRIPT 85
FIGURE 4-2: VOR- UND NACHTEILE VON NODE.JS 86
FIGURE 4-3: VOR- UND NACHTEILE VON NEXTJS 87
FIGURE 4-4: VOR- UND NACHTEILE VON PYTHON 88
FIGURE 4-5: VOR- UND NACHTEILE VON JAVA 89
FIGURE 4-6: VOR- UND NACHTEILE VON RUBY 90
FIGURE 4-7: VOR- UND NACHTEILE VON PHP 91
FIGURE 4-8: VOR- UND NACHTEILE VON GO 92
FIGURE 4-9: VOR- UND NACHTEILE VON C# 93
FIGURE 4-10: FUNKTION ZUR VERBINDUNG DER DATENBANK 95
FIGURE 4-11 FUNKTION FÜR PRODUCTSPAGE 96
FIGURE 4-12: SCREENSHOT DER ALL-PRODUCT PAGE 97
FIGURE 4-13: FUNKTION ZUR VERBINDUNG DER SINGELPRODUCTPAGE 98
FIGURE 4-14: FUNKTION FÜR DIE SINGLEPRODUCTPAGE 99
FIGURE 4-15: HTML CODE FÜR DIE SINGLEPRODUCTPAGE 102
FIGURE 4-16: SCREENSHOT DER SINGLE-PRODUCT PAGE 102
FIGURE 4-17: FUNKTION ZUR PRODUCTSGRID 103
FIGURE 4-18: FUNKTION FÜR DEN GRAPHEN 106
FIGURE 4-19: BILD VOM GRAPHEN 107
FIGURE 4-20: SCREENSHOT PRIVACY POLICY GENERATOR TEIL 1 108
FIGURE 4-21: SCREENSHOT PRIVACY POLICY GENERATOR TEIL 2 109
FIGURE 4-22: SCREENSHOT PRIVACY POLICY GENERATOR TEIL 3 110
FIGURE 4-23: SCREENSHOT PRIVACY POLICY GENERATOR TEIL 4 110
FIGURE 4-24: EINGEFÜGTES IMPRESSUM IN DER WEB-APPLIKATION 111
FIGURE 4-25: FUNKTION FÜR DEN FOOTER 112
FIGURE 5-1: TYPES OF SOFTWARE TESTING ARRANGED (21.02.2024) 128
FIGURE 5-2: LEVELS OF TESTING 131
FIGURE 5-3: IMPORT GIT REPOSITORY 134
FIGURE 5-4: CONFIGURE PROJECT WINDOW 136
FIGURE 5-5: VERCEL DASHBOARD 136
FIGURE 5-6: SELECT ROOT DIRECTORY 136
FIGURE 5-7: VERCEL ANALYTICS, HERE REQUEST AMOUNT 136
FIGURE 5-8: COST-BENEFIT ANALYSIS FOR DEPLOYMENT SERVICES 138
FIGURE 5-9: PACKAGE.JSON INCLUDING ONLY CYPRESS AS A DEPENDENCY 140
FIGURE 5-10: CYPRESS LAUNCHER 142
FIGURE 5-11: CYPRESS LAUNCHER: BROSWER SELECTION 142
FIGURE 5-12: CYPRESS CONFIG FILE 143
FIGURE 5-13: E2E TEST 144
FIGURE 5-14: E2E TEST LOG 144
FIGURE 5-15: SEARCHBAR AFTER TEST EXECUTION 144
FIGURE 5-16: COMPONENT TEST 145

FIGURE 5-17: CYPRESS LOG FOR COMPONENT TEST 145
FIGURE 5-18: COST-BENEFIT ANALYSIS FOR TESTING SERVICES 147
FIGURE 5-19: INPUT COMPONENT TEST 148
FIGURE 5-20: HEADER E2E TEST 149
FIGURE 5-21: E2E TEST 149
FIGURE 5-22: GITHUB ACTIONS .YML FILE 150
FIGURE 5-23: EXAMPLE FOR WORKING ACTION IN GITHUB 150
FIGURE 5-24: SEARCHBOX COMPONENT (SIMPLIFIED) 151
FIGURE 5-25: GETSERVERSIDEPROPS FUNCTION (SIMPLIFIED) 152
FIGURE 5-26: RANDOM PRODUCT SELECTION FUNCTION (SIMPLIFIED) 152
FIGURE 5-27: RECOMMENDATION PANEL IMPLEMENTATION 152
FIGURE 5-28: RECOMMENDATIONS 153

7 Table of abbreviations

Abbreviations	Meaning
ACID	Atomicity, Concistency, isolation, Durability
API	Aplication Programming Interface
AWS	Amazon Web Servieces, a cloud computing platform
BSON	Binary JSON, a binaty-encoded serialation of JSON- like documents
CDN	Content Delivery Network
CI/CD	Continuous Integration / Continuous Deployment
CLI	Command-Line interface, a text-based for interacting with software
CORS	Cross-Origin Resource Sharing, a mechanism that allows resources on a web page to be requested from another domain outside the domain from which the resource originated.
CRUD	Create, Read, Update, Delete, basic operations for persistent storage.
CSS	Cascading Style Sheets, a style sheet language used for describing the presentation of a document.
E2E Test	End-to-End Test
HMAC	Hash-based Message Authentication Code, a mechanism for verifying the integrity and authenticity of a message.
HMR	Hot Module Replacement, a feature that updates modules in real-time without a full browser refresh.
HOC	Higher-Order Component, a function that takes a component and returns a new component.
IAM	Identity and Access Management, a framework for managing access to resources in a secure manner.
IDE	Integrated Development Environment
JSON	JavaScript Object Notation, a lightweight data interchange format.
JWT	JSON Web Token, a compact and self-contained way for securely transmitting information between parties as a JSON object.
MO365	Microsoft Office 365
OAuth	Open Authorization, a protocol for secure authorization.
ODM	Object-Document Mapping, a technique for converting data between incompatible type systems using object-oriented programming languages.
OIDC	OpenID Connect, an authentication layer on top of OAuth 2.0, an identity layer that allows clients to verify the identity of end-users based on the authentication performed by an authorization server.
QA	Quality Assurance, ensuring that the software meets specified requirements and standards.
RBDMS	Relational Database Management System
S3	Amazon Simple Storage Service, an object storage service offered by Amazon Web Services that provides industry-

	leading scalability, data availability, security, and performance.
SDK	Software Development Kit, a set of tools and libraries for developing applications.
SEO	Search Engine Optimization
SSR	Server-Side Rendering
UI	User Interface
URI	Uniform Resource Identifier, a string that identifies a resource.
URL	Uniform Recource Locator
(S)QA	(Software) Quality Assurance

8 List of sources and references

Amazon. (2024). *What is Amazon S3? - Amazon Simple Storage Service*. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

Baeldung. (2023, February 21). *What Is an API Endpoint? | Baeldung on Computer Science*. https://www.baeldung.com/cs/api-endpoints

Benmoussa, K., Laaziri, M., Larbi, K., & Yamami, A. (2019). A new model for the selection of web development frameworks: Application to PHP frameworks. *International Journal of Electrical and Computer Engineering*, *9*, 695–703. https://doi.org/10.11591/ijece.v9i1.pp695-703

Bootstrap, Otto, M., & Thornton, J. (2022, December 24). *Get started with Bootstrap*. https://getbootstrap.com/docs/5.3/getting-started/introduction/

Boyce, B. (2020, July 8). *Answer to 'Understanding what an HTTP endpoint is'*. Stack Overflow. https://stackoverflow.com/a/62788347

C#. (2023, November 10). *C# | Modern, open-source programming language for .NET*. Microsoft. https://dotnet.microsoft.com/en-us/languages/csharp

Cherian, J. (2018, July 10). *What are low-level APIs and high-level APIs in reactjs* [Forum post]. Stack Overflow. https://stackoverflow.com/q/51267465

Duvall, R. (n.d.). *What is Node.js? The JavaScript runtime explained | InfoWorld*.

Flanagan, D., & Matsumoto, Y. (2008). *The Ruby Programming Language: Everything You Need to Know*. O'Reilly Media, Inc.

GeekforGeeks. (2019, March 27). *Software Quality Assurance—Software Engineering*. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-software-quality-assurance/

GeekforGeeks. (2020, June 23). *Levels of Software Testing*. GeeksforGeeks. https://www.geeksforgeeks.org/levels-of-software-testing/

GeekforGeeks. (2021, May 11). *What is CI/CD?* GeeksforGeeks. https://www.geeksforgeeks.org/what-is-ci-cd/

GeekforGeeks. (2023, February 16). *Working with Monitoring and Logging Services*. GeeksforGeeks. https://www.geeksforgeeks.org/working-with-monitoring-and-logging-services/

Google. (2024a). *OAuth 2.0 für clientseitige Webanwendungen | Authorization*. Google for Developers. https://developers.google.com/identity/protocols/oauth2/javascript-implicit-flow?hl=de

Google. (2024b). *Verwenden von OAuth 2.0 für den Zugriff auf Google APIs | Authorization*. Google for Developers. https://developers.google.com/identity/protocols/oauth2?hl=de

Google | NextAuth.js. (2023, October 30). https://next-auth.js.org/providers/google

Graph. (n.d.). Retrieved 26 February 2024, from https://chat.openai.com/share/ac51f950-bd36-4d5e-884d-bbad880a6d43

Hall, J. (2022, April 7). *Getting Started with MongoDB & Mongoose | MongoDB*. https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/

Http-Endpoint. (n.d.). Retrieved 26 February 2024, from https://chat.openai.com/share/975e095b-b10f-42ae-8faa-3b4f654c92d2

Informatik-Verstehen. (2023). *Einführung in C# | C# Tutorial—C# lernen—C# Grundlagen*. Informatik - für Schüler und Studenten. https://www.informatik-verstehen.de/tutorials/csharp-tutorial/c-sharp-einfuehrung/

IONOS. (2021, May 27). *SSL-Zertifikat*. IONOS Digital Guide. https://www.ionos.de/digitalguide/websites/webseiten-erstellen/ssl-zertifikat/

IONOS. (2023a, July 5). *Python vs. Java: Ein Vergleich der beiden Programmiersprachen*. IONOS Digital Guide. https://www.ionos.at/digitalguide/websites/web-entwicklung/python-vs-java/

IONOS. (2023b, July 21). *Java vs. JavaScript: Die beiden Sprachen im Vergleich*. IONOS Digital Guide. https://www.ionos.at/digitalguide/websites/web-entwicklung/java-vs-javascript/

IONOS. (2023c, August 18). *Was ist PHP? Die Skriptsprache erklärt*. IONOS Digital Guide. https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-php/

Java. (2024). *What is Java and why do I need it?* Java. https://www.java.com/en/download/help/whatis_java.html

Kelly, S. (2016). What Is Python? In S. Kelly (Ed.), *Python, PyGame and Raspberry Pi Game Development* (pp. 3–5). Apress. https://doi.org/10.1007/978-1-4842-2517-2_2

Kesav, A. K., Kushagra. (2022, May 12). *How to Integrate MongoDB Into Your Next.js App | MongoDB*. https://www.mongodb.com/developer/languages/javascript/nextjs-with-mongodb/

Liendo, Michael. (2022, November 17). *Deploy a NextJS 13 application to Amplify with the AWS CDK | Front-End Web & Mobile*. https://aws.amazon.com/blogs/mobile/deploy-a-nextjs-13-application-to-amplify-with-the-aws-cdk/

Luo, L. (n.d.). *Software Testing Techniques*. School of Computer Science Carnegie Mellon University. Retrieved 20 February 2024, from https://ignite.org.pk/wp-content/uploads/2018/12/1388051766_rfp1_Software-testing-techniques.pdf

Matt, A. (2021, April 21). *How Next.js aims to simplify front-end development—TechRepublic*. https://web.archive.org/web/20200603082105/https://www.techrepublic.com/article/how-next-js-aims-to-simplify-front-end-development/

Matt, A., & Github. (2024). Next.js. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Next.js&oldid=1212159975

MaxSped. (2016, October 31). *What is the difference between a high-level and low-level Java API?* [Forum post]. Stack Overflow. https://stackoverflow.com/q/30897001

MongoDB. (2023). *MongoDB: Die Plattform Für Anwendungsdaten*. MongoDB. https://www.mongodb.com/de-de

MySQL, A. B. (2001). *MySQL*. http://box.cs.istu.ru/public/docs/other/_New/Books/Data/DB/MySQL/MySQL%20Bible.pdf

Next.js. (2024a, last updated). *Building Your Application: Styling | Next.js*. https://nextjs.org/docs/app/building-your-application/styling

Next.js. (2024). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Next.js&oldid=1206703432

Next.js. (2024b). *Next.js by Vercel—The React Framework*. https://nextjs.org/

Oamkuma, R. (2021, July 19). *Advantages and Disadvantages of Node.JS | Software Developer India*. https://www.software-developer-india.com/de/vor-und-nachteile-von-node-js/

OAuth | NextAuth.js. (2023, November 28). https://next-auth.js.org/configuration/providers/oauth

Oberm, M., & BSc. (October, 17). *A Monitoring Agent for Go (Golang)*. https://epub.jku.at/obvulihs/download/pdf/5672521?originalFilename=true

Olan, M. (2003). Unit testing: Test early, test often. *Journal of Computing Sciences in Colleges - JCSC*, *19*.

PHP. (2001, 2024). *PHP: What is PHP? - Manual*. https://www.php.net/manual/en/intro-whatis.php

Python. (2024, March 20). *Applications for Python*. Python.Org. https://www.python.org/about/apps/

Roscoe Will. (2024, January 23). *Email | NextAuth.js*. https://next-auth.js.org/providers/email

Ruby. (2023). *Getting Started with Rails*. Ruby on Rails Guides. https://guides.rubyonrails.org/getting_started.html

TailwindCSS.com. (2020, November 15). *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. https://tailwindcss.com/

Termsfeed. (2012, 2023). *Privacy Policy Generator—TermsFeed*. https://app.termsfeed.com/wizard/privacy-policy

Termsfeed. (2023, September 6). *Legal Pages*. TermsFeed. https://www.termsfeed.com/blog/legal-pages/

Togs, Eatthis86, & Weitere. (2023). Webanwendung. In *Wikipedia*. https://de.wikipedia.org/w/index.php?title=Webanwendung&oldid=240689195

9 Appendix

Preliminary

EVERGREEN ROOTS
VERSION 1.0
DATE: 23 10. 23

Project sponsor: Pennina Njuguna
Project supervisor: Marcel Schwarz
Project manager: Everlyn Njeri

Andrei, Braun, Paukner | Thesis| 25.03.2024/24

EVERGREEN ROOTS

Version number	Date	Changes	Creator
0.1	12.06.2023	Preliminary study created	Paukner Sarah Jessica
0.2	11.09.2023	Preliminary study started	Paukner Sarah Jessica
0.3	13.09.2023	Preliminary study changed according to the template	Paukner Sarah Jessica, Braun Valerian
0.4	14.09.2023	Preliminary study design changed, system architecture created and inserted, and change directory added, Project goals corrected, Project risks added, and current state vs target state added	Paukner Sarah Jessica, Braun Valerian, Njeri Everlyn, Andrei Elias
0.5	21.09.2023	System architecture corrected (4.3) and added	Paukner Sarah Jessica
0.6	03.10.2023	Personal topics = 7, 7.1, 7.2, 7.3, 7.4 Personal topic = 5, 5.1, 5.1.1, 5.1.2, 5.2, 5.3, 5.4, 5.5	Paukner Sarah Jessica, Njeri Everlyn
0.7	04.10.2023	Personal topics, Citation	Paukner Sarah Jessica, Njeri Everlyn
0.8	05.10.2023	11.1, 11.2, 11.2.1, 11.2.2, 11.2.3, 11.2.4, 11.2.5, 11.2.6, 11.2.7, 11.2.8, 11.2.9, 11.2.10, 11.2.1.1, 11.2.1.2, 11.2.2.1, 11.2.2.2, 11.2.3.1, 11.2.3.2, 7.5, 7.5.1 – 7.5.14	Braun Valerian, Andrei Elias, Paukner Sarah Jessica
0.9	09.10.2023	Project organigram added	Njeri Everlyn, Paukner Sarah Jessica
0.10	10.10.2023	11.2	Valerian Braun
0.11	11.10.2023	Improved citations, 7 and following, database models	Paukner Sarah Jessica, Braun Valerian, Njeri Everlyn
0.12	12.10.2023	Pros and Cons from 7.2.1 to 7.2.6.	Paukner Sarah Jessica
0.13	13.10.2024	Database management systems added	Njeri Everlyn
0.14	14.10.2023	12, 12.1 Conclusion	Valerian Braun
0.15	16.10.2023	Subdivision of point 8	Paukner Sarah Jessica
0.16	17.10.2023	Utility analysis, TOWS-Analysis and SWOT-Analysis, errors in document improved, system architecture improved	Paukner Sarah Jessica, Njeri Everlyn, Andrei Elias
0.17	18.10.2023	Title page updated	Everlyn Njeri, Paukner Sarah Jessica
0.18	19.10.2023	Looking for and correcting first person POV in the document, 8.4, 9 and descending	Paukner Sarah Jessica
0.19	20.10.2023	Database solution and authentication solution written and completed	Everlyn Njeri
0.20	22.10.2023	Individual topic “Security finished”, correction of the document, filling in blank spaces underneath titles and correcting grammatical and syntax errors.	Paukner Sarah Jessica, Braun Valerian, Njeri Everlyn, Andrei Elias
0.21	22.10.2023	Preliminary finished	Paukner Sarah Jessica, Everlyn Njeri
1.0	23.10.2023	Added captions	Paukner Sarah Jessica, Everlyn Njeri, Valerian Braun

Content

1 MANAGEMENT SUMMARY 5

1.1 PROJECT TEAM 5
1.2 PROJECT GOAL 6
1.3 PROJECT RISKS 6

2 SWOT AND TOWS ANALYSIS 7

2.1 SWOT ANALYSIS 7
2.1.1 Strengths: 7
2.1.2 Weaknesses: 7
2.1.3 Opportunity: 8
2.1.4 Threats: 8
2.2 TOWS-ANALYSIS 9

3 DESCRIPTION OF THE CURRENT AND TARGET STATE 10

3.1 CURRENT STATE 10
3.2 TARGET STATE 10

4 MARKET ANALYSIS 10

4.1 SOME EXAMPLES INCLUDE: 10

5 POSSIBLE SYSTEM ARCHITECTURE 11

5.1 TOMCAT AND BROWSER 11
5.1.1 Tomcat Server 11
5.1.2 Browser 12
5.2 REACT UND NODEJS 13
5.2.1 Frontend 13
5.2.2 Backend 14
5.2.3 Hosting 14

6 BASIC OF PERSONAL TOPIC – NJERI EVERLYN 15

6.1 WHAT IS DATA? 15
6.1.1 Structured Data 15
6.1.2 Unstructured Data 16
6.2 WHAT IS A DATABASE 16
6.2.1 Characteristics of a database 17
6.3 DATABASE MODELS 17
6.3.1 Relational databases 17
6.3.2 Object oriented databases 18
6.4 DATABASE MANAGEMENT SYSTEMS 18
6.4.1 Relational databases management system 19
6.4.2 NoSQL databases 20
6.4.3 NoSQL database management systems 21
6.5 WHAT IS AUTHENTICATION? 22
6.6 WHAT KIND OF AUTHENTICATION METHODS COULD BE USED 22
6.6.1 Password-based authentication 23
6.6.2 OAuth 24
6.6.3 Two-Faktor Authentication 24

7 SOLUTION VARIANTS AND EVALUATION OF THE PERSONAL TOPIC – NJERI EVERLYN 25

7.1 SOLUTION FOR THE DATABASE TYPES 25
7.1.1 How to connect Redis 25
7.1.2 How to connect MySQL 27
7.1.3 Utility analysis 30
7.1.4 Conclusion 30
7.2 SOLUTION FOR AUTHENTICATION METHODS 30
7.2.1 How to connect Amazon OAuth/ Amazon Login 30
7.2.2 How to connect Google OAuth/ Google Login 31
7.2.3 Utility analysis 33
7.2.4 Conclusion 33

8 BASICS OF THE PERSONAL TOPIC – PAUKNER SARAH JESSICA 34

8.1 GENERAL 34
8.2 FRAMEWORKS AND LIBRARIES 34
8.2.1 React 34
8.2.2 Angular 35
8.2.3 Vue.js 35
8.2.4 Ember.js 36
8.2.5 Svelte 37
8.2.6 Backbone.js 37
8.2.7 Aurelia 38
8.2.8 Meteor 38
8.2.9 Mithril 39
8.3 USER EXPERIENCE 40
8.3.1 Expectations vs. Product Qualities 40
8.3.2 Emotions Drive Experience 40
8.3.3 Expectations Are Shaped by Experience 40
8.3.4 Context Matters 40
8.3.5 Expectations Comprise Various Aspects 40
8.4 SECURITY 41
8.4.1 Up to Date 41
8.4.2 Encryption 41
8.4.3 VPN Usage 41
8.4.4 Security Testing 41
8.4.5 Penetration Tests 41
8.4.6 Comprehensive Security 41
8.5 WEB APPLICATION SECURITY 41
8.5.1 Involving Security Early 41
8.5.2 Addressing Architecture-Level Flaws 41
8.5.3 Providing Secure Equivalent Functionality 42
8.5.4 Consider the Impact of Security Flaws 42
8.5.5 Anticipate Exploitation 42
8.5.6 Mitigate Reputation Damage 42
8.5.7 Prioritize Security in Architecture 42

9 SOLUTION VARIANTS AND EVALUATION OF THE PERSONAL TOPIC – PAUKNER SARAH JESSICA 43

9.1 FRONTEND FRAMEWORKS 43
9.1.1 React 43
9.1.2 Angular 44
9.1.3 Vue.js 45
9.2 COST-BENEFIT ANALYSIS 45
9.3 CONCLUSION 46

10 BASICS ON PERSONAL TOPICS – ANDREI ELIAS 47

10.1 DEPLOYMENT 47
10.1.1 Environment Setup 47
10.1.2 Continuous Integration/Continuous Deployment (CI/CD) 47
10.1.3 Deployment Strategies 47
10.1.4 Monitoring and Logging 48
10.1.5 Scaling 48
10.1.6 WebApp Deployment Services 48
10.2 TESTING 52
10.2.1 Unit Testing 52
10.2.2 Integration Testing 52
10.2.3 End-to-End (E2E) Testing 53
10.2.4 Automated Testing 53
10.2.5 Testing Software 53
10.3 QUALITY ASSURANCE 54
10.3.1 Code Reviews 54
10.3.2 Static Code Analysis 54

11 SOLUTION VARIANTS AND EVALUATION OF THE PERSONAL TOPIC – ANDREI ELIAS 55

11.1 DEPLOYMENT COST-BENEFIT ANALYSIS 55
11.2 TESTING COST-BENEFIT ANALYSIS 56
11.2.1 Cypress 56

12 BASICS ON PERSONAL TOPICS – BRAUN VALERIAN 59

12.1 INTRODUCTION 59
12.2 SUGGESTED SOLUTION 59
12.2.1 JavaScript/Node.js 59
12.2.2 Python 60
12.2.3 Java 60
12.2.4 Ruby 60
12.2.5 PHP 61
12.2.6 Go (Golang) 61
12.2.7 C# 61
12.3 HTTP ENDPOINT 62
12.3.1 Examples 62

13 SOLUTION VARIANTS AND EVALUATION OF THE PERSONAL TOPIC – BRAUN VALERIAN 67

13.1 UTILITY ANALYSIS 67
13.1.1 Criteria 68
13.2 CONCLUSION FOR THE LANGUAGE 68

14 AFTERWORD 69

14.1 CURRENT PROTOTYPE 69
14.2 LITERATURE CITED 70

1 Management Summary

In Kiambu, Kenya a Women Self Help Group wants to improve their sales of locally farmed goods in order to expand their consumer base. To achieve this, they need a web-based platform that enables online ordering of their products, with payment to be made upon pick-up.

1.1 Project Team

Role	Name
Project sponsor	Njuguna Pennina
Project manager	Njeri Everlyn
Project team member	Braun Valerian
Project team member	Andrei Elias
Project team member	Paukner Sarah Jessica
Project supervisor	Schwarz Marcel
Project coach	Hell Florian

1.2 PROJECT GOAL

An E-Commerce web application with an admin dashboard to upload new products and a client dashboard to order products and make an appointment for the evaluation of the farmed good getting sold.

- Client interface
  - product page with all products created.
  - one product page created.
- User
  - user created and edited.
  - products ordered.
  - view previous orders with a history view.
  - implement a search functionality.
- Admin Interface
  - products with data persistence created, edited, and deleted.
  - categories with data persistence created, edited, and deleted.
  - appointments booked (reservations)
  - products, categories and admin and user login data in the database stored.

1.3 PROJECT RISKS

1. Risk of External Database Link Failure: The possibility that the external databases link may not function as intended, leading to potential data retrieval and synchronization issues within the project.

2. Deployment Risk for the Web Application: The risk associated with deploying the web application, including potential challenges related to server configuration, compatibility issues, and unforeseen obstacles that could impact the project's timeline.

3. Integration Risk Between Frontend and Backend: The risk that the integration process between the frontend and backend components may encounter difficulties, potentially causing delays, functionality gaps, or data synchronisation problems within the project.

2 SWOT and TOWS Analysis

SWOT and TOWS analyses help identify potential weaknesses and threats, enabling one to analyse and discover strengths and opportunities as well.

2.1 SWOT ANALYSIS

With the help of the Swot analysis, an analysis of the internal strengths and weaknesses as-well as external opportunities and threats of this project we're created.

2.1.1 Strengths:

- Good cooperation between the project team
- Trust in each other to deliver tasks well and on time.
- Unique market opportunity: The project addresses a specific demand in the Kenyan market, as there are few competitors offering similar services for locally grown products.
- Support for women's empowerment: By partnering with a women's self-help group, the project contributes to the economic empowerment of women in the region, which can be a strong selling point and garner community support.
- Low initial investments: The project doesn't require significant initial investments, which can lead to a faster return on capital.

2.1.2 Weaknesses:

- **Technical Challenges**
  - Developing an e-commerce platform with authentication and data persistence can be complex and susceptible to technical issues, especially in regions with limited technical infrastructure.
- **Dependency on External Databases**
  - The project identifies the risk of the connection to external databases failing, which could disrupt the entire system.
- **Integration Complexity**
  - Integrating frontend and backend components can be challenging and may lead to delays if not executed properly.

- **Dependency on External Databases**
  - The project identifies the risk of the connection to external databases failing, which could disrupt the entire system.
- **Integration Complexity**
  - Integrating frontend and backend components can be challenging and may lead to delays if not executed properly.

2.1.3 Opportunity:

- **Market Expansion**
  - Upon success, the project could expand its business model to other regions, generating additional revenue.
- **Service Diversification**
  - In the future, the project could offer additional services to diversify its sources of income.
- **Enhanced Customer Engagement**
  - Introducing reviews, wish lists, and newsletters could strengthen customer engagement and loyalty.
- **Partnerships**
  - The project could establish partnerships with other stakeholders in the agricultural value chain to expand its offerings.
- **Positive Social Impact**
  - Supporting the women's self-help group contributes to the social development and empowerment of women in Kenya.

2.1.4 Threats:

- **Competition**
  - The possibility of entry by global e-commerce giants like Amazon into the Kenyan market could lead to intense competition.
- **Technical Issues**
  - Technical glitches or outages could disrupt operations and jeopardize customer trust.
- **Security Risks**
  - Managing user data and payment transactions increases the risk of security breaches, which could undermine customer trust.
- **Regulatory Changes**
  - New regulations or legal issues related to e-commerce could impact the project.
- **Economic Uncertainty**
  - Economic fluctuations in Kenya could reduce the purchasing power of the target audience and jeopardise the project's success.

2.2 TOWS-ANALYSIS

The TOWS Matrix, also known as the TOWS Strategic Alternatives Matrix or SWOT/TOWS Analysis, is a strategic planning tool used by organizations to identify and evaluate their strategic options and formulate strategies. (cf. Weihrich, 1982) In this part of the document the project's TOWS-Matrix is shown.

TOWS		External analysis	
		Opportunity	Threat
Internal Analysis	Strength	• Good teamwork and market expansion The efficient teamwork allows the project to expand into other self-help groups and thus take advantage of additional opportunities. • Trust and punctuality and improved customer retention Trust and punctuality reinforce customer retention through the introduction of reviews, wish lists and newsletters.	• Good teamwork and competition Good teamwork can be used to compete against big e-commerce giants like Amazon. • Trust and punctuality and technical problems Trust can help to overcome technical problems and maintain customer confidence.
	Weakness	• Technical challenges and improved customer retention Despite technical challenges, customer retention can be improved by introducing ratings, wish lists and newsletters. • Dependency on external databases and market expansion Dependency on external databases could be reduced by expanding into other self-help groups, using more local solutions.	• Technical challenges and technical problems Due to poor infrastructure in the country and problems with connecting to the web application via deployment, technical problems could occur that jeopardise the sale. • Dependency on external databases and security risks Dependence on external databases increases the risk of security breaches

3 Description of the current and target state

This description is going to give an overview of the project's current and target state.

3.1 Current state

The project sponsor, a Kenyan Self-help Group, is currently operating within the confines of person-to-person sales, which has constrained their capacity to tap into a broader customer demographic. This limitation has, unfortunately, resulted in the emergence of a significant food waste challenge. The restricted reach of their current sales approach has led to excess or unsold food products, contributing to the larger issue of food waste in the region. Addressing this dual concern, expanding their customer outreach, and mitigating food waste, is a pivotal aspect of this project's mission.

3.2 Target state

The e-commerce web application boasts an intuitive admin dashboard for seamless product management, along with a customer dashboard that empowers clients to order products through appointment scheduling. The focused customer acquisition strategy is geared towards broadening the sponsor's customer base, with the overarching goal of making a substantial impact in the battle against food waste.

4 Market analysis

In other countries there are many providers that enable the service of ordering goods. In Kenya there aren't any/many.

4.1 Some examples include:

- JioMart
  - https://www.jiomart.com/c/groceries/fruits-vegetables/fresh-vegetables/229
  - It is an online shopping market for food, baby products and books in India. It has different filters to search for products.
- Fresh Produce Shoppe
  - https://www.freshproduceshoppe.com/
  - An online shopping service with a large variety of foods in India.
- Amazon
  - https://www.amazon.de/
  - A worldwide online shopping service it has a large variety on different kind of goods. It is also the most used online shopping service in the world.
- Shop Market Afrika
  - https://shopmarketafrica.com/
  - An African grocery online shop that focuses on African specialities and delivers its products abroad.
- E-mart.co.ke
  - https://e-mart.co.ke/
  - Kenian online shop for general goods. Claims it is Kenia's No. 1 online supermarket.

The examples mentioned represent significant players in various global regions, however the product in Kenia aims to tap into a market that is still in its.

5 Possible system architecture

Possible system architectures for the project could include:

5.1 TOMCAT AND BROWSER

The “Tomcat and Browser system” architecture is one of the most used system architectures in projects. Here is an example of how the system architecture works:

5.1.1 Tomcat Server

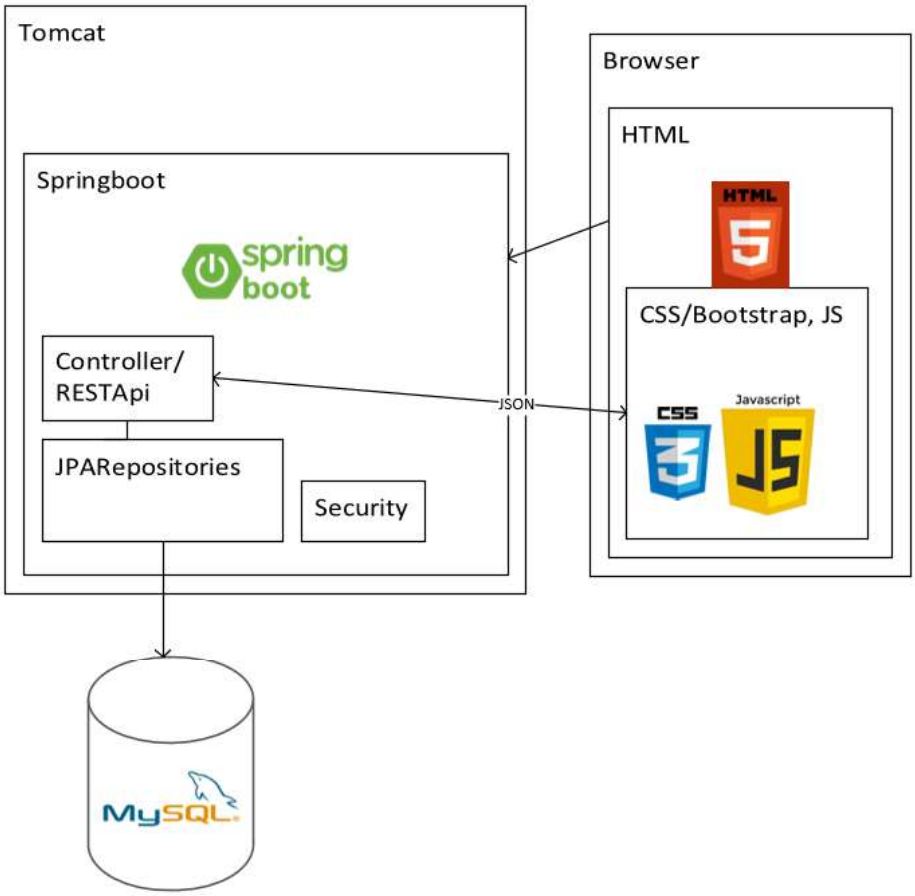


fig. 1 Tomcat/Browser system architecture (src.: Mr. Radlbauer's lesson about system architecture)

Tomcat is a web server and servlet container that serves as the runtime environment for a web application. It handles HTTP requests and manages the servlets and JSP (Java Server Pages) components of the application.

Spring Boot: Spring Boot is used as the application framework. It simplifies the configuration and development of Java applications, making it easier to create robust, production-ready applications. It provides the infrastructure for building web applications and managing application components.

Controller (Spring Boot): In the Spring Boot application, controllers are responsible for handling incoming HTTP requests, processing them, and returning appropriate responses. They serve as the entry point for a REST API.

REST API: The REST API is implemented using Spring Boot's capabilities for creating RESTful web services. It defines a set of endpoints that clients (e.g., browsers) can use to interact with the application. These endpoints are typically responsible for CRUD (Create, Read, Update, Delete) operations on resources stored in the database.

JPA Repository: Java Persistence API (JPA) is a specification for managing relational data in Java applications. The JPA repository in the architecture abstracts the interaction with the database. Spring Data JPA, a part of the Spring framework, makes it easier to work with JPA by providing a high-level, repository-based abstraction.

MySQL Database: MySQL is used as the database management system to store and manage the application's data. JPA and the repository layer interact with MySQL to perform database operations.

5.1.2 Browser

End-users access the web application through a web browser. The browser sends HTTP requests to the Tomcat server and receives HTML, CSS, JavaScript, and other resources in response.

HTML, CSS, Bootstrap, **and JS**: These are front-end technologies used to create the user interface (UI) of the web application. HTML defines the structure of the web pages, CSS and Bootstrap are used for styling and layout, and JavaScript is used for interactivity and enhancing the user experience. These resources are served by the Tomcat server and integrated into the HTML pages generated by the Spring Boot application.

5.2 REACT UND NODEJS

The project's system architecture uses Next.js' React and Node.js for its functionality:

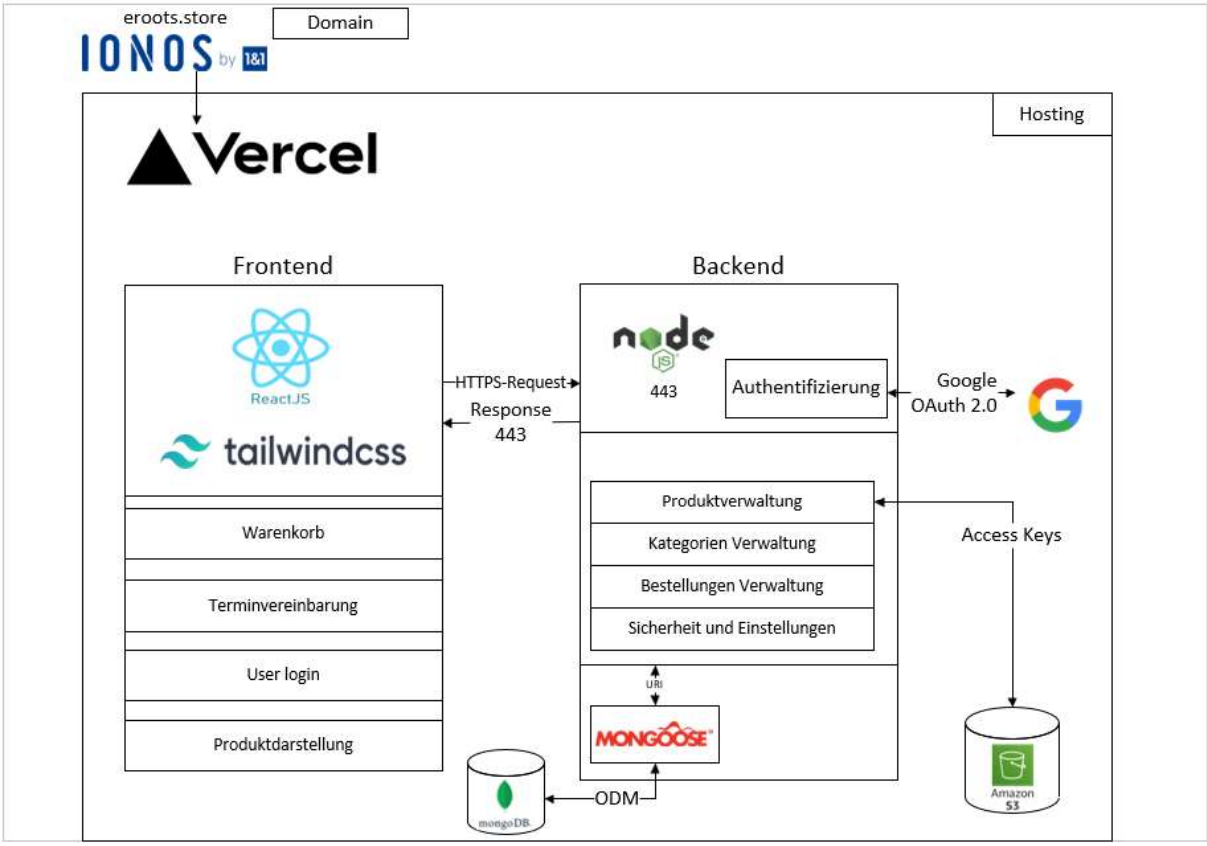


fig. 2 React/Node.js system architecture

5.2.1 Frontend

The application's user interface is the face of the operation, responsible for creating a user-friendly and visually appealing experience. To make this happen, **React.js** was picked as framework.

When it comes to styling and aesthetics, the application uses **Tailwind CSS**. **Tailwind** is a CSS framework that offers a ton of pre-designed classes, making it super easy to style HTML elements. This simplifies the styling process and contributes to an overall polished user experience.

5.2.2 Backend

Behind the scenes, the backend, built on the **Node.js** platform, takes care of handling data, logic, and communication with the database. **Node.js** provides a speedy runtime environment for server-side tasks, making it an excellent choice for building scalable and high-performance backend systems.

For managing the database, **Mongoose**, an Object Data Modeling (ODM) library tailored for **MongoDB** is used. **Mongoose** simplifies the interactions with the **MongoDB** database by introducing a schema-based modelling system, validation features, and a robust query-building interface.

The application's database of choice is **MongoDB**, a NoSQL database known for its flexibility and scalability. **MongoDB** stores data in a JSON-like format, making it perfect for accommodating dynamic data structures.

Connecting via an access key is one of the methods used to access the **AWS S3 bucket,** where the storage of images is securely provided for the product inventory and management. This access key serves as a means of authentication and authorization, allowing the application to interact with the cloud-based storage service and perform various tasks, such as uploading, retrieving, and managing product images.

The admin login uses **Google OAuth 2.0** to make the security safer and simpler. It lets administrators log in securely with their Google accounts, so there is no need to store passwords. Two-factor authentication can be used for extra security, and it works on different devices. This makes admin access easier and more reliable.

5.2.3 Hosting

The web application gets hosted with **Vercel**, a platform known for fast and reliable web deployment. The domain, "**eroots.store**", is managed by **IONOS**, a domain registrar. When someone enters the domain in the browser, **IONOS** directs them to the web application hosted on **Vercel**. This setup ensures that the site is easily accessible and well-maintained.

6 Basic of personal topic – Njeri Everlyn

"How can data persistence be planned and implemented in web applications with authentication?" is the question that Njeri Everlyn answers in this preliminary.

6.1 WHAT IS DATA?

Data refers to any collection or representation of facts, statistics, information, or symbols that can be processed or analysed by computers, humans, or other devices. It can take various forms, including numbers, text, images, sound, and more. Data can be structured, like a spreadsheet or a database, or unstructured, like a block of text in a document. (Regez et al., 2023)

6.1.1 Structured Data

Data that is organized in a specific format, such as databases with tables and columns, making it easy to search, retrieve, manipulate, and analyse. 1974 IBM developed structured query language (SQL) a programming language used to manage structured data. (cf. Education, 2021)

6.1.1.1 Pros

- Simple for machine learning (ML) programs: Structured data is well-organized and makes it easy for ML software to work with and ask questions about the data.(cf. Education, 2021)
- User-friendly for business folks: Structured data doesn't need a deep knowledge of various data types and how they work. If you have a basic understanding of the topic related to the data, you can easily use and understand it.(cf. Education, 2021)
- Works with a wide range of tools: Because structured data came before unstructured data, there are more tools out there that can be used to work with and analyse structured data.(cf. Education, 2021)
- Quick to locate: Structured data enables rapid and effective retrieval, sorting, and examination, making it a go-to tool for immediate access to information.(cf. Education, 2021)

6.1.1.2 Cons

- Limited versatility: Data with a fixed structure can only be used for its original purpose, which restricts how flexible and useful it can be.(Education, 2021)
- Limited storage flexibility: Structured data is typically stored in inflexible systems with strict rules (like "data warehouses"). This means that when data needs change, you have to spend a lot of time and resources updating all the structured data.(cf. Education, 2021)

6.1.2 Unstructured Data

Unlike structured data, which follows a set data model, unstructured data lacks a specific structure or format, making it more challenging to process automatically. To handle it effectively, non-relational (NoSQL) databases are a better choice. Examples include text documents, emails, social media posts, and multimedia files.(cf. Spitta & Bick, 2009, p. 168)

6.1.2.1 Pros

- Rich Information Source: Unstructured data can contain valuable, diverse information, including text, images, audio, and more, providing a rich source of insights. (cf. Spitta & Bick, 2009, p. 168)
- Real-World Relevance: Unstructured data often represents real-world conversations, feedback, and interactions, making it highly relevant for understanding customer sentiment and behaviour. (Dieterich, 2017, p. 16)
- Adaptability: Unstructured data can adapt to evolving information needs and capture unexpected insights without requiring extensive changes to data structures. (cf. Spitta & Bick, 2009, p. 168)
- Innovation and Discovery: Unstructured data can be a treasure trove for innovation and discovery, especially when exploring new trends, patterns, or opportunities. (cf. Spitta & Bick, 2009, p. 168)
- Competitive Advantage: Effectively harnessing unstructured data can provide a competitive edge by uncovering unique insights that others may overlook. (Dieterich, 2017, p. 16)

6.1.2.2 Cons

- Complex Processing: Due to their lack of structure, unstructured data is challenging to process. It often requires complex algorithms and tools to extract meaningful information from them. (Dieterich, 2017, p. 16)
- High Storage Requirements: Unstructured data, especially when it includes multimedia content like images and videos, can consume a significant amount of storage space.(Dieterich, 2017, p. 16)
- Complex Analysis: Analysing unstructured data is complex and often requires advanced technologies like Natural Language Processing (NLP) for text data or image recognition for multimedia content. (cf. Spitta & Bick, 2009, p. 168)
- Privacy and Security: Unstructured data can contain sensitive information, raising privacy and security concerns. Managing and protecting these data can be demanding.(Dieterich, 2017, p. 16)

6.2 WHAT IS A DATABASE

“Eine Datenbank ist eine Sammlung von Daten, die untereinander in einer logischen Beziehung stehen und von einem eigenen Datenbankverwaltungssystem (Database Management System, DBMS) verwaltet werden.” (*Datenbanken und SQL*, n.d., p. 3)

6.2.1 Characteristics of a database

- Data Integrity
  - Databases enforce rules and constraints to maintain the quality and consistency of data, preventing errors and ensuring accuracy. (cf. *Datenbanken und SQL*, n.d., p. 8)
- Concurrency Control
  - Databases handle multiple users or processes accessing and modifying data simultaneously, ensuring data consistency and preventing conflicts. (cf. *Datenbanken und SQL*, n.d., p. 9)
- Data Security
  - Database systems include security features to control access to data, protecting sensitive information from unauthorized users.(cf. *Datenbanken und SQL*, n.d., p. 8)
- Concurrency Control
  - Databases handle multiple users or processes accessing and modifying data simultaneously, ensuring data consistency and preventing conflicts.(cf. *Datenbanken und SQL*, n.d., p. 10)

6.3 DATABASE MODELS

At this time there are various types of database models. They describe how data is structured, how relationships are represented, and how data is manipulated. The difference between these models lies in their logical structure. These listed examples are the most used.

6.3.1 Relational databases

In a relational database, tables represent different data entities. These tables are connected through relationships, which are maintained using keys. These keys include primary keys (uniquely identify rows in a table) and foreign keys (linking data in one table to another). These relationships are stored within the tables themselves, making it easier to organize and retrieve data efficiently. (cf. Datenbanken und SQL, n.d., p. 12)

6.3.1.1 Pros

- Simple Changeability of the Database Structure: One of the notable strengths of relational databases is their flexibility. Tables, or relations, can be easily modified to accommodate new data fields or alter existing ones without disrupting the overall data structure. (cf. *Datenbanken und SQL*, n.d., p. 12)
- Mathematically Sound: Relational databases are built on a solid mathematical foundation, primarily based on set theory and relational algebra. This ensures data integrity and consistency, allowing for data manipulation and query operations. (cf. *Datenbanken und SQL*, n.d., p. 12)
- Easy to Program and Manage: The structured nature of relational databases simplifies the process of designing and managing data systems. (cf. *Datenbanken und SQL*, n.d., p. 12)

6.3.1.2 Cons

- Data Flow Complexity: Large, complex databases require careful handling of data for tasks like input, retrieval, and syncing. (cf. *Datenbanken und SQL*, n.d., p. 12)
- High Hardware Demands: Dealing with lots of data and complex queries can be tough on computer resources. It might need better hardware and efficient database design, which can be costly. (cf. *Datenbanken und SQL*, n.d., p. 12)
- Data Redundancies: Sometimes, the same data is stored in multiple places in a database. This can make queries faster but might cause problems with data accuracy if not managed and organized well. (cf. *Datenbanken und SQL*, n.d., p. 12)

6.3.2 Object oriented databases

Object-oriented databases consist of various types of objects, including people, departments, real and abstract objects. They are often considered an extension of relational databases but also incorporate object-oriented concepts like classes and inheritance. Object-relational databases represent a hybrid approach between object-oriented and relational databases, extending the latter toward object orientation. Despite their more complex structure, they enable a realistic representation of complex objects, particularly beneficial in technical and multimedia applications. Prominent examples of object-relational databases include Oracle, DB2, and PostgreSQL. (cf. *Datenbanken und SQL*, n.d., p. 13)

6.3.2.1 Pros

- Natural Object-Oriented Structure: Object-oriented databases offer a natural and realistic approach to data modelling. (cf. *Datenbanken und SQL*, n.d., p. 13)
- Versatility: They are highly versatile and well-suited for handling complex data structures. (cf. *Datenbanken und SQL*, n.d., p. 13)
- Partial Compatibility with Relational Databases: They can often be used in combination with relational databases, allowing for gradual migration. (cf. *Datenbanken und SQL*, n.d., p. 13)

6.3.2.2 Cons

- High Computational Demands: Processing extensive data and complex queries can strain computing potentially leading to increased costs for high-performance hardware.
- Learning Curve: Using object-oriented databases may require learning new skills and approaches, which can be a hurdle for some users.
- Complex Structure in Some Cases: The design and implementation of object-oriented databases can be more challenging due to their intricate structure.

6.4 DATABASE MANAGEMENT SYSTEMS

A DBMS is software that defines how a database system works. It's essential for creating, managing, and using a database. Once the DBMS is installed and configured, users can add and retrieve data. This is made possible through application-specific interfaces and a suitable database language. The most common ones are listed below.
(*Datenbankmanagementsystem (DBMS) erklärt*, 2020)

6.4.1 Relational databases management system

These databases use the relational model to store data in tables, where relationships between data are established using keys.

6.4.1.1 MySQL

MySQL is an open-source, high-performance relational database system widely used for storing and retrieving structured data. It's known for its speed, scalability, and ease of use. MySQL is popular in web applications and other software where efficient data management is essential. (cf. MySQL, 2001, p. 43)

6.4.1.1.1 Pros

- Speed: It is known for its fast data retrieval and processing, making it suitable for applications that require quick responses. (cf. MySQL, 2001, p. 40)
- Scalability: MySQL can handle both small and large databases, scaling to meet the needs of various applications. (*MySQL*, n.d.)
- Ease of Use: It has a relatively simple learning curve, with a wide range of documentation and community support available. (*MySQL*, n.d.)

6.4.1.1.2 Cons

- Resource Intensive: Large databases or complex queries may require substantial hardware resources. (cf. MySQL, 2001, p. 40)
- Lack of Some Enterprise Features: In comparison to certain enterprise databases, MySQL may lack specific features that large enterprises require, such as advanced auditing and data masking capabilities. (cf. MySQL, 2001, p. 40)
- Less Advanced Security: It may not offer the same level of security features as some other databases. (cf. MySQL, 2001, p. 40)

6.4.1.2 PostgreSQL

PostgreSQL is a free, and open-source database system that's has been used for managing complex data. It's been around since 1986 when it started at the University of California at Berkeley. (cf. *PostgreSQL: About*, n.d.)

6.4.1.2.1 Pros

- Scalability: PostgreSQL can handle both small and large datasets and is known for its scalability and ability to manage high-traffic applications. (cf. MySQL, 2001, p. 42)
- Data Integrity: It enforces data integrity through constraints, foreign keys, and robust transaction management, ensuring data accuracy and reliability. (cf. MySQL, 2001, p. 42)
- Active Community: PostgreSQL has a vibrant and active open-source community, which ensures regular updates, bug fixes, and security enhancements. (cf. MySQL, 2001, p. 42)

6.4.1.2.2 Cons

- Use Case Specificity: Depending on the application, PostgreSQL may lack certain advanced features or optimizations found in specialized databases. (cf. MySQL, 2001, p. 43)
- Specialized GIS: For certain geographic information system (GIS) applications, PostgreSQL's geospatial capabilities might not fully meet specialized requirements. (cf. MySQL, 2001, p. 43)
- Ecosystem Maturity: The PostgreSQL ecosystem, including third-party tools and extensions, may not be as mature as some commercial counterparts. (cf. MySQL, 2001, p. 43)

6.4.1.3 Oracle Database

Oracle Database is a highly respected and widely used relational database management system (RDBMS) developed by Oracle Corporation. It's widely used in various industries for data storage and management, offering different editions to suit diverse business needs. (cf. MySQL, 2001, p. 40)

6.4.1.3.1 Pros

- Security: Oracle provides comprehensive security features, including encryption, access controls, authentication, and auditing, to protect sensitive data. (cf. MySQL, 2001, p. 40)
- Reliability: It is known for its high availability and reliability, making it a preferred choice for mission-critical applications. (cf. MySQL, 2001, p. 40)
- Scalability: Oracle Database can scale from small to large enterprise-level systems, making it suitable for a wide range of applications. (cf. MySQL, 2001, p. 40)

6.4.1.3.2 Cons

- Cost: Oracle Database can be expensive, particularly for larger deployments, and may include licensing and support costs.
- Vendor Lock-In: The reliance on Oracle's ecosystem and proprietary technologies can lead to vendor lock-in, making it challenging to migrate to alternative solutions.

6.4.2 NoSQL databases

NoSQL, which means "not only SQL," is a database model that complements traditional relational databases rather than replacing them. It overcomes the limitations of relational systems by using alternative data models.(cf *NoSQL – Funktion und Vorteile von NoSQL-Datenbanken*, 2023)

6.4.2.1 Pros

- Wide Adoption: NoSQL databases are widely adopted for use in big data, IoT, and real-time analytics applications. (cf. *NoSQL – Funktion und Vorteile von NoSQL-Datenbanken*, 2023)

- Schema-less: NoSQL databases are typically schema-less, allowing for flexibility in data structure and easy modification as data requirements evolve. (cf. *NoSQL – Funktion und Vorteile von NoSQL-Datenbanken*, 2023)
- Flexible Data Models: They can accommodate various data types, making them suitable for unstructured or semi-structured data. (cf. *NoSQL – Funktion und Vorteile von NoSQL-Datenbanken*, 2023)

6.4.2.2 Cons

- Data Integrity: Without the rigid structure of SQL databases, data integrity and validation may need to be handled at the application level. (cf. *NoSQL – Funktion und Vorteile von NoSQL-Datenbanken*, 2023)
- Data Migration: Transitioning from traditional SQL databases to NoSQL databases or between different NoSQL databases can be challenging. (cf. *NoSQL – Funktion und Vorteile von NoSQL-Datenbanken*, 2023)
- Lack of ACID Properties: Many NoSQL databases sacrifice ACID (Atomicity, Consistency, Isolation, Durability) properties for performance, which may not be suitable for applications requiring strict data consistency. (cf. *NoSQL – Funktion und Vorteile von NoSQL-Datenbanken*, 2023)

6.4.3 NoSQL database management systems

6.4.3.1 MongoDB

MongoDB is a popular NoSQL database management system that is designed for flexibility, scalability, and high-performance data storage and retrieval. Unlike traditional relational databases, MongoDB is classified as a document-oriented database, and it stores data in a format known as BSON (Binary JSON), which allows for rich and flexible data structures. (cf. *MongoDB*, n.d.)

6.4.3.1.1 Pros

High Performance: MongoDB's document storage and indexing make it fast for read-heavy operations, especially when the data structure aligns well with the application's needs.

Flexible Schema: MongoDB uses a flexible document-oriented data model, which allows you to store data in a format that can change over time. This flexibility is beneficial for evolving applications where the schema may change frequently.

Scalability: MongoDB is designed for horizontal scalability. It can handle large amounts of data and high traffic by distributing data across multiple servers. This makes it well-suited for applications that need to scale quickly.

6.4.3.1.2 Cons

Lack of ACID Compliance: MongoDB sacrifices some of the ACID (Atomicity, Consistency, Isolation, Durability) properties for performance and scalability.

Learning Curve: Developers with a background in relational databases may face a learning curve when transitioning to MongoDB due to its unique data model and query language.

6.4.3.2 Redis

Redis, which stands for "Remote Dictionary Server," is an open-source, high-performance, and in-memory key-value data store.

6.4.3.2.1 Pros

High Performance: Redis stores data in-memory, which makes it extremely fast for read-heavy operations. It's well-suited for caching frequently accessed data, reducing the load on primary databases.

Data Structures: Redis supports a wide range of data structures, including strings, lists, sets, sorted sets, hashes, bitmaps, and more. This versatility allows developers to use Redis for a variety of use cases.

Persistence Options: Redis offers different persistence options. You can configure it to periodically save data to disk or use snapshotting. This allows data recovery in case of a system restart or failure.

6.4.3.2.2 Cons

Limited Storage: Since Redis is an in-memory database, its storage is limited by the available RAM. Storing large datasets may require more memory than some alternative databases.

Data Durability: While Redis provides options for data persistence, it's not as durable as disk-based databases. If the server crashes before data is saved to disk, data loss can occur.

Complexity: Redis can be complex to configure and manage in clustered or high-availability setups, especially for those new to the technology.

6.4.3.3 Amazon S3

6.5 WHAT IS AUTHENTICATION?

Authentication is the process of verifying the identity of a user, system, or entity attempting to access a computer system, network, application, or resource. It ensures that the entity claiming a particular identity is indeed who or what it purports to be. Authentication is a fundamental component of cybersecurity and is crucial for maintaining the security and integrity of digital systems. It is typically the first step in granting or denying access to protected resources.(cf. Syed Idrus et al., 2013, p. 2)

6.6 WHAT KIND OF AUTHENTICATION METHODS COULD BE USED

Authentication methods are techniques or mechanisms used to verify the identity of a user. These methods ensure that the entity claiming a particular identity is legitimate and authorized to access the requested resource. There are various authentication methods available, each with its own strengths, weaknesses, and use cases. Here are some common authentication methods. (cf. Syed Idrus et al., 2013, p. 5)

6.6.1 Password-based authentication

Password-based authentication is the most widespread method. It involves users creating passwords, which are stored securely. When users log in, the system checks if the entered password matches the stored one. (cf. *What Is Authentication?*, n.d.)

6.6.1.1 Pros

Low Cost: Implementing password-based authentication is relatively low cost compared to some other methods, as it often requires minimal infrastructure.

No Special Hardware: Users do not need additional hardware or software to use passwords, which makes it widely applicable.

6.6.1.2 Cons

Weak Passwords: Users often choose weak passwords that are easy to guess, which can lead to security vulnerabilities.

Password Reuse: Many users use the same passwords for multiple accounts, meaning that a breach of one account can compromise others.

6.6.1.3 Possible look

This is one possible way a password-based authentication could look.

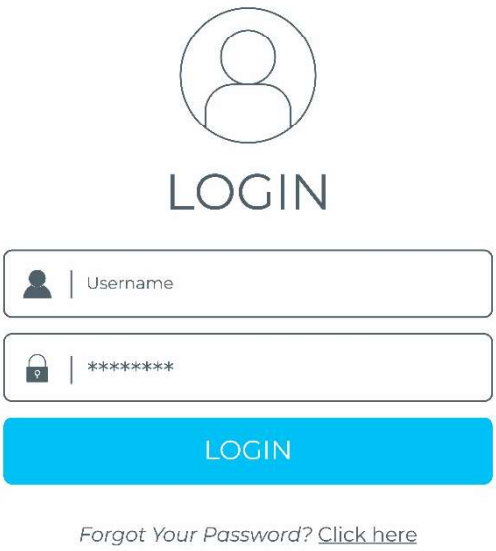


fig. 3(Download Login and Sign in User Interface. Business Website Modern Ui Template. for Free, n.d.)

6.6.2 OAuth

OAuth (Open Authorization) is an open standard and protocol used for delegated authorization and authentication. It allows one application to access resources on behalf of a user from another application or service without revealing the user's credentials, like their username and password. OAuth is commonly used for scenarios like single sign-on (SSO). (cf. Hardt, 2012, p. 1; cf. Leiba, 2012)

6.6.2.1 Pros

Enhanced Security: OAuth reduces the risk of exposing user credentials since the client never sees the user's password.

Widely Adopted: OAuth is widely adopted and supported by many service providers and applications.

6.6.2.2 Cons

Complexity: Implementing OAuth can be complex, especially for developers new to the protocol.

Reliance on Authorization Server: The security and reliability of the authorization server are critical; if it's compromised, it can affect the security of all connected applications.

6.6.2.3 Examples using OAuth.

6.6.2.3.1 Sign in with Google.

Many websites and applications offer "Sign in with Google" as an option. Users can log in using their Google account, which simplifies the registration and login process and often allows the app to access basic user profile information. (cf. *Verwenden von OAuth 2.0 für den Zugriff auf Google APIs | Authorization*, n.d.)

6.6.2.3.2 Amazon OAuth

Amazon provides OAuth for its users to log in to third-party websites and services using their Amazon credentials. (cf. *Authorization Code Grant | Login with Amazon*, n.d.)

6.6.2.3.3 GitHub OAuth

Many developer-focused services and applications use GitHub OAuth, allowing developers to sign in with their GitHub accounts. It simplifies access to code repositories and developer tools.(cf. *Authorizing OAuth Apps*, n.d.)

6.6.3 Two-Faktor Authentication

Two-Factor Authentication (2FA), also known as two-step verification or multi-factor authentication (MFA), is a security process that requires users to provide two different authentication factors to prove their identity. This method adds an extra layer of security compared to traditional single-factor authentication (like a password)(cf. Olanrewaju et al., 2021, p. 5)

6.6.3.1 Pros

Enhanced Security: 2FA significantly improves security by requiring not only something the user knows (like a password) but also something the user has (like a mobile device or security token). This makes it much harder for unauthorized individuals to gain access. (cf. Olanrewaju et al., 2021, p. 5)

Reduced Password Vulnerabilities: With 2FA in place, even if passwords are compromised, attackers cannot access accounts without the second factor. (cf. Olanrewaju et al., 2021, p. 5)

6.6.3.2 Cons

User Complexity: Some users may find 2FA to be an extra step that adds complexity to the login process, which can be perceived as an inconvenience. (cf. Olanrewaju et al., 2021, p. 5)

Loss of Second Factor: If the second factor is lost, damaged, or not readily accessible (e.g., a lost phone or token), users may have difficulty accessing their accounts. (cf. Olanrewaju et al., 2021, p. 5)

6.6.3.3 Examples using Two-Faktor Authentication

We would have to program a User Login and connect it with a phone number.

7 Solution variants and evaluation of the personal topic – Njeri Everlyn

By trying different kinds of databases and authentication methods, Njeri Everlyn was able to decide on which one fit the requirements the best.

7.1 SOLUTION FOR THE DATABASE TYPES

In this chapter the database types will be tried and using a utility analysis a decision will be made, as to which database is best to use.

7.1.1 How to connect Redis

First Redis has to be installed in Visual Code.

```
npm install ioredis
```

Here using Nodejs an application was created to interact with Redis. This is an example of how to set and retrieve data from Redis.

```
const Redis = require('ioredis');
const redis = new Redis();

redis.set('myKey', 'Hello, Redis!');

redis.get('myKey', (err, result) => {
  if (err) {
    console.error('Fehler:', err);
  } else {
    console.log('Wert:', result); // Ausgabe: Wert: Hello, Redis!
  }
});

async function exampleAsync() {
  try {
    await redis.set('myAsyncKey', 'Hello, Async Redis!');
    const value = await redis.get('myAsyncKey');
    console.log('Async Wert:', value); // Ausgabe: Async Wert: Hello, Async Redis!
  } catch (error) {
    console.error('Fehler:', error);
  }
}

exampleAsync();

redis.quit();
```

fig. 4 (cf. Node.Js Guide, n.d.)

7.1.2 How to connect MongoDB

To connect Node.js with MongoDB a driver needs to be downloaded. For this example, Mongoose was used.

The first step is to download install MongoDB.

```
npm install mongodb
```

Then create a connection to MongoDB.

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

const { MongoClient } = require('mongodb');

const uri = 'mongodb+srv://admin:evergreen@cluster0.9sytcpe.mongodb.net/?retryWrites=true&w=majority';

const client = new MongoClient(uri, { useNewUrlParser: true });

async function connectToMongoDB() {
  try {
    await client.connect();
    console.log('Connected to MongoDB');
  } catch (err) {
    console.error('Error connecting to MongoDB:', err);
  }
}

connectToMongoDB();
```

fig. 5 (cf. Node.Js Guide, n.d.)

```
npm install mongoose
```

The next step is to download Mongoose.

After that with the help of Mongoose a schema product can be created.

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


const uri = 'mongodb+srv://admin:evergreen@cluster0.9sytcpe.mongodb.net/?retryWrites=true&w=majority';

mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true });

const connection = mongoose.connection;

connection.on('error', (err) => {
  console.error('Error connecting to MongoDB:', err);
});

connection.once('open', () => {
  console.log('Connected to MongoDB');
});


const produkt = new mongoose.Schema({
  title: String,
  name: String,
  image: {}
});

const User = mongoose.model('User', produkt);
```

fig. 6 (cf. Connect to a MongoDB Database Using Node.Js | MongoDB Blog, n.d.)

7.1.3 How to connect MySQL

As in the other databases MySQL has to be installed. This time the packed is called mysql2.

```
npm install mysql2
```

The second step is to create a connection to MySQL.

```
const mysql = require('mysql2');

// Create a MySQL connection
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'root',
});
```

fig. 7 (cf. 'Connecting to the MySQL Database Server from Node.Js', n.d.)

After connecting to MySQL a database can be created.

```
// Connect to MySQL
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL:', err);
    return;
  }

  console.log('Connected to MySQL');

  // Step 3: Create the database
  createDatabase();
});
```

fig. 8 (cf. 'Connecting to the MySQL Database Server from Node.Js', n.d.)

A database has to be created. In this example the database eroots was created.

```
// Step 3: Create the database
function createDatabase() {
  connection.query('CREATE DATABASE IF NOT EXISTS eroots', (err) => {
    if (err) {
      console.error('Error creating the database:', err);
    } else {
      console.log('Database created (or already exists)');

      // Step 4: Switch to the newly created database
      connection.query('USE eroots', (err) => {
        if (err) {
          console.error('Error selecting the database:', err);
        } else {
          console.log('Using database: eroots');

          // Step 5: Create the "produkt" table
          createProduktTable();
        }
      });
    }
  });
}
```

fig. 9 (cf. 'Connecting to the MySQL Database Server from Node.Js', n.d.)

In the database a table hast to be created here a table product. With this table the products id, title, name and images can be saved.

```
// Step 5: Create the "produkt" table
function createProduktTable() {
  const createTableSQL = `
    CREATE TABLE IF NOT EXISTS product (
      id INT AUTO_INCREMENT PRIMARY KEY,
      titel VARCHAR(255) NOT NULL,
      name VARCHAR(255) NOT NULL,
      images JSON
    )
  `;

  connection.query(createTableSQL, (err) => {
    if (err) {
      console.error('Error creating the table:', err);
    } else {
      console.log('Table "product" created (or already exists)');
    }

    // Step 6: Close the connection
    connection.end();
  });
}
```

fig. 10 (cf. 'Connecting to the MySQL Database Server from Node.Js', n.d.)

7.1.4 Utility analysis

After considering all aspects of that databases types a utility analysis to show how the decision was made to choose MongoDB.

Criteria	Weight	Redis		MongoDB		MySQL	
		Rating	Total	Rating	Total	Rating	Total
Scaling	25%	3	0,75	2	0,5	4	1
Compatibility	35%	4	1,4	2	0,7	3	1,05
Learnig curve	10%	4	0,4	3	0,3	2	0,2
Cost	15%	3	0,45	1	0,15	1	0,15
Security	15%	5	0,75	4	0,6	4	0,6
Total	100%		3,75		**2,25**		3

Rating	1	5
Lerning Curve	Low effort	High effort
Skalierung	Easily scalable	Hard to scale
Kosten	Low cost	High cost
Compatibility	Good compatibility	schwer kompatibel
Security	Better Security	Poorer security

7.1.5 Conclusion

To sum it up, as discussed in section 7.4.2.2, using MongoDB as the database for this web application is a great choice. It offers scalability, flexibility, and strong performance. MongoDB's flexible structure, powerful search features, and the support from a large community make it a perfect fit for this project.

7.2 SOLUTION FOR AUTHENTICATION METHODS

Before the Conclusion all stated authentication methods

7.2.1 How to connect Amazon OAuth/ Amazon Login

Creating an Amazon Developer account and in the Amazon Developer Console create a new application.

The next step is to install the package for Amazon OAuth.

```
npm install express passport passport-amazon
```

The last step would be to implement OAuth 2.0 in NodeJS

(cf. *Authorization Code Grant | Login with Amazon*, n.d.)

7.2.2 How to connect Google OAuth/ Google Login

The first step in setting up a connection to Google OAuth is to install the required dependencies. This ensures that the Node.js application has the essential tools for a secure and smooth integration.

```
mkdir my-google-login-app
cd my-google-login-app
npm init -y
```

```
npm install express passport passport-google-oauth20 express-session
```

Setting up Google Auth with the help the libraries “passport” and “passport-google-oauth20”.

```
const express = require('express');
const passport = require('passport');
const GoogleStrategy = require('passport-google-oauth20').Strategy;
const session = require('express-session');

const app = express();

// Set up Express session middleware
app.use(
  session({
    secret: 'private',
    resave: false,
    saveUninitialized: true,
  })
);
```

fig. 11 (cf. Node.Js Client Library | Google Cloud, n.d.)

```
// Initialize Passport.js
app.use(passport.initialize());
app.use(passport.session());

// Configure the Google OAuth 2.0 Strategy
passport.use(
  new GoogleStrategy(
    {
      clientID: 'eroots',
      clientSecret: 'eroots-keys',
      callbackURL: 'http://localhost:3000/auth/google/callback',
    },
    (accessToken, refreshToken, profile, done) => {
      // Handle user data from Google OAuth
      // You can save the user's data to your database here
      return done(null, profile);
    }
  )
);
```

fig. 12 (cf. Node.Js Client Library | Google Cloud, n.d.)

```
// Serialize and deserialize user data
passport.serializeUser((user, done) => {
  done(null, user);
});

passport.deserializeUser((user, done) => {
  done(null, user);
});

// Implement routes
app.get('/', (req, res) => {
  res.send('Welcome to the Google OAuth App');
});
```

fig. 13 (cf. Node.Js Client Library | Google Cloud, n.d.)

```
// Redirect to Google for authentication
app.get('/auth/google', passport.authenticate('google', { scope: ['profile', 'email'] }));

// Google OAuth callback
app.get(
  '/auth/google/callback',
  passport.authenticate('google', {
    successRedirect: '/profile', // Redirect on successful authentication
    failureRedirect: '/', // Redirect on failure
  })
);
```

fig. 14 (cf. Node.Js Client Library | Google Cloud, n.d.)

```
// Profile route
app.get('/profile', (req, res) => {
  res.send('Welcome to your profile page');
});

// Start the server
const port = 3000;
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

fig. 15 (cf. Node.Js Client Library | Google Cloud, n.d.)

7.2.3 Utility analysis

After trying and considering all kinds of criteria the decision was made to use Google OAuth as the technology use for a User Login.

Criteria	Weight	Amazon Oauth		User Login		Google OAuth	
		Rating	Total	Rating	Total	Rating	Total
User-Friendly	25%	3	0,75	3	0,75	4	1
Security	35%	4	1,4	2	0,7	4	1,4
Scalability	10%	3	0,3	2	0,2	5	0,5
User Management	15%	3	0,45	2	0,3	3	0,45
Integration	15%	1	0,15	5	0,75	5	0,75
Total	100%		3,05		2,7		**4,1**

Rating	1	5
User-Friendly	High Effort	Low Effort
Scalability	Hard to Scale	Easy to Scale
User Management	High Maintenance	Low Maintenance
Integration	Hard to Integrate	Easy to Integrate
Security	Poorly Secure	Well Secure

7.2.4 Conclusion

In conclusion, after careful consideration of Google OAuth, Amazon OAuth, and User Login options, the decision was made that Google OAuth is the most suitable choice for the authentication that is needed. Google OAuth offers robust security, seamless integration, and a user-friendly experience, making it an optimal solution for this web application.

8 Basics of the personal topic – Paukner Sarah Jessica

This document area gives an overview of the individual topic from Paukner Sarah Jessica.

8.1 General

The subject area for this part of the document deals with providing and implementing security and the planning and development of the project's user experience for the web application.

8.2 Frameworks and Libraries

Frontend frameworks are software libraries or collections of tools and pre-written code that help web developers build the user interface (UI) of a website or web application. These frameworks provide a structured and efficient way to create interactive and responsive web pages. They typically include features and components for handling common tasks, such as rendering content, managing user input, and handling data from the backend.

8.2.1 React

Developed and maintained by Facebook, React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components and efficiently update the DOM (Document Object Model) for better performance. (cf. *React*, n.d.; cf. *React – Eine JavaScript Bibliothek Zum Erstellen von Benutzeroberflächen*, n.d.)

8.2.1.1 Pros

- **Easy to learn and easy use** – the simple syntax and a clear documentation help get started quickly.
- **Virtual DOM (document object model)** – document object model improves the performance and efficiency. It only updates parts of the User Interface that get changed, instead of re-rendering the full page.
- **Reusable components** – react allows to create reusable components that can be used in different pages. This helps save time, reduce code duplication, and maintains consistency.
- **One-way dataflow** – it supports one-way dataflow, which makes the code easier to understand and debug.
- **SEO-friendly** – ReactJS can render the application on the server-side, which makes it faster and more accessible as it ranks higher on search engines.

8.2.1.2 Cons

- **Fast development pace** – due to React's fast development progress, it's hard to keep up with the latest updates and features.
- **Lack of comprehensive documentation** – ReactJS is more of a library than a framework, which means it does not provide complete solutions and might need the help of other tools and libraries to complement.
- **JSX** – it requires the use of JSX, which is a syntax extension that combines HTML and JavaScript. Although JSX can make the code more readable and expressive, it can also be a barrier to developers that are not familiar with it or prefer using plain JavaScript.
- **Incomplete tooling support** – because ReactJS does not have a built-in tooling system it may be required to use additional configuration and setup.

8.2.2 Angular

Developed and maintained by Google, Angular is a comprehensive framework for building dynamic web applications. It provides features like two-way data binding, dependency injection, and a robust ecosystem for building large-scale applications. (cf. *Angular*, n.d.)

8.2.2.1 Pros

- **Supported by Google** – Angular is supported by Google, which means it has a large and active community that updates regularly and has high-quality documentation.
- **Component-based architecture** – this architecture boosts developer productivity and provides high quality of code. These components can be reused, customized, and combined.
- **Platform-agnostic** – it is platform-agnostic, which means it can run on any browser, device, or operating system.
- **Angular material** – Angular material is a set of User Interface components that follow the Material Design guidelines.
- **Seamless updates** – the Angular CLI tool can automate the process of upgrading the application to the latest version.

8.2.2.2 Cons

- **Difficult learning process** – Angular has a steep learning curve that can make it difficult for beginners to understand its many concepts, features, and syntaxes.
- **Complex dependency injection system** – due to this technique for managing dependencies between components, it can make the code more prone to errors and difficult to debug.
- **Verbose and complex** – Angular requires more code and configuration than other frameworks, which does not make it very suitable for simple or small applications.
- **Low ranking** – Angular has a low ranking on some developer surveys.

8.2.3 Vue.js

Vue.js is a progressive JavaScript framework for building user interfaces. It is known for its simplicity and flexibility, making it easy to integrate into existing projects and start small and gradually scale up. (cf. *Vue.Js - The Progressive JavaScript Framework* | *Vue.Js*, n.d.)

8.2.3.1 Pros

- **Easy to learn** – Vue.js has a simple syntax and a clear documentation that can help get started quickly.
- **One-way dataflow** – it supports one-way dataflow, which makes the code easier to understand and debug.
- **High performance** – due to built-in tools such as change detection, lazy loading, code splitting and ahead-of-time compilation, it has a high performance.
- **Angular material** – Angular material is a set of User Interface components that follow the Material Design guidelines.
- **Seamless updates** – the Angular CLI tool can automate the process of upgrading the application to the latest version.

8.2.3.2 Cons

- **Difficult learning process** – Vue.js has a steep learning curve that can make it difficult for beginners to understand its many concepts, features, and syntaxes.
- **Complex dependency injection system** – due to this technique for managing dependencies between components, it can make the code more prone to errors and difficult to debug.
- **Verbose and complex** – Vue.js requires more code and configuration than other frameworks, which does not make it very suitable for simple or small applications.
- **Low ranking** – Vue.js has a low ranking on some developer surveys.

8.2.4 Ember.js

Ember.js is a full-featured JavaScript framework that follows the convention over configuration (CoC) and don't repeat yourself (DRY) principles. It provides a structured way to build ambitious web applications. (cf. *Ember.Js - A Framework for Ambitious Web Developers*, n.d.)

8.2.4.1 Pros

- **Convention-over-configuration** – Ember.js is a comprehensive framework that provides a convention-over-configuration approach, which makes operating with the boilerplate code or the architecture of the application easier.
- **Strong router** – it has a strong router, which allows to define the URL structure and the transitions between different states of the application.
- **Built-in testing framework** – the built-in testing framework makes it easier to write and run tests for the application.
- **Forward-thinking** – Ember.js adopts the latest standards and features of JavaScript.
- **Server-side rendering** – with FastBoot, Ember.js supports server-side rendering, which improves the performance of the application.

8.2.4.2 Cons

- **Opinionated and inflexible** – developers must follow the Ember way for coding, which is very restricting.
- **Difficult learning process** – Ember.js has a steep learning curve and requires familiarity with many concepts and tools.
- **Popularity and usage** – Ember.js is not as popular or widely used as some other frameworks, so it may have less community support, resources, or job opportunities.

8.2.5 Svelte

Svelte is a relatively new framework that compiles components into highly efficient vanilla JavaScript during build time. It's known for its small bundle sizes and high performance. (cf. *Svelte • Cybernetically Enhanced Web Apps*, n.d.)

8.2.5.1 Pros

- **Focus on performance** – the process of transforming components into plain JavaScript at the build phase can result in a faster runtime execution and smaller bundle sizes.
- **Simplicity and readability** – the syntax is easy to learn and understand. Svelte avoids using boilerplate code and complex abstractions and allows the usage of different code languages in the same file.
- **Reactivity and state management** – Svelte's reactivity system automatically updates the User Interface when the state changes, without using extra libraries or hooks.

8.2.5.2 Cons

- **Lack of IDE support and dev tools** – its tooling is not as mature and robust as other frameworks, so it may not have the same level of integration and features in the editor or browser.
- **Small community and ecosystem** – due to it being a newer and growing framework it does not have many resources, tutorials, libraries, or plugins.
- **No cross-platform support** – Svelte does not have a solution for building cross-platform applications and may require other frameworks or tools, which may introduce complexity or compatibility issues.

8.2.6 Backbone.js

Backbone.js is a lightweight JavaScript framework that provides the basic structure for developing web applications. It offers models, views, collections, and routers to help organize code. (cf. *Backbone.Js*, n.d.)

8.2.6.1 Pros

- **JavaScript functions** – It allows to create an application by using JavaScript functions without any special syntax or extension.
- **Automated updates** – the html code for the Backbone.js developed application gets updated automatically whenever the data changes.
- **Open source** – Backbone.js is free to use and has a large community of developers that contribute to its development and documentation.
- **Data Model** – it provides a simple way to manage the data using models that can be validated, saved, and synced with the server.
- **RESTful** – it supports the RESTful API design which makes it easier to communicate with the server using HTTP methods.
- **Backbone collections** – It allows the grouping of models into collections that can be sorted, filtered, and manipulated.
- **Backbone router** – It enables the creation of dynamic URLs for the application that can be bookmarked and navigated.
- **Backbone sync** – Backbone.js allows customization of how the models and collections interact with the server using different options and events.

8.2.6.2 Cons

- **Lack of structure** – It does not enforce any specific structure or convention for the application, which can lead to messy and inconsistent code.
- **Boilerplate code** – Backbone.js requires the writing of a lot of boilerplate code for common tasks such as rendering views, binding events, and fetching data.
- **Dependency on other libraries** – It depends on other libraries such as Underscore.js and jQuery for its core functionality, which can increase the size and complexity of the application.
- **Limited support for complex UIs** – It does not have built-in features for creating complex user interfaces.

8.2.7 Aurelia

Aurelia is a modern, open-source JavaScript framework that focuses on clean code and extensibility. It's designed to be easy to learn and integrate with other libraries. (cf. *Home | Aurelia*, n.d.)

8.2.7.1 Pros

- **Easy to Learn** – It uses standard JavaScript syntax, simple conventions, and minimal configuration.
- **Great Performance** – Aurelia supports two-way data binding, routing, User Interface composition, dependency injection, and modular design.
- **Extensibility** – It integrates well with other components, such as Web Components, TypeScript, RxJS, and Ionic.
- **Testable** – Aurelia supports various testing tools and frameworks.

8.2.7.2 Cons

- **Smaller Community** – Aurelia has a smaller community and less documentation compared to some competitors like Angular or React.
- **Dependency Overhead** – It relies on many dependencies and is maintained by a small team, potentially leading to less stability.
- **Limited UI Libraries** – It lacks a variety of free and rich UI frameworks or libraries, limiting UI design options.

8.2.8 Meteor

Meteor is a full-stack JavaScript framework that allows developers to build real-time web applications with ease. It includes both frontend and backend components for a seamless development experience. (cf. *Stop Fighting with Frameworks and Start Shipping Real Apps - Meteor.Js*, n.d.)

8.2.8.1 Pros

- **Developer-Friendly** – Meteor offers a fast and enjoyable developer experience.
- **Scalable and Secure** – It supports scalability, security, and offline mode.
- **Strong Community** – It has a strong community and a wealth of resources.

8.2.8.2 Cons

- **Limited Database Options** – Tied to MongoDB as the default database Meteor may not suit all projects.
- **Testing Challenges** – Meteor lacks an official testing framework.
- **Performance Issues** – It can be heavy and face performance issues on mobile devices.

8.2.9 Mithril

Mithril is a minimal JavaScript framework for building web applications. It's known for its small size and simplicity, making it a good choice for small to medium-sized projects. (cf. *Mithril.Js*, n.d.)

8.2.9.1 Pros

- **Lightweight** – Small (12kb zipped), no dependencies, and a compact API.
- **Simple Syntax** – Mithril's features are simple and have an expressive syntax that resembles HTML.
- **Supports Features** – It includes features like transitions, animations, slots, and custom elements.

8.2.9.2 Cons

- **Compatibility Issues** – Mithril is not compatible with some existing libraries and tools relying on a virtual DOM or a framework runtime.
- **New and** Experimental – Being relatively new, it may have some bugs or limitations.
- **Smaller Community** – Mithril has a smaller community and ecosystem compared to some other frameworks.

These frameworks provide tools and patterns to help developers organize code, manage state, handle routing, and build interactive user interfaces efficiently. The choice of which frontend framework to use often depends on project requirements, team expertise, and personal preferences. Since the web development landscape evolves rapidly, it's essential to keep up with the latest trends and updates in frontend development.

8.3 User Experience

Moser says that: In order for a product to be successful, it must evoke positive emotions in the customer - because they create the necessary motivation to use the product with joy and actively recommend it. (cf Moser, 2012, p. 3)

8.3.1 Expectations vs. Product Qualities

User experience is the result of the interaction between a user's expectations and the qualities of the product. When the product exceeds these expectations, the experience is positive, and when it falls short, the experience is negative. (cf. Moser, 2012, pp. 20–22)

8.3.2 Emotions Drive Experience

User experience is heavily influenced by the emotions it evokes. Emotions are an unconscious evaluation of the situation by the limbic system. Positive experiences trigger happiness, leading users to seek them out again, while negative experiences are avoided. This emotional aspect is at the core of User Experience Design. (cf. Moser, 2012, pp. 20–22)

8.3.3 Expectations Are Shaped by Experience

Expectations are shaped by a user's prior experiences and allow them to estimate the value of something or what is possible. With technological advancements, expectations evolve over time. What is considered exciting today becomes normal in the future and is taken for granted the day after. (cf. Moser, 2012, pp. 20–22)

8.3.4 Context Matters

User expectations vary based on the context in which a product is used. For example, a user has different expectations for their mobile phone when using it for work compared to leisure. (cf. Moser, 2012, pp. 20–22)

8.3.5 Expectations Comprise Various Aspects

User expectations encompass a range of factors, including utility, usability, aesthetics, and accessibility. During the analysis phase, it's crucial to gather these expectations along with all relevant contextual information and find innovative design solutions to meet them. (cf. Moser, 2012, pp. 20–22)

8.4 Security

Security refers to the proactive measures taken to minimize risks and safeguard both data and systems. It involves a combination of technology, training, awareness, and continuous updates to stay ahead of constantly evolving threats.

8.4.1 Up to Date

In order to keep the web application safe it is vital to keep the software and systems consistently updated to patch vulnerabilities. (cf. *Hacking & Security*, 2018, p. 54)

8.4.2 Encryption

It is vital to ensure that all devices have encrypted file systems to protect the data. (cf. *Hacking & Security*, 2018, p. 54)

8.4.3 VPN Usage

The access to mission-critical data should only be possible through a Virtual Private Network (VPN). (cf. *Hacking & Security*, 2018, p. 55)

8.4.4 Security Testing

It is of utmost importance to conduct regular security tests on the IT infrastructure to identify weaknesses. (cf. *Hacking & Security*, 2018, p. 55)

8.4.5 Penetration Tests

External penetration tests should be conducted as a precaution. (cf. *Hacking & Security*, 2018, p. 55)

8.4.6 Comprehensive Security

It should be considered to use external resources, services, physical security, and legal aspects of IT security. (cf. *Hacking & Security*, 2018, p. 55)

8.5 Web Application Security

In order to ensure the security of a web application, the following points must be considered: (cf. Hoffman, 2020, pp. 193–203)

8.5.1 Involving Security Early

Security should be integrated into the application's architecture and design from the beginning of the development process. The application should be analysed for potential flaws and vulnerabilities. (cf. Hoffman, 2020, p. 193)

8.5.2 Addressing Architecture-Level Flaws

Identifying and rectifying deep security flaws in the architecture, such as improper authentication schemes or flawed integration with external components like search engines is important. Resolving security issues at this stage is more cost-effective than addressing them in production. (cf. Hoffman, 2020, pp. 193–195)

8.5.3 Providing Secure Equivalent Functionality

It is important to ensure that any insecure functionality used by customers is replaced with secure alternatives. Offer customers a migration plan to minimize downtime during the transition. (cf. Hoffman, 2020, p. 203)

8.5.4 Consider the Impact of Security Flaws

It is vital to understand that deep architecture-level security issues may necessitate the rewriting of multiple modules within the application. For example, fixing a flaw in a multiplayer module may require changes in networking and game modules built on top of it. (cf. Hoffman, 2020, p. 203)

8.5.5 Anticipate Exploitation

It’s important to be aware that if a security flaw is exploited, it can result in financial losses and consume engineering resources for remediation. (cf. Hoffman, 2020, p. 203)

8.5.6 Mitigate Reputation Damage

The publication of security flaws can lead to bad publicity and loss of customers, affecting the business's reputation and engagement. (cf. Hoffman, 2020, p. 203)

8.5.7 Prioritize Security in Architecture

Emphasizing the importance of addressing security concerns during the architecture phase of application development is vital because early identification and resolution of security issues save time and money in the long run. (cf. Hoffman, 2020, p. 203)

9 Solution variants and evaluation of the personal topic – Paukner Sarah Jessica

This document area gives an overview of the solution variants and evaluation of the individual topic from Paukner Sarah Jessica.

9.1 Frontend Frameworks

This document area gives an overview of the three solution variants with exemplary tic tac toe code snippets.

9.1.1 React

Tic tac toe code snippets using React.

9.1.1.1 JSX

```
import React, { useState } from 'react';

function TicTacToe() {
  const [squares, setSquares] = useState(Array(9).fill(null));
  const [xIsNext, setXIsNext] = useState(true);

  const handleClick = (i) => {
    const newSquares = squares.slice();
    if (calculateWinner(newSquares) || newSquares[i]) {
      return;
    }
    newSquares[i] = xIsNext ? 'X' : 'O';
    setSquares(newSquares);
    setXIsNext(!xIsNext);
  };

  const renderSquare = (i) => {
    return (
      <button className="square" onClick={() => handleClick(i)}>
        {squares[i]}
      </button>
    );
  };
```

```
  const winner = calculateWinner(squares);
  const status = winner ? `Winner: ${winner}` : `Next player: ${xIsNext ? 'X' : 'O'}`;

  return (
    <div>
      <div className="status">{status}</div>
      <div className="board-row">
        {renderSquare(0)}
        {renderSquare(1)}
        {renderSquare(2)}
      </div>
      <div className="board-row">
        {renderSquare(3)}
        {renderSquare(4)}
        {renderSquare(5)}
      </div>
      <div className="board-row">
        {renderSquare(6)}
        {renderSquare(7)}
        {renderSquare(8)}
      </div>
    </div>
  );
}
```

```
function calculateWinner(squares) {
  // Winning combinations logic
}

export default TicTacToe;
```

fig. 16 React.js tic tac toe code snippet

9.1.2 Angular

Tic tac toe code snippets using Angular.

9.1.2.1 HTML

```html
// TicTacToe.component.html
<div>
  <div class="status">{{ status }}</div>
  <div class="board-row">
    <button class="square" *ngFor="let square of squares; let i = index" (click)="handleClick(i)">{{ square }}</button>
  </div>
</div>

```

fig. 17 Angular tic tac toe snippet for the html

9.1.2.2 TypeScript

```typescript
// TicTacToe.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-tic-tac-toe',
  templateUrl: './tic-tac-toe.component.html',
  styleUrls: ['./tic-tac-toe.component.css']
})
export class TicTacToeComponent {
  squares: string[] = Array(9).fill(null);
  xIsNext: boolean = true;

  handleClick(i: number): void {
    // Click event logic
  }

  status: string = this.calculateStatus();

  private calculateStatus(): string {
    // Status calculation logic
  }
}

```

fig. 18 Angular tic tac toe snippet for typescript

9.1.3 Vue.js

Tic tac toe code snippets using Vue.js.

9.1.3.1 HTML

```
<template>
  <div>
    <div class="status">{{ status }}</div>
    <div class="board-row">
      <button class="square" v-for="(square, i) in squares" @click="handleClick(i)">{{ square }}</button>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      squares: Array(9).fill(null),
      xIsNext: true
    };
  },
  methods: {
    handleClick(i) {
      // Click event logic
    },
    calculateStatus() {
      // Status calculation logic
    }
  },
  computed: {
    status() {
      return this.calculateStatus();
    }
  }
};
</script>
```

fig. 19 Vue.js tic tac toe code snippet

9.2 COST-BENEFIT ANALYSIS

This document area gives an overview of the three solution variants with the cost-benefit analysis.

CRITERIA	GEWICHT	FRAMEWORKS AND LIBRARIES								
		REACT	PUNKTE	ANGULAR	PUNKTE	VUE.js	PUNKTE		Ember.js	PUNKTE
Performance	5	4	20	4	20	4	20		3	15
Flexibility and adaptability	4	4	16	3	12	4	16		2	8
Community & Support	4	5	20	5	20	5	20		2	8
Documentation and training	3	5	15	5	15	4	12		2	6
Accessibility and SEO friendliness	3	3	9	4	12	3	9		2	6
Mobile Responsiveness	3	4	12	4	12	4	12		2	6
Security	4	5	20	5	20	4	16		2	8
Cost and Licensing	2	5	10	5	10	5	10		5	10
Speed of development	4	4	16	3	12	4	16		2	8
Scalability	4	5	20	5	20	4	16		2	8
Total		15,8		15,3		14,7			8,3	

fig. 20 Cost-benefit-analysis for Frameworks and Libraries of React.js, Angular and Vue.js (src.: ERoots-Nutzwertanalyse-v2)

RATING	
5	excellent
4	good
3	medium
2	bad
1	very bad

9.3 CONCLUSION

In conclusion, the decision to use **React** is based on a thorough cost-benefit analysis. **React** offers superior performance, robust community support, security prioritization, and cost-effectiveness, making it an ideal choice for this project's frontend development needs.

10 Basics on personal topics – Andrei Elias

"Planning and implementing/developing of deployment, testing and quality assurance" is the individual topic that Andrei Elias writing about.

10.1 Deployment

Deployment encompasses all the activities involved in making the application accessible to end-users. It involves setting up the necessary infrastructure, automating the process of pushing code changes to production, and ensuring the application runs smoothly in its intended environment.

This phase includes selecting an appropriate hosting platform, configuring servers, databases, and other necessary components.

Continuous Integration and Continuous Deployment (CI/CD) pipelines are established to automate the build and release process, reducing manual intervention. Different deployment strategies, like blue-green deployments or canary releases, are employed to minimize downtime and mitigate risks during updates. Additionally, robust monitoring and logging practices can be implemented to keep track of the application's performance and identify and address any issues promptly. Scaling strategies are put in place to handle increased traffic, ensuring a seamless user experience.

10.1.1 Environment Setup

The initial phase entails selecting an appropriate hosting platform in alignment with project requisites. Factors such as scalability, performance, cost, and ease of use should be carefully considered. Following this, server configuration is imperative, involving the setup of virtual machines or containerized solutions such as Docker. Additionally, the selection and configuration of a suitable database system, such as PostgreSQL or MongoDB, is essential for maintaining a robust foundation. Server Configuration:

Depending on the expected traffic and resource requirements, set up servers with appropriate CPU, RAM, and storage. Utilize cloud providers' offerings for easy scalability.

10.1.2 Continuous Integration/Continuous Deployment (CI/CD)

To facilitate streamlined development workflows, a robust CI/CD pipeline must be established. This involves the integration of version control systems, such as Git, with a CI/CD tool, enabling automated builds and deployments upon code commits.

10.1.3 Deployment Strategies

Employing effective deployment strategies minimizes downtime and guarantees a seamless release process. Among these strategies, the blue-green deployment approach is notable for maintaining two identical environments (blue and green), enabling traffic redirection during updates. Canary releases provide the opportunity for incremental updates to a limited user base, thereby enabling early issue detection. Additionally, rolling updates allow for a controlled update process by incrementally updating a subset of servers while ensuring uninterrupted service availability.

10.1.4 Monitoring and Logging

For effective deployment management, robust monitoring and logging mechanisms should be implemented. Tools like Prometheus for metric collection and Grafana for visualization prove invaluable in tracking critical performance metrics. Alerting configurations should be in place to promptly notify the team of anomalies or critical issues. Furthermore, centralized logging solutions, such as the ELK Stack or Splunk, should be integrated to facilitate efficient troubleshooting and debugging.

10.1.5 Scaling

Anticipating increased traffic loads, it is imperative to establish scalability measures. This involves the implementation of both horizontal scaling, achieved by adding more servers or instances, and vertical scaling, achieved by upgrading individual servers with additional resources. Load balancing mechanisms should be employed to evenly distribute traffic among multiple servers, thereby optimizing performance.

10.1.6 WebApp Deployment Services

10.1.6.1 AWS Amplify

AWS Amplify is a comprehensive platform provided by Amazon Web Services (AWS) that simplifies the process of building, deploying, and hosting modern web applications. It offers a seamless integration with AWS services and provides a variety of deployment options, including automatic CI/CD pipelines for React applications.

10.1.6.2 Pros

- Offers a wide range of AWS services for building and deploying applications.
- CI/CD Pipelines
- Integrates seamlessly with other AWS services for scalable applications.

10.1.6.3 Cons

- Complex deployment procedure
- Costs can escalate if not managed carefully, especially for larger applications.

10.1.6.4 Azure

Microsoft Azure is a cloud computing platform that offers a range of services for building, deploying, and managing applications. Azure provides various options for hosting React applications, including Azure App Service, which allows for easy deployment and scaling of web apps.

10.1.6.5 Pros

- Offers seamless integration with Microsoft products and services.
- Provides an easy-to-use platform for deploying web applications.
- Can easily scale applications based on demand.

10.1.6.6 Cons

- Requires familiarity with Microsoft's ecosystem.
- The Pricing model can be complex.

10.1.6.7 Firebase

Firebase, developed by Google, is a platform that provides a wide range of tools and services for developing and deploying web and mobile applications. It offers a hosting service that allows for straightforward deployment of React applications along with other features like real-time database, authentication, and more.

10.1.6.8 Pros

- Provides a real-time database that synchronizes data across devices in real-time.
- Offers easy-to-implement authentication services.
- Provides hosting and CDN (content-delivery network) capabilities for web applications.

10.1.6.9 Cons

- Tightly integrated with Google's ecosystem
- Limited Backend Options for complex Applications

10.1.6.10 GitHub Pages

GitHub Pages is a free hosting service provided by GitHub. It's designed for hosting static websites directly from a GitHub repository. While it's great for simple React applications, it may require additional steps for more complex apps.

10.1.6.11 Pros

- Provides free hosting for static websites directly from GitHub repositories.
- Version Control: Integrates seamlessly with Git for version control.

10.1.6.12 Cons

- Limited to hosting static content.
- Doesn't support server-side processing or dynamic content generation.

10.1.6.13 Heroku

Heroku is a cloud platform-as-a-service (PaaS) that simplifies application deployment. It supports various programming languages, including Node.js for React applications. Heroku offers a straightforward deployment process and provides a range of add-ons for extending functionality.

10.1.6.14 Pros

- Offers a user-friendly interface for application deployment.
- Provides a wide range of add-ons for extending functionality.
- Automatic Scaling

10.1.6.15 Cons

- There's a free tier, but it has limitations for resource-intensive applications.
- Latency: weak Performance compared to other services

10.1.6.16 Netlify

Netlify is a modern hosting and automation platform tailored for web projects. It excels in continuous deployment, allowing for automatic builds and deployments directly from a Git repository. Netlify also provides features like serverless functions and form handling.

10.1.6.17 Pros

- Offers automatic builds and deployments from Git repositories.
- Supports serverless architecture for backend logic.
- Provides a global content delivery network for optimal performance.

10.1.6.18 Cons

- Complex/High Pricing
- For more complex projects, additional build configuration may be needed.

10.1.6.19 Vercel

Vercel is a platform optimized for frontend development and deployment. It specializes in hosting React applications and offers seamless integrations with popular frameworks and build tools. Vercel provides a global content delivery network (CDN) for optimal performance.

10.1.6.20 Pros

- Tailored specifically for frontend development and hosting, making it a good option for react-based web applications.
- Provides automatic deployments from Git repositories with minimal configuration.
- Global CDN: Ensures fast and reliable content delivery worldwide.

10.1.6.21 Cons

- Limited Backend Functionality
- Limited Database Options

10.1.6.22 Render

Render is a cloud platform that focuses on simplicity and developer-friendliness. It offers a streamlined deployment process for web applications, including React projects. Render provides automatic scaling, SSL, and a straightforward pricing model.

10.1.6.23 Pros

- Designed to be straightforward and developer friendly.
- Automatic Scaling
- Transparent Pricing

10.1.6.24 Cons

- May not have as extensive a set of services or integrations as larger cloud platforms.
- Less Flexibility: it offers less customization options for complex applications.

10.1.6.25 S3/CloudFront

Amazon S3 (Simple Storage Service) combined with CloudFront provides a powerful combination for hosting and delivering web content. S3 stores the static assets while CloudFront acts as a content delivery network, ensuring low latency and high availability.

10.1.6.26 Pros

- Scalability: Provides high scalability and reliability for static content.
- Cost-Efficient

10.1.6.27 Cons

- Complex Setup
- Primarily designed for static content, requires additional services for dynamic applications

10.1.6.28 Surge

Surge is a straightforward platform designed for quickly deploying static web pages. While it's perfect for hosting single-page React applications, it's important to note that it's optimized for simpler projects and may not provide all the features of larger hosting platforms.

10.1.6.29 Pros

- Simplicity
- Free Hosting

10.1.6.30 Cons

- Limited Functionality for dynamic Applications
- No Server-Side Processing

10.2 TESTING

Testing is a critical phase in software development aimed at ensuring the application functions correctly, meets its requirements, and remains robust in various scenarios. It encompasses multiple levels of examination, starting with unit tests that validate individual components. Integration testing focuses on interactions between different parts of the application, while end-to-end (E2E) testing simulates real user interactions. Automation is key in this phase, with automated tests integrated into the Continuous Integration/Continuous Deployment (CI/CD) pipeline. Load and performance testing ensure the application can handle the expected load without compromising on performance. Comprehensive testing practices instil confidence in the application's reliability and help catch and address issues early in the development process.

10.2.1 Unit Testing

“A unit test exercises a "unit" of code in isolation and compares actual with expected results.” (Olan, 2003, p. 320)

In the context this project, unit testing is pivotal for ensuring the correctness and functionality of individual components. This involves utilizing testing libraries like Jest and React Testing Library to construct test cases that verify the rendering and behaviour of components under various scenarios. Additionally, the practice of mocking external dependencies or services plays a crucial role in isolating unit tests and maintaining test independence.

10.2.2 Integration Testing

Integration testing addresses the interaction between different components or modules within the application. By focusing on scenarios where multiple components collaborate, it ensures seamless integration and functionality. Test cases are designed to encompass scenarios such as form submissions, data fetching, and state management. Integration tests may involve the use of stubs or mocks to simulate interactions with external services or APIs.

10.2.3 End-to-End (E2E) Testing

Employing tools like Cypress or Selenium, E2E tests simulate user interactions with the application, validating critical user flows such as registration, product search, and the checkout process. These tests facilitate the identification of any discrepancies in behaviour, ensuring a consistent and satisfactory user experience.

10.2.4 Automated Testing

“Automated testing tools are a collection of software products designed specifically to assist software testers and software testing managers with different aspects of a testing project.” (Everett & McLeod Jr, 2007, p. 159)

Incorporating automated testing into the Continuous Integration/Continuous Deployment (CI/CD) pipeline is instrumental in maintaining code quality and integrity. This involves configuring the CI/CD tool to automatically execute unit, integration, and E2E tests upon each code commit. By automating testing procedures, potential issues can be found quickly and prevented.

10.2.5 Testing Software

10.2.5.1 Protractor

Protractor is an end-to-end testing framework designed specifically for testing Angular applications. It is an open-source automation testing framework built on top of WebDriverJS, which allows it to interact with web elements just like a user would.

10.2.5.1.1 Pros:

- Asynchronous Execution
- Built-in Waits
- Good available Documentation

10.2.5.1.2 Cons:

- Angular-Centric
- Limited Browser Support

10.2.5.2 Jest

Jest is a widely used testing framework primarily built for JavaScript and especially popular in the React ecosystem. It is developed by Facebook and provides a robust testing environment.

10.2.5.2.1 Pros:

- Performance
- Snapshot Testing helps in detecting unexpected changes in UI components.
- Good Documentation
- Integration with React

10.2.5.2.2 Cons:

- Focused on JavaScript
- Less Focus on Browser Interaction, more coding

10.2.5.3 Cypress

Cypress is an end-to-end and component/Integration testing framework that provides fast, reliable, and interactive testing for web applications. It runs directly in the browser and is known for its clear and intuitive API.

10.2.5.3.1 Pros:

- Performance
- Real-Time Browser Interaction
- Excellent Documentation

10.2.5.3.2 Cons:

- Limited Browser Support

10.3 Quality Assurance

Quality Assurance (QA) is an approach to ensure that the application not only functions correctly but also meet high standards of quality in terms of code, security, accessibility, and usability. It involves thorough code reviews to maintain code quality and adherence to coding standards. Static code analysis tools are used to catch potential issues and enforce best practices.

10.3.1 Code Reviews

Code reviews serve as a fundamental practice in maintaining code quality and adherence to established standards. They involve a systematic examination of code by peers to identify and rectify potential issues, promote best practices, and facilitate knowledge sharing. Establishing clear code review guidelines is essential for conducting constructive and productive reviews.

10.3.2 Static Code Analysis

Leveraging tools like ESLint or TSLint aids in enforcing coding standards and identifying common errors. By configuring linters to analyse the codebase, potential issues can be addressed, ensuring consistency and adherence to coding conventions.

11 Solution variants and evaluation of the personal topic – Andrei Elias

This document area gives an overview of the solution variants and evaluation of the individual topic from Andrei Elias.

11.1 Deployment Cost-Benefit analysis

The cost-benefit analysis assessed various cloud hosting services against defined criteria, with the aim of identifying the most suitable platform for deploying a React web application. The criteria included performance, user-friendliness, pricing, flexibility/compliance, availability, and Content Delivery Network (CDN) capabilities. Each criterion was weighted according to its perceived importance, allowing for a comprehensive evaluation.

Criteria	Weight	AWS Amplify	Points	Azure	Points	Firebase	Points	GitHub Pages	Points	Heroku	Points
Performance/Scalability	5	4	20	4	20	2	10	4	20	3	15
User-friendliness	3	2	6	3	9	5	15	5	15	4	12
Cost	4	2	8	2	8	4	16	5	20	2	8
Flexibility/Compliance	5	3	15	4	20	2	10	2	10	4	20
Availability	4	4	16	4	16	4	16	4	16	4	16
CDN	3	3	9	3	9	3	9	3	9	2	6
Total		**61,67%**		**68,33%**		**63,33%**		**75,00%**		**64,17%**	

Netlify	Points	Vercel	Points	Render	Points	S3/CloudFront	Points	Surge	Points
4	20	5	25	3	15	4	20	2	10
4	12	5	15	4	12	2	6	5	15
2	8	4	16	4	16	5	20	5	20
4	20	4	20	2	10	4	20	2	10
5	20	5	20	5	20	5	20	4	16
4	12	5	15	4	12	5	15	3	9
76,67%		**93%**		**70,83%**		**84,17%**		**66,67%**	

The criteria were weighted from 1 to 5, 5 being the most important. The different Services were graded also from 1 to 5, 5 being the best in the specific criteria. After that the weightings were multiplied with the given Points and the percentage to the maximal possible points was calculated.

Rating	
Weight	1 unimportant ↔ important 5
Points	1 bad ↔ excellent 5

Compared were 10 Web Application Deployment Services: AWS Amplify (Amazon), Azure (Microsoft), Firebase (Google), GitHub Pages, Heroku, Netlify, Vercel, Render, S3/Cloudfront and Surge. The Service with the most points is Vercel (92.5%).

11.2 Testing Cost-Benefit analysis

This analysis assessed 3 Test Software (Protractor, Jest and Cypress) using 6 criteria: Performance/Speed, Available Documentation, Flexibility/Compliance, Tool-Integration, Pricing/Licensing, Maintainability/Stability.

Criteria	Weight	Protractor	Points	Jest	Points	Cypress	Points
Performance/Speed	5	4	20	4	20	5	25
Documentation	5	4	20	4	20	5	25
Flexibility/Compliance	3	1	3	4	12	4	12
Tool Integration (CI, IDEs)	4	4	16	4	16	4	16
Cost/Licensin	4	5	20	5	20	5	20
Maintainability/Stability	3	4	12	4	12	4	12
Total		**75,83%**		**83,33%**		**92%**	

Rating	
Weight	1 unimportant ↔ important 5
Points	1 bad ↔ excellent 5

Again, the criteria were weighted from 1 to 5, 5 being the most important. The different Services were graded also from 1 to 5, 5 being the best in the specific criteria. After that the weightings were multiplied with the given Points and the percentage to the maximal possible points was calculated.

According to this analysis, the best Test Software being Cypress, providing the User with easy and fast tests, CI, and a lot of Documentation.

11.2.1 Cypress

Cypress is an open-source test automation software primarily used for testing web applications. It provides a user-friendly and powerful platform for writing, executing, and debugging tests for web applications. It is also compatible with CI.

11.2.1.1 E2E Testing

Cypress provides the user with an easy platform to execute E2E-testing, as shown in the simple example below:(cf. *Writing Your First E2E Test | Cypress Documentation*, 2023)

```
JS visit.cy.js ×
eroots-front > cypress > e2e > JS visit.cy.js > describe('templat
1  describe('template spec', () => {
2    it('passes', () => {
3      cy.visit('http://localhost:3000')
4    })
5  })
```

fig. 21 (cf. Writing Your First E2E Test | Cypress Documentation, 2023)

This screenshot shows a sample test script. The script's purpose is to visit the local web application and verify if the HTTP status code returned falls within the range of 2xx or 3xx.

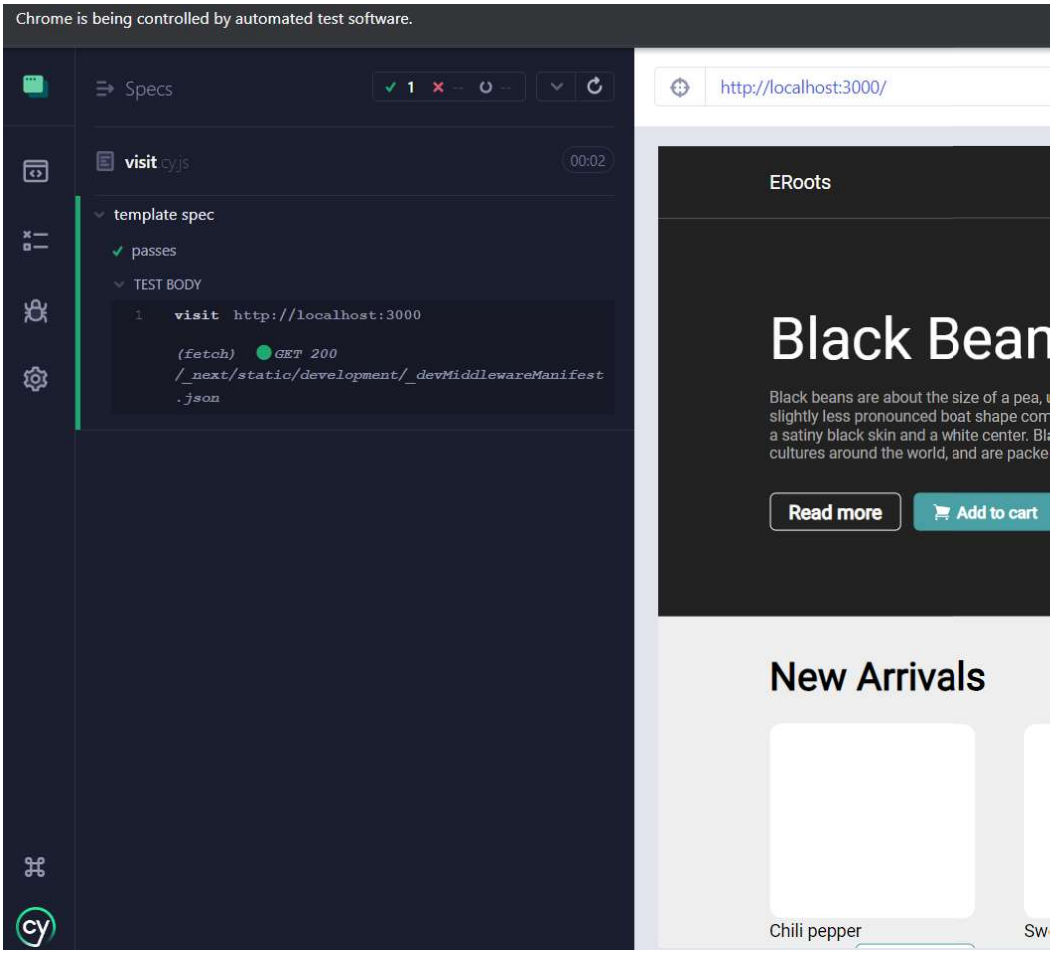


fig. 22 (cf. Writing Your First E2E Test | Cypress Documentation, 2023)

This screenshot displays the Chrome automated test browser window after the Cypress test has been executed. On the right it loads the page, and, on the left, it executes the written tests shown in the previous screenshot.

11.2.1.2 Component Testing

Below is the sample script and result of a simple component test made with cypress.

```
JS Input.cy.js U ×
eroots-front > cypress > component > JS Input.cy.js > ...
1  import React from 'react'
2  import Input from '../../components/Input'
3
4  describe('<Input />', () => {
5    it('renders', () => {
6      cy.mount(<Input />)
7      cy.get('.Input__StyledInput-sc-3f4a8800-0').should('exist');
8    });
9  })
```

fig. 23 (cf. Writing Your First E2E Test | Cypress Documentation, 2023)

This script mounts the Input component from the webapp. This means that it simulates rendering the component as it would happen in an actual React application. The component is rendered and attached to the DOM. After that it checks if the Input (generated by Styled) got rendered and therefore exists. (cf. *Writing Your First E2E Test | Cypress Documentation*, 2023)

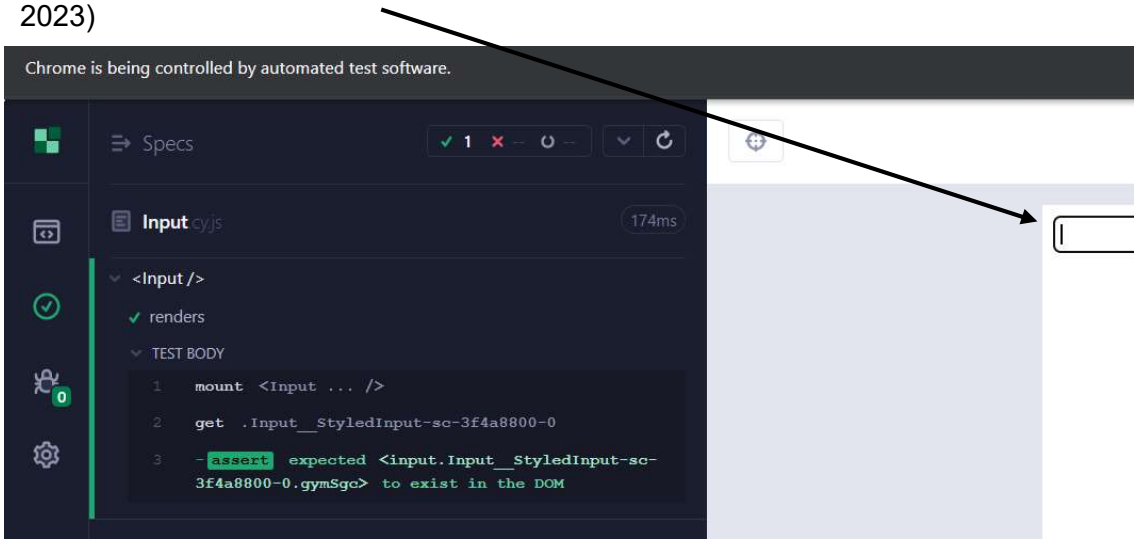


fig. 24 (cf. Writing Your First E2E Test | Cypress Documentation, 2023)

The Test runs automatically in the test Browser (Chrome) once the file is saved. Test Results are positive.

12 Basics on personal topics – Braun Valerian

"How can a web application be implemented that provides HTTP endpoints, receives events, and the development of this system?" is the topic that Braun Valerian writes about it this preliminary.

12.1 Introduction

In this part will be a detailed description of the Question to: “How to implement an application with a responsive HTTP-Endpoint”. What kind of languages/frameworks you can use and what tools are necessary to accomplish this.

Steps that must be done to create a web-application:

- The Selection of a programming language
  - It is important to choose a language that has all the necessary tools to create a web application.
- What Framework to use
  - After choosing a language the next task is to evaluate what framework to use like Spring for Java, Flask for Python, and others.
- Install all the necessary libraries.
- Create the application.
- Create the HTTP Endpoint
- Implement a Request Handling
- Send a Response

12.2 Suggested solution

In this section will be looked at the different option that offer a solution.

12.2.1 JavaScript/Node.js

Node.js is a cross-platform JavaScript runtime environment known for its low latency and high throughput in serving web requests. It achieves this by taking a non-blocking approach to handle requests, where it doesn't waste time or resources waiting for I/O operations to complete. (*About*, n.d.)

12.2.1.1 Advantages of Node.js

- Single Language (JavaScript)
- Real-time Applications
- Cross-platform

12.2.1.2 Disadvantages of Node.js

- Single-threaded

12.2.2 Python

Python is a modern programming language that supports object-oriented, functional, and imperative programming styles. It is ideal for the beginner because of its readability and ease of use. Python is first and foremost a scripting language but can be compiled into computer-readable binary. The upside to all of this is that you can write programs in less lines of code than an equivalent C/C++ or Java program. (*Applications for Python*, n.d.)

12.2.2.1 Advantages of Python

- Easy to learn.
- Rich Ecosystem
- Scalability

12.2.2.2 Disadvantages of Python

- Performance
- Less Suitable for Real-Time Applications

12.2.3 Java

Java is a programming language and computing platform that's been around since 1995. It's used to create a wide range of digital applications. Java is known for its reliability and is often used in enterprise-level applications. (*What Is Java and Why Do I Need It?*, n.d.)

12.2.3.1 Advantages of Java

- Example programs are on the internet.
- It is safe.
- Many use java so there are a lot of tutorials and other helpful things available (*Java Ist Auch Eine Insel - Ullenboom 2001.Pdf*, n.d.)

12.2.3.2 Disadvantages of Java

- Hard to learn.
- It is slower than other languages like C#.
- Compared to other languages Java forces you to write a lot of code to work which can take a longer time and it can get confusing for later reviews

12.2.4 Ruby

Ruby on Rails is a popular web application framework that can handle HTTP requests and events effectively. It allows for an easy creation of a web-application because of the many features that allow you to work faster. It has a clean syntax which allows it to write and understand the code more effectively. (*Getting Started with Rails*, n.d.)

12.2.4.1 Advantages of Ruby

- Fast development
- High-Quality code

12.2.4.2 Disadvantages of Ruby

- Performance
- Scalability
- Learning Curve

12.2.5 PHP

PHP is a language designed for developing a web-application. Not only that it is also used for database-management and perform various server-side tasks. It is embedded within HTML code to create dynamic and interactive web applications. (*PHP: What Is PHP? - Manual*, n.d.)

12.2.5.1 Advantages of PHP

- Database support
- Integration with Web Servers
- Used for small or large web applications.

12.2.5.2 Disadvantages of PHP

- Not secure enough.
- It isn't consistent in terms of performance.
- Other languages can handle larger applications better.
- It doesn't have the newest features.

12.2.6 Go (Golang)

Go is an open-source programming language which was develop by Google in 2007. It is known for its simplicity, reliability, and efficiency and that is why it is mainly used for building distributed systems, web applications, and network servers. (*5672521.Pdf*, n.d.)

12.2.6.1 Advantages of Go

- Simple
- Reliable
- Efficient

12.2.6.2 Disadvantages of Go

- Bigger learning curve
- No GUI, it is not used for a user interface.

12.2.7 C#

C# is a programming language that was developed by Microsoft in the year 2000. It was developed do create Windows applications and web applications in an easier way. (*C# | Modern, Open-Source Programming Language for .NET*, n.d.)

12.2.7.1 Advantages of C#

- Easy Syntax
- Many Plugins to use.

12.2.7.2 Disadvantages of C#

- Takes time to learn.
- Takes time to develop a web-application.
- For larger projects it can cause some performance issues

12.3 HTTP ENDPOINT

Creating an HTTP endpoint is an important part for building a web application. An HTTP endpoint is a specific URL on a web server that can receive and responding to HTTP requests from clients, such as web browsers or other servers. HTTP endpoints can perform various tasks, such as retrieving data, storing data, processing requests, or performing actions, depending on how they are implemented.

And as the already listed options in 12.2 there are frameworks that can provide solutions to it. For JS/ Node.js it is Express, for Python it is Flask or Django, for Java it is Spring, for Ruby Rails and Sinatra. Go has a standard library which allows It to create HTTP endpoints, while on the other hand PHP has Laravel and Slim and C# has ASP.NET Core.

12.3.1 Examples

- Java Spring:
  - Pros:
    - Already familiar
    - Many options for development with the amount of library's
    - Good for Security with Spring Security
  - Cons:
    - Hard to understand.
    - Takes a lot of time to program the application.
  - Example: (*Getting Started | Building an Application with Spring Boot*, n.d.)
    - Create a project with all the dependency's
      For maven:

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

*fig. 29: Dependency for Java/Spring (Source: (*Getting Started | Building an Application with Spring Boot*, n.d.))*

- Create a create a class with a repository and Controller Class:

```java
package com.example.springboot;

import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class Application {

	public static void main(String[] args) {
		SpringApplication.run(Application.class, args);
	}

	@Bean
	public CommandLineRunner commandLineRunner(ApplicationContext ctx) {
		return args -> {

			System.out.println("Let's inspect the beans provided by Spring Boot:");

			String[] beanNames = ctx.getBeanDefinitionNames();
			Arrays.sort(beanNames);
			for (String beanName : beanNames) {
				System.out.println(beanName);
			}

		};
	}

}
```

*fig 30: Code for a Example Class: (Source: (*Getting Started | Building an Application with Spring Boot*, n.d.))*

- Repository: For Example, not needed
- Controller:

```java
package com.example.springboot;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

	@GetMapping("/")
	public String index() {
		return "Greetings from Spring Boot!";
	}

}
```

*fig 31: Code for an Example Controller (Source: (*Getting Started | Building an Application with Spring Boot*, n.d.))*

- Test:

```
@SpringBootTest
@AutoConfigureMockMvc
public class HelloControllerTest {

        @Autowired
        private MockMvc mvc;

        @Test
        public void getHello() throws Exception {
                mvc.perform(MockMvcRequestBuilders.get("/").accept(MediaType.APPLICATION_JSON))
                                .andExpect(status().isOk())
                                .andExpect(content().string(equalTo("Greetings from Spring Boot!")));
        }
}
```

*fig 32: Code for a Controller-Test (Source: (*Getting Started | Building an Application with Spring Boot*, n.d.))*

- Python Flask:
  - Pros:
    - Easy to understand and learn.
    - Many options
  - Cons:
    - For larger projects it is not well suited
    - Not as many security options as other languages
    - Needs time to write the code.
  - Example:

```
Diplomarbeit_Test.py
from flask import Flask, redirect, url_for, request

app = Flask(__name__)

@app.route('/succeess/<name>')
def success(name):
    return 'welcome %s' & name

@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        user = request.form['nm']
        return redirect(url_for('success', name=user))
    else:
        user = request.args.get('nm')
        return redirect(url_for('success', name=user))


if __name__ == '__main__':
    app.run(debug=True)
```

fig 33: Code for a Flask example (Source: ('Flask - (Creating First Simple Application)', 2017))

- JS/ Node.js Express:
  - Pros:
    - Easy to use.
    - Is very similar to JavaScript.
    - Has also many options for development
  - Cons:
    - Could lead to callbacks problems.
  - Example:

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
    res.send('Hello World!')
})

app.listen(port, () => {
    console.log('Example for listening on port ${port}')
})
```

*fig 34: Code for an Example HTTP Endpoint with Express (Source: (*Express 5.x - API Reference*, n.d.))*

- Ruby Rails:
  - Pros:
    - Can be fast at developing web-application.
  - Cons:
    - Needs a lot of time to learn.
    - Has a slower performance
  - Example:

```
class Article < ApplicationRecord
  include Visible

  has_many :comments

  validates :title, presence: true
  validates :body, presence: true, length: { minimum: 10 }
end
```

*fig 35: Example Class with Ruby on Rails (Source: (*Getting Started with Rails*, n.d.))*

```
<h1>Articles</h1>

Our blog has <%= Article.public_count %> articles and counting!

<ul>
  <% @articles.each do |article| %>
    <% unless article.archived? %>
      <li>
        <%= link_to article.title, article %>
      </li>
    <% end %>
  <% end %>
</ul>

<%= link_to "New Article", new_article_path %>
```

*fig 36: HTML File for the example (Source: (*Getting Started with Rails*, n.d.))*

```ruby
class CommentsController < ApplicationController
  def create
    @article = Article.find(params[:article_id])
    @comment = @article.comments.create(comment_params)
    redirect_to article_path(@article)
  end

  def destroy
    @article = Article.find(params[:article_id])
    @comment = @article.comments.find(params[:id])
    @comment.destroy
    redirect_to article_path(@article), status: :see_other
  end

  private
    def comment_params
      params.require(:comment).permit(:commenter, :body, :status)
    end
end
```

*fig 37: Controller with Ruby on Rails (Source: (*Getting Started with Rails*, n.d.))*

13 Solution variants and evaluation of the personal topic – Braun Valerian

This document area gives an overview of the solution variants and evaluation of the individual topic from Braun Valerian.

13.1 Utility analysis

Criteria		Java	Points	JS/ NodeJS	Points	Python	Points	Ruby	Points
Functionality	5	4	20	4	20	3	15	3	15
User-friendliness	5	3	15	4	20	4	20	4	20
Security	4	4	20	4	20	3	15	3	15
Technical Feasibility	2	4	20	4	20	3	15	3	15
Risks	2	3	15	3	15	3	15	2	10
Overall rating		90		**95**		80		75	

fig 38: Utility Analysis (Source: ERoots_Nutzwertanalyse_v2.xslx)

13.1.1 Criteria

Weight	
5	Very importan
4	important
3	Satisfactory
2	Not important
1	Only for additional features
Rating: 1-5	
5	Excellent
4	Good
3	Satisfactory
2	Sufficient
1	Poor

El

fig 39: Criteria for Utility Analysis 13.1 (Source: ERoots_Nutzwertanalyse_v2.xslx)

Explanation of the weightings and the rating for the different languages

Functionality	Evaluation of functionalities and features
User-friendliness	How user-friendly is the software?
Security	How secure is it to use the program
Technical Feasibility	Is there a significant amount of preparation required to use the program
Risks	How often do problems occur, and how do they impact the project

fig 40: Explanation for the Criteria Points for the Utility Analysis 13.1 (Source: ERoots_Nutzwertanalyse_v2.xslx)

Clearer description of what the criteria is.

13.2 Conclusion for the Language

In conclusion, when assessing the suitability of programming languages for building a web application, the allocation of points for functionality, user-friendliness, security, technical feasibility, and risks, JavaScript/ Node.js, Java, Python, Ruby, each have their strengths and weaknesses, and the distribution of points reflects their relative performance in these criteria.

JavaScript/ Node.js and Java are strong contenders, scoring well in functionality, security, and technical feasibility. Python and Ruby excel in user-friendliness, making them good choices for rapid development.

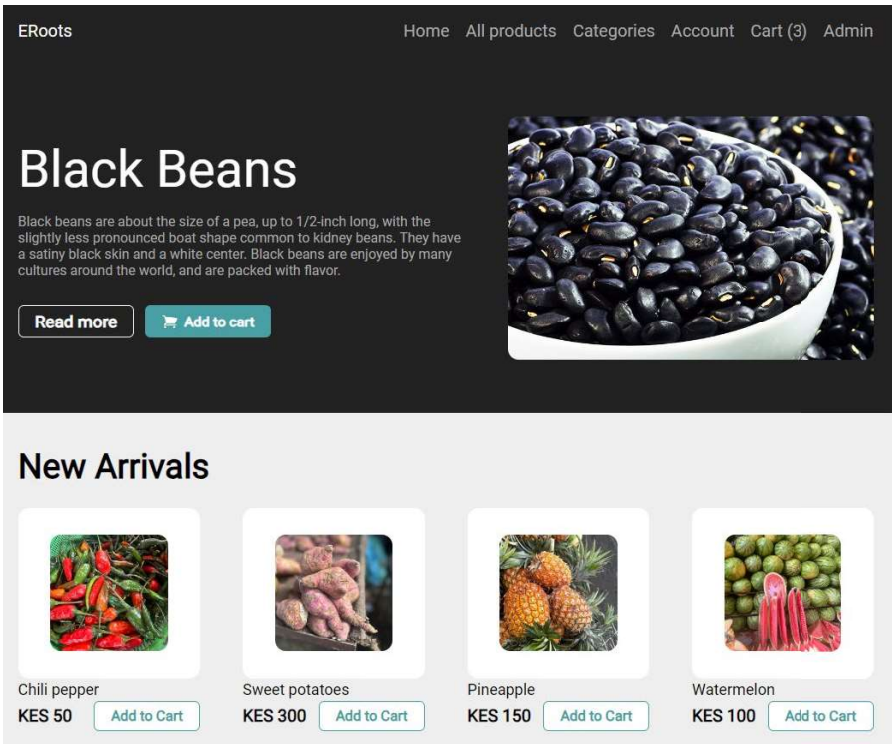
But for the project the most suitable option will be JavaScript/ Node.js.

14 Afterword

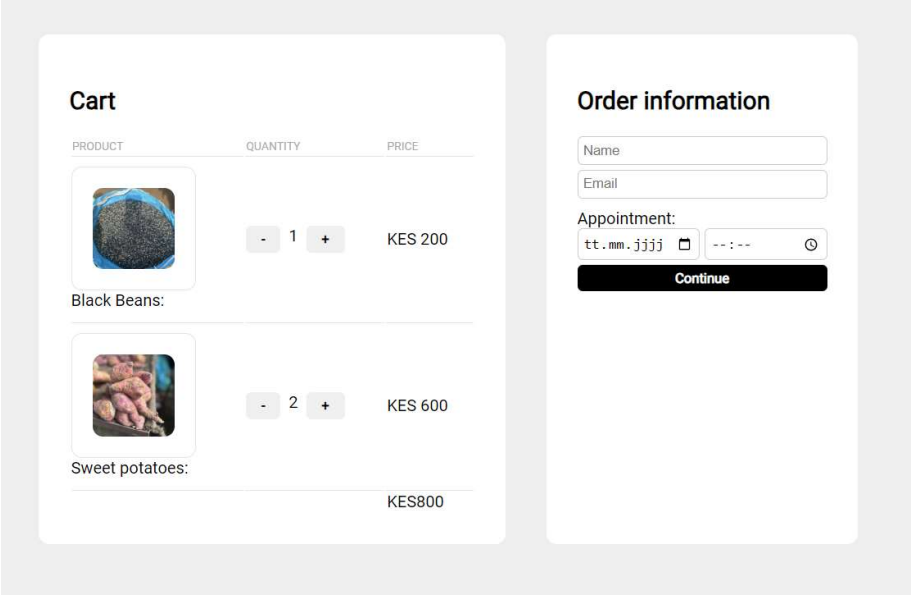
In this preliminary study, the project team explored various stacks and technologies that could be used to identify their advantages and disadvantages, with the ultimate goal of selecting the most suitable technology.

14.1 Current Prototype

After Trial and Error, we were able to make a prototype, that has some of the functions in our chosen system architecture. Under the domain eroots.store we were able to deploy our current repository status.



All products can be put in the cart with the total value being calculated automatically. Other features are being planned and according to our milestone plan the prototype will be finished at the 8th of January.



14.2 Literature Cited

5672521.pdf. (n.d.). Retrieved 11 October 2023, from https://epub.jku.at/obvulihs/download/pdf/5672521?originalFilename=true

About. (n.d.). Node.Js. Retrieved 19 October 2023, from https://nodejs.org/en/about

Angular. (n.d.). Retrieved 11 October 2023, from https://angular.io/

Applications for Python. (n.d.). Python.Org. Retrieved 20 October 2023, from https://www.python.org/about/apps/

Authorization Code Grant | Login with Amazon. (n.d.). Retrieved 19 October 2023, from https://developer.amazon.com/docs/login-with-amazon/authorization-code-grant.html

Authorizing OAuth apps. (n.d.). GitHub Docs. Retrieved 19 October 2023, from https://ghdocs-prod.azurewebsites.net/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps

Backbone.js. (n.d.). Retrieved 11 October 2023, from https://backbonejs.org/

C# | Modern, open-source programming language for .NET. (n.d.). Microsoft. Retrieved 20 October 2023, from https://dotnet.microsoft.com/en-us/languages/csharp

Datenbankmanagementsystem (DBMS) erklärt. (2020, May 18). IONOS Digital Guide. https://www.ionos.at/digitalguide/hosting/hosting-technik/datenbankmanagementsystem-dbms-erklaert/

Dieterich, P. (2017). *Evaluierung verschiedener Technologien zur Speicherung und Verknüpfung von strukturierten und unstrukturierten Daten* [bachelorThesis]. http://elib.uni-stuttgart.de/handle/11682/9325

Education, I. C. (2021, June 29). Structured vs. Unstructured Data: What's the Difference? *IBM Blog*. https://www.ibm.com/blog/structured-vs-unstructured-data/

Ember.js—A framework for ambitious web developers. (n.d.). Retrieved 11 October 2023, from https://emberjs.com/

Express 5.x—API Reference. (n.d.). Retrieved 19 October 2023, from https://expressjs.com/en/5x/api.html

Flask—(Creating first simple application). (2017, October 23). *GeeksforGeeks*. https://www.geeksforgeeks.org/flask-creating-first-simple-application/

Getting Started | Building an Application with Spring Boot. (n.d.). Getting Started | Building an Application with Spring Boot. Retrieved 19 October 2023, from https://spring.io/guides/gs/spring-boot/

Getting Started with Rails. (n.d.). Ruby on Rails Guides. Retrieved 19 October 2023, from https://guides.rubyonrails.org/getting_started.html

Hardt, D. (2012). *The OAuth 2.0 Authorization Framework* (Request for Comments RFC 6749). Internet Engineering Task Force. https://doi.org/10.17487/RFC6749

Helmut Eirund, Herbert Kopp, & Axel Viereck (Eds.). (n.d.). *Datenbanken und SQL Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL* (5., aktualisierte und erweiterte Auflage). Springer Vieweg.

Hoffman, A. (2020). *Web Application Security—Exploitation and Countermeasures for Modern Web Applications* (Third Release). O'Reilly Media, Inc.

Home | Aurelia. (n.d.). Retrieved 11 October 2023, from https://aurelia.io/

Java ist auch eine Insel—Ullenboom 2001.pdf. (n.d.). Retrieved 12 October 2023, from https://www.raffia.ch/content/docplus/Java%20ist%20auch%20eine%20Insel%20-%20Ullenboom%202001.pdf

Kofler, M., Zingsheim, A., Gebeshuber, K., Widl, M., Aigner, R., Hackner, T., Kania, S., Kloep, P., & Neugebauer, F. (Eds.). (2018). *Hacking & Security—Das umfassende Handbuch* (1. Auflage 2018). Rheinwerk Verlag.

Leiba, B. (2012). OAuth Web Authorization Protocol. *IEEE Internet Computing*, *16*(1), 74–77. https://doi.org/10.1109/MIC.2012.11

Mithril.js. (n.d.). Retrieved 11 October 2023, from https://mithril.js.org/

MongoDB: Die Plattform Für Anwendungsdaten. (n.d.). MongoDB. Retrieved 11 October 2023, from https://www.mongodb.com/de-de

Moser, C. (2012). User Experience Design. In C. Moser (Ed.), *User Experience Design: Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern* (pp. 1–22). Springer. https://doi.org/10.1007/978-3-642-13363-3_1

MySQL. (n.d.). Retrieved 11 October 2023, from https://www.mysql.com/

MySQL, A. B. (2001). *MySQL*. http://box.cs.istu.ru/public/docs/other/_New/Books/Data/DB/MySQL/MySQL%20Bible.pdf

NoSQL – Funktion und Vorteile von NoSQL-Datenbanken. (2023, June 19). IONOS Digital Guide. https://www.ionos.at/digitalguide/hosting/hosting-technik/nosql/

Olanrewaju, R. F., Khan, B. U. I., Morshidi, M. A., Anwar, F., & Kiah, M. L. B. M. (2021). A Frictionless and Secure User Authentication in Web-Based Premium Applications. *IEEE Access*, *9*, 129240–129255. https://doi.org/10.1109/ACCESS.2021.3110310

PHP: What is PHP? - Manual. (n.d.). Retrieved 20 October 2023, from https://www.php.net/manual/en/intro-whatis.php

PostgreSQL: About. (n.d.). Retrieved 11 October 2023, from https://www.postgresql.org/about/

React. (n.d.). Retrieved 11 October 2023, from https://react.dev/

React – Eine JavaScript Bibliothek zum Erstellen von Benutzeroberflächen. (n.d.). Retrieved 11 October 2023, from https://de.legacy.reactjs.org/

Regez, A., Dirksen, U., & der PHBern, A. K. (2023). *Was sind Daten?* https://ttim.phbern.ch/wp-content/uploads/2023/06/Artikel-1_Was-sind-Daten_V3.pdf

Spitta, T., & Bick, M. (2009). Unstrukturierte Daten. In T. Spitta & M. Bick, *Informationswirtschaft* (pp. 161–180). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-85116-5_9

Stop fighting with frameworks and start shipping real apps—Meteor.js. (n.d.). Retrieved 11 October 2023, from https://meteor-universe.meteorapp.com/

Svelte • Cybernetically enhanced web apps. (n.d.). Retrieved 11 October 2023, from https://svelte.dev/

Syed Idrus, S. Z., Cherrier, E., Rosenberger, C., & Schwartzmann, J.-J. (2013). A Review on Authentication Methods. *Australian Journal of Basic and Applied Sciences*, *7*(5), 95–107.

Verwenden von OAuth 2.0 für den Zugriff auf Google APIs | Authorization. (n.d.). Google for Developers. Retrieved 19 October 2023, from https://developers.google.com/identity/protocols/oauth2?hl=de

Vue.js—The Progressive JavaScript Framework | Vue.js. (n.d.). Retrieved 11 October 2023, from https://vuejs.org/

Weihrich, H. (1982). The TOWS matrix—A tool for situational analysis. *Long Range Planning*, *15*(2), 54–66. https://doi.org/10.1016/0024-6301(82)90120-0

What Is Authentication? Definition and Methods | Microsoft Security. (n.d.). Retrieved 17 October 2023, from https://www.microsoft.com/en-us/security/business/security-101/what-is-authentication

What is Java and why do I need it? (n.d.). Retrieved 20 October 2023, from https://www.java.com/en/download/help/whatis_java.html

Writing Your First E2E Test | Cypress Documentation. (2023, August 29). https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test

PROJECT PROPOSAL

Project title	Evergreen Roots
Project abbreviation	ERoots
Year / Class	2023/24 / 5AHBGM
Project manager	Njeri Everlyn
Project sponsor	Njuguna Pennina
Supervising teacher	Schwarz Marcel (SWM)
Project start date	01.06.2023
Project finish date	04.04.2024

Task description	Management Summary
	Evergreen Roots is a project that offers an eCommerce web-application for a Kenyan Women Self Help Group, which allows them to upload their farmed goods. The Web-Application provides an admin dashboard to manage the products. The front interface offers a wide-range product page, and ordering page, a history view, and a search functionality. The web-application offers the creation of User Accounts, with which they can book appointments to be able to inspect the goods before they are purchased.

Initial situation	background, problem, boundary condition
	In Kiambu, Kenya a Women Self Help Group wants to improve their sales of locally farmed goods in order to expand their consumer base. To achieve this, they need a web-based platform that enables online ordering of their products, with payment to be made upon pick-up.
	Market analysis
	In other countries there are many providers that enable the service of ordering goods. In Kenya there aren't any/many. Some examples include: • JioMart (India): https://www.jiomart.com/c/groceries/fruits-vegetables/fresh-vegetables/229 It is an online shopping market for food, baby products and books. It has different filters to search for products. • Fresh Produce Shoppe (India): https://www.freshproduceshoppe.com/ An online shopping service with a large variety of foods. • Amazon (worldwide): https://www.amazon.de/ An online shopping service it has a large variety on different kind of goods. It is also the most used online shopping service in the world.

Usage goals	Objectives from the point of view of the project client (Partners)
	• Extended customer base. • A reliable and easy to use web-application.

Targets (of the project)	Aim

	• Client interface
	  ○ Create product page with all and one product.
	  ○ User
	  ▪ create and edit a user account.
	  ○ be able to order products.
	  ○ view previous orders with a history view.
	  ○ implement a search functionality.
	• Admin Interface
	  ○ Create, Delete or Edit products with data persistence.
	  ○ Create, Delete or Edit categories with data persistence.
	• Book appointments Appointment booking (reservations)
	• Store Products, categories and admin and user login data in the database.

	Can-do goals (Options)
	• Introduce Reviews from user for products. • By checkout be able to use coupons and discount codes. • Add products to a wish list. • Register as a user for a newsletter.
	Non-goals (Demarcation)
	• Online payment system

Project contents	Result
	An eCommerce web-application with an admin dashboard to upload new products and a client dashboard to order products and make an appointment for the evaluation of the farmed good getting sold.
	Content components (Objects of observation)
	• Admin dashboard • Product management • Ordering management • Security • Client dashboard • Product page • Database • Testing

Risks	Critical success factors
	• External databases link doesn’t work. • Deployment of the web-application. • Integration between frontend and backend

Main	Main (Phases)
	• Project management • Analysis • Design • Implementation • Testing

Milestones	Milestone	Deadline
	Project proposal signed by the project sponsor (project start)	22.09.2023
	product specifications finished	06.11.2023
	technical specifications finished	20.12.2023
	Prototype 1 completed	08.01.2024
	Project completion protocol signed by the project sponsor	29.02.2024 (04.04.24)

Effort / Costs	Position	Quantity	Remark
	Effort	720 hours	180 hours per member
	Investment	1€	No investments
	External costs	~10€	Hosting fees

Project organization	Role	Name	Individual tasks
	Project sponsor	Njuguna Penina	catalog of available agricultural produce
	Project manager	Njeri Everlyn	how can data persistence be planned and implemented in e-commerce with authentication?
	Project team member	Braun Valerian	how to implement an application, HTTP endpoint that receives events and development?
	Project team member	Andrei Elias	planning and implementing/developing of deployment, testing and quality assurance
	Project team member	Paukner Sarah-Jessica	providing and implementing security, planning and development of the user experience of the web application

	Supervising teacher	Schwarz Marcel	supervision and feedback
Signatures	Place / Date	Signature	
Project manager Njeri Everlyn	Kenya,		
Project team member Braun Valerian	Vienna,		
Project team member Andrei Elias	Vienna,		
Project team member Paukner Sarah	Vienna,		
Project sponsor Njuguna Penina	Kenya,		
Supervising teacher Schwarz Marcel	Vienna,		

Project manual

Evergreen Roots

Version: 2.0
Project manager: Everlyn Njeri
Date: 20.11.2023

Content

1 Project plans 5
1.1 Project Assignment 5
1.2 Project Objectives (objectives, non-objectives) 6
1.3 Description of Pre- and Post Project Phase 7
1.4 Project Environment Analysis 8
1.5 Project Organisation Chart 10
1.6 Plan of Objects of Consideration of the Project 12
1.7 Work Breakdown Structure (WBS) 13
1.8 Project Work-Package Specification 14
1.9 Project Responsibility Matrix 16
1.10 Milestoneplan 18
1.11 Project Bar Chart ..... **Error! Bookmark not defined.**
1.12 Resource Plan 21
1.13 Project Cost Plan 22
1.14 Project Communication 23
1.15 Project „Rules" 24
1.16 Project Risk Analysis 25
1.17 Project Documentation 26

2 Project Controlling 27
2.1 Project Status Report 27

3 Project Close Down 28
3.1 Project Close Down Report 28
3.2 Minutes – Project Close Down 29

Document versions

Versionno.	Date	Change	Author
0.1	13.09.2023	Layout	Braun Valerian
0.2	27.09.2023	1.16	Braun Valerian
0.3	01.10.2023	1.6, 1.1, 1.18	Braun Valerian
0.4	02.10.2023	1.18	Braun Valerian
0.6	11.10.2023	1.4	Braun Valerian
0.7	25.10.2023	Various Mistakes corrected	Braun Valerian
0.8	06.11.2023	Project Environment Table, inserted 1.8, 1.10	Njeri Everlyn
			Andrei Elias
1	13.11.2023	1.9, 1.10, 1.3	Braun valerian
		1.12, 1.16, 1.5	Njeri Everlyn
		1.7, 1.8, 1.9, 1.12, 2.1	Paukner Sarah Jessica
		1.9	Andrei Elias
2	20.11.2023	1.11	Braun Valerian

Distribution list

Name	Organisational Unit	Role in project	Telephone	E-mail
Njeri Everlyn	5AHBGM	Project manager		Nje20157@spengergasse.at
Braun Valerian	5AHBGM	Project team member		And20168@spengergasse.at
Andrei Elias	5AHBGM	Project team member		Bra20173@spengergasse.at
Paukner Sarah Jessica	5AHBGM	Project team member		Pau20159@spengergasse.at
Hell Florian	Teacher	Project coach		
Schwarz Marcel	Teacher	Project supervisior		
Pennina Njuguna	Mutundu Women Self Help Group	Project client		

1 Project plans

1.1 Project Assignment

PROJECT-ASSIGNMENT	
Project start event: • Project application signed	**Project start date:** • 01.06.2023
project close down event in terms of content: • Web application published/ finished **Formal project close down event:** • Acceptance protocol signed	**Project close down dates:** • 04.04.2024 •

Project objectives:
• Client interface
  • product page with all products created
  • one product page created
  • User
  • user created and edited
  • products ordered
  • view previous orders with a history view
  • implement a search functionality
• Admin Interface
  • products with data persistence created, edited and deleted
  • categories with data persistence created, edited and deleted
• appointments booked (reservations)
• Products, categories and admin and user login data in the database stored
	Non-objectives:
	• Online payment system

Main tasks (Project phases): • Project Management • Analyse • Design • Implementation • Test	**Project resources and costs*:** resource/type of cost | unit | Costs Hosting fees | € | ~10€ Work hours | hours | 150 hours per member
Project owner: • Schwarz Marcel	**Project manager:** • Njeri Everlyn
Project team members: • Paukner Sarah Jessica • Andrei Elias • Braun Valerian	
---------- <*Name*>, (Project owner)	---------- *<Name>* (Project manager)

1.2 Project Objectives (objectives, non-objectives)

Evergreen Roots PROJECT OBJECTIVES		
Type of objective	**Project objectives**	**Adjusted project objectives as of…**
objectives: • **Main objectives**	* Client interface * product page with all products created * one product page created * User * products ordered * view previous orders with a history view * implement a search functionality * Admin Interface * products with data persistence created, edited and deleted * categories with data persistence created, edited and deleted * appointments booked (reservations) * Products, categories and admin and user login data in the database stored	
Additional objectives	* user created and edited * Reviews from user for products introduced * use coupons and discount codes usable * products to a wish list added * user for a newsletter registered	
Non-objectives	* Online payment system created	

1.3 Description of Pre- and Post Project Phase

Evergreen Roots	DESCRIPTION OF PRE- AND POST-PROJECT PHASE
1) Pre-project phase	
	What triggered the project? The trigger for the Evergreen Roots project was the initiative by a Women Self Help Group in Kiambu, Kenya, aiming to enhance the sales of locally farmed goods. The group identified the need for a web-based platform to facilitate online ordering of their products, with the convenience of payment upon pick-up.
	Relevant documents for the project Project Proposal * Requirements Specification * Project Manual * Project Requirements Document * Project Acceptance Protocol * Meeting Minutes
	Experience from similar projects * Project management is the same. * The Project team is the same. * New Same programming environment * Similar programming language
2) Post-project phase	
	What will happen after the project has ended? (follow-up activities, further projects, …)? * After completing the project, the Self-Help Group can choose to use it

1.4 Project Environment Analysis

Evergreen Roots

PROJECT ENVIRONMENT GRAPHIC

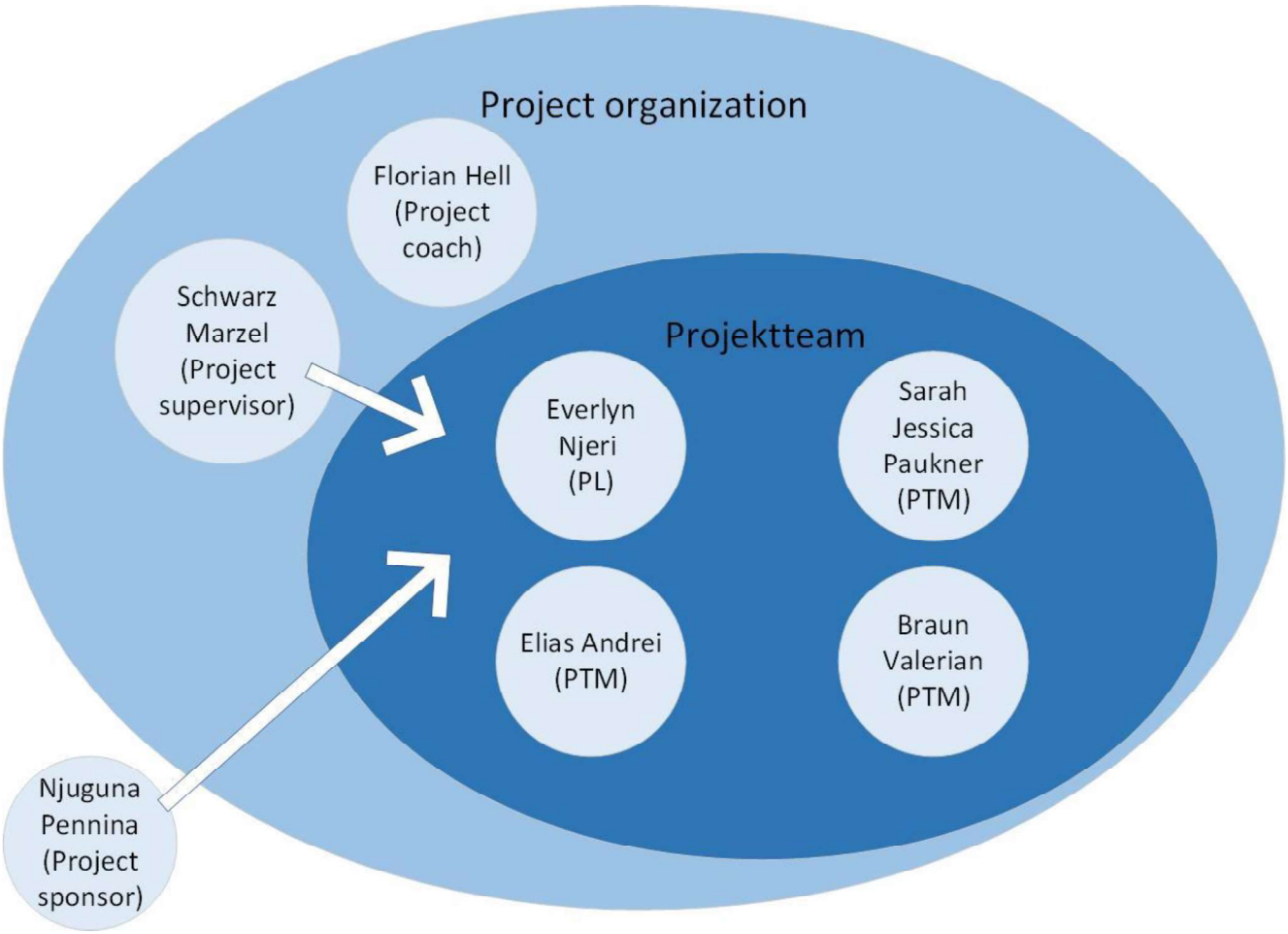
Evergreen Roots

PROJECT ENVIRONMENT TABLE

Environment	**Relationship** (potential/conflict)	**Measures**	**Who / when** **PSP Code**
Schwarz Marcel (project supervisor)	Neutral	Project Discussion every 2 weeks, frequent Communication	Project Team + Project Supervisor once every 2 weeks
Hell Florian (project coach)	Neutral	Communicate regularly	Project Team on Monday in School
Njuguna Pennina	Neutral	Weekly progress updates	Project Team on Monday in School
Njeri Everlyn	Positive	Weekly progress updates	Project Team on Monday in School
Paukner Sarah	Positive	Weekly progress updates	Project Team on Monday in School

Andrei Elias	Positive	Weekly progress updates	Project Team on Monday in School
Braun Valerian	Positive	Weekly progress updates	Project Team on Monday in School

1.5 Project Organisation Chart

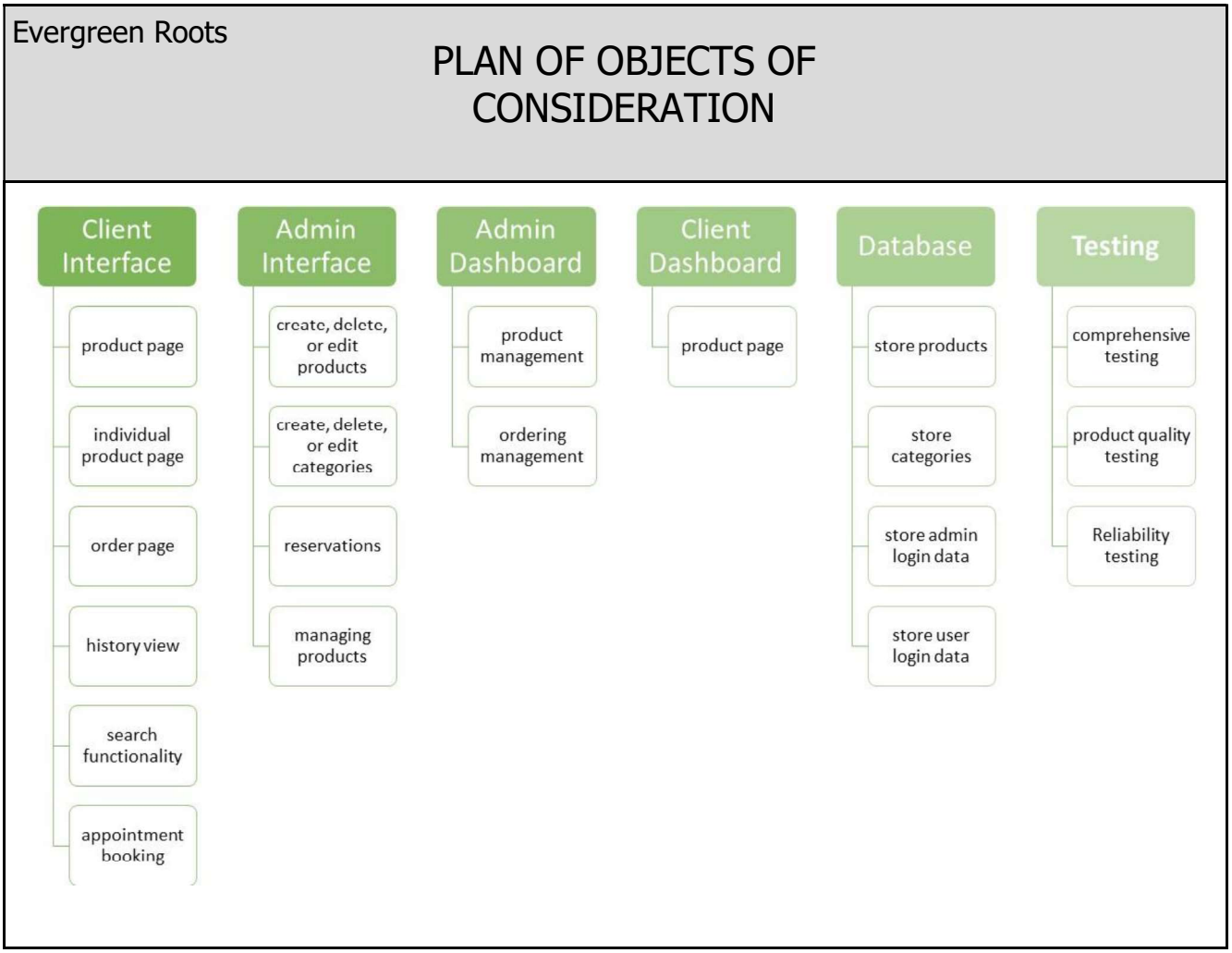


Hell Florian

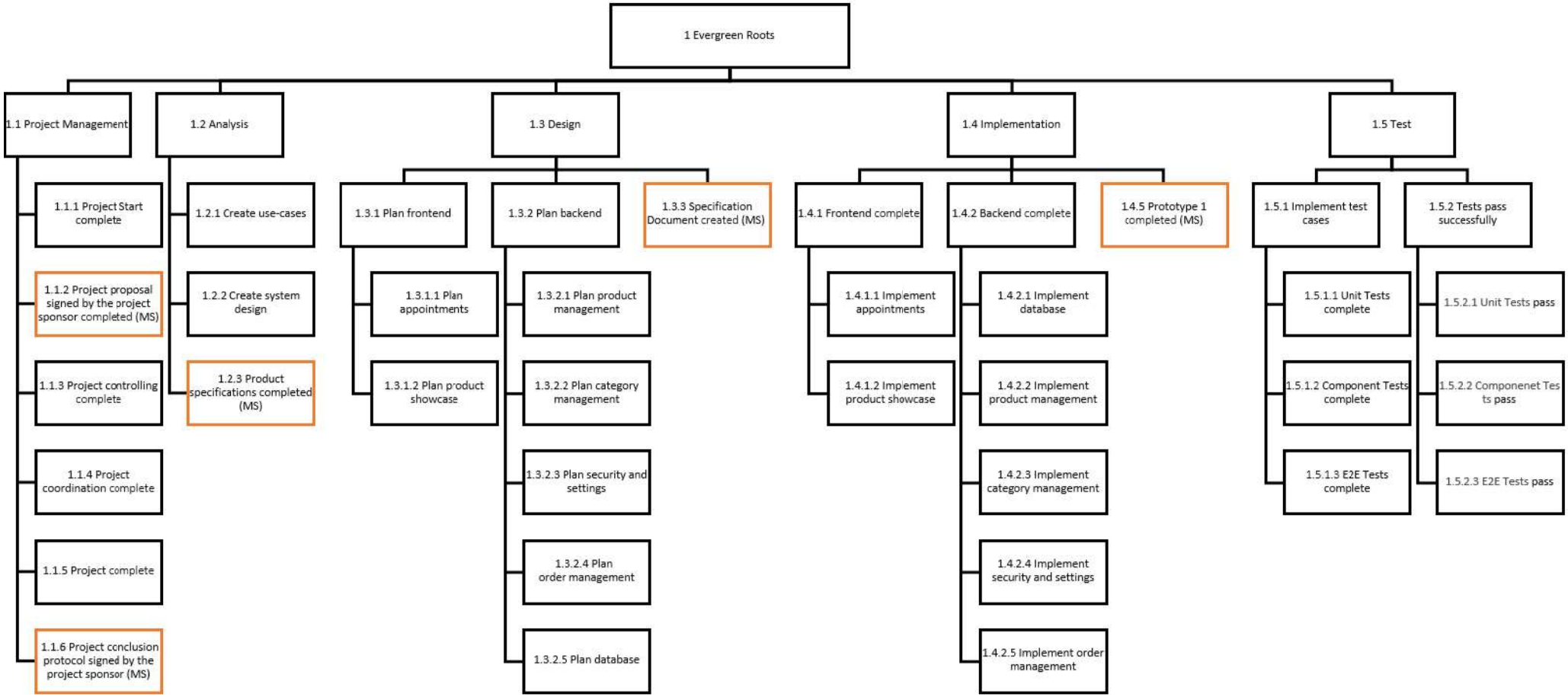
PROJECT-ORGANISATION

Role in Project	Field of duties/Skills	Name
Project supervisor	supports and helps with the project	Schwarz Marzel
Project sponsor	providing a catalog of available agricultural produce	Njuguna Pennina
Project coach	Supports and controls the documents	Hell Florian
Project leader	how can data persistence be planned and implemented in e-commerce with authentication?	Njeri Everlyn
Project team members	providing and implementing security, planning and development of the user experience of a web application, project management.	Paukner Sarah Jessica
Project team members	planning and implementing/developing of deployment, testing and quality assurance	Andrei Elias
Project team members	how to implement an application, HTTP endpoint that receives events and development?	Braun Valerian

1.6 Plan of Objects of Consideration of the Project



1.7 Work Breakdown Structure (WBS)



1.8 Project Work-Package Specification

Evergreen Roots	PROJECT WORK-PACKAGE SPECIFICATION
1.4.2.5 Implement Order Management	**WP Content** * Implement the ordering process and functionalities. * Save Orders in Database * Send email to the user about the order.
	Non-WP Content * Payment system implemented.
	WP Result * Ordering process and functionalities implemented.
	Progress Measurement * Ordering process (30%) * Functionalities (80%) * Sending order email (100%)

Evergreen Roots	PROJECT WORK-PACKAGE SPECIFICATION
1.3.2.3 Plan Security and Settings	**WP Content** * Plan user account creation and editing functionalities. * Plan methods to enforce user authentication. * Plan methods to enforce user authorization. * Plan methods to implement data encryption.
	Non-WP Content * Implement overly complex security measures that could impact user experience without providing significant benefits.
	WP Result * User account management functionalities planned. * user authentication planned. * user authorization planned. * data encryption planned.
	Progress Measurement * User account management functionalities planned (30%) * user authentication planned. (50%) * user authorization planned. (70%) * data encryption planned. 100%)

Evergreen Roots	PROJECT WORK-PACKAGE SPECIFICATION
1.3.1.2 Plan Product Showcase	**WP Content** * Plan the design and implementation of the product showcase.
	Non-WP Content * overcomplicating the user interface with unnecessary features.
	WP Result * Product showcase design and implementation plan defined
	Progress Measurement * Design planned (50%) * Implementation planned (100%)

Evergreen Roots	PROJECT WORK-PACKAGE SPECIFICATION
1.5.1 Implement Test Cases	**WP Content** * Implement Test Cases
	Non-WP Content * Including test cases that simulate unrealistic or extreme user scenarios that are not representative of typical usage.
	WP Result * Test cases covering the main functions of the project
	Progress Measurement * Test Cases implemented (100%)

1.9 Project Responsibility Matrix

Evergreen Roots								
WBS-Code	WP-Title / Roles & Environment	Njuguna Pennina	Schwarz Marcel	Hell Florian	Njeri Verlyn	Paukner Sarah	Andrei Elias	Braun Valerian
1.1	**Project management**							
1.1.1	Project Start complete			I	R	C	C	C
1.1.2	Project proposal sign by the project sponsor completed (MS)							
1.1.3	Project controlling complete				C	R	C	C
1.1.4	Project coordination complete				C	C	R	C
1.1.5	Project complete	I	I	I	C	C	C	R
1.1.6	Project conclusion protocol signed by project sponsor (MS)							
1.2	**Analysis**							
1.2.1	Create Use-Cases							R
1.2.2	Create system design				R	C		
1.2.3	Product specifications completed (MS)							
1.3	**Design**							
1.3.1	Plan Frontend							
1.3.1.1	Plan appointments				R			
1.3.1.2	Plan product showcase							R
1.3.2	Plan Backend							
1.3.2.1	Plan product management				R	C	C	C
1.3.2.2	Plan category management				R	C	C	C
1.3.2.3	Plan security and settings					R		
1.3.2.4	Plan order management				C	C	R	C
1.3.2.5	Plan database				R	C	C	C
1.3.3	Specification Document created (MS)							
1.4	**Implementation**							
1.4.1	Frontend complete		I					
1.4.1.1	Implement appointments				R	C	C	C
1.4.1.2	Implement product showcase				C	C	R	C
1.4.2	Backend complete		I					
1.4.2.1	Implement database				R			
1.4.2.2	Implement product management				C	R	C	C
1.4.2.3	Implement category management				C	C	R	C
1.4.2.4	Implement security and settings					R		
1.4.2.5	Implement order management						R	C

1.4.5	Prototype 1 completed (MS)							
1.5	**Test**							
1.5.1	Implement test cases							
1.5.1.1	Unit Tests complete						R	
1.5.1.2	Component Tests complete						R	
1.5.1.3	E2E Tests complete						R	
1.5.2	Tests pass successfully							
1.5.2.1	Unit Tests pass						R	
1.5.2.2	Component Tests pass						R	
1.5.2.3	E2E Tests pass						R	

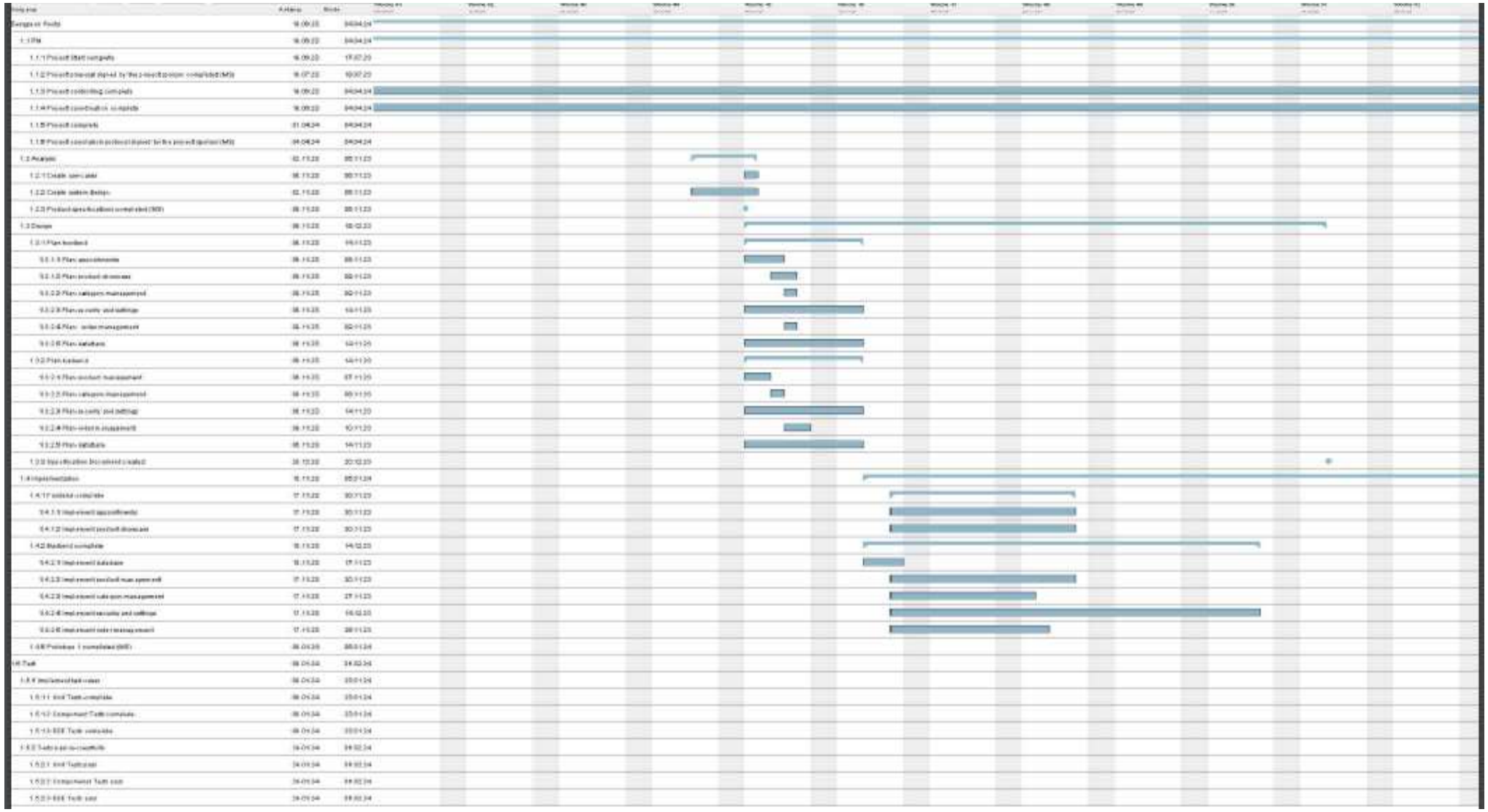
Functions
R …….Responsible
C …….Contribution
I ……..has to be informed

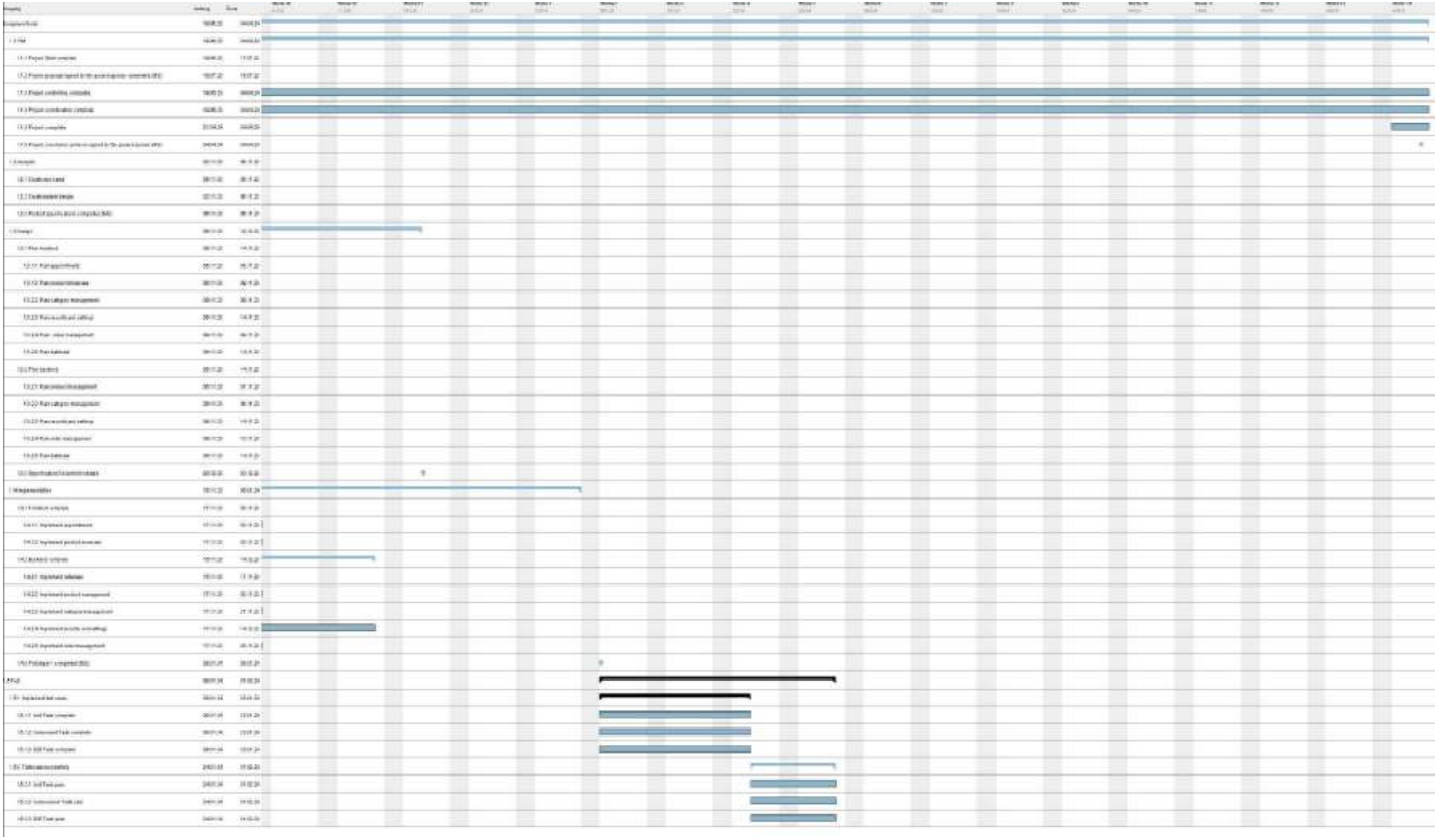
1.10 Milestoneplan

Evergreen Roots MILESTONEPLAN				
WBS-Code	**Milestone**	**Plan date**	**Revised date**	**Actual date**
	Project proposal signed by the project sponsor (project start)	22.09.2023		07.07.2023
	product specifications finished	06.11.2023		06.11.2023
	technical specifications finished	20.12.2023		
	Prototype 1 completed	08.01.2024		
	Project completion protocol signed by the project sponsor	29.02.24		

*In order of plan dates.

1.11 Project Bar Chart





1.12 Resource Plan

Evergreen Roots	Resource Plan					
WBS-Code	**Phase/Work-package**	**Type of resource**	**Planned quantity (days)**	**Revised quantity (days)**	**Actual quantity (days)**	**Deviation (days)**
	Project management	working hours	100:00		56:15	55:45
	Analysis	working hours	50:00		14:00	43:45
	Design	working hours	50:00		1:10	48:50
	Implementation	working hours	300:00		126:00	189:55
	Test	working hours	100:00		6:00	95:45
Total	Deadline: 13.11.2023		600:00		203:25	434:00

*as of 13.11.2023

1.13 Project Cost Plan

*to be completed

Evergreen Roots	PROJECT COST PLAN				
WBS-Code, WP-Title	Type of Cost	Planned cost	Revised cost	Actual cost	Deviation
	• Personnel				
	• Material				
	• External services				
	• Other				
	Total				
	• Personnel				
	• Material				
	• External services				
	• Other				
	Total				
	• Personnel				
	• Material				
	• External services				
	• Other				
	Total				
Project cost					

1.14 Project Communication

Evergreen Roots	PROJECT-COMMUNICATION			
Title	**Objectives, Content**	**Participants**	**Schedule**	**Location**
Project owner meeting	• project status • decisions • acceptance of progress report	Project sponsor, Project leader		
Project controlling meeting	• project status • controlling of tasks, schedule, resources, costs • controlling of project environments • social controlling • prepare proposal for decision	Project leader, Project team, Project coach		
Project controlling meeting	* project status * controlling of tasks, schedule, resources, costs * controlling of project environments * social controlling * prepare proposal for decision *	Project leader, Project team, Project coach		
	*			

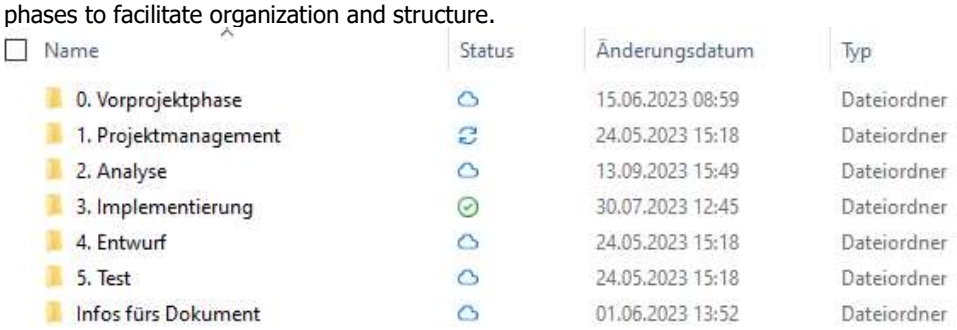
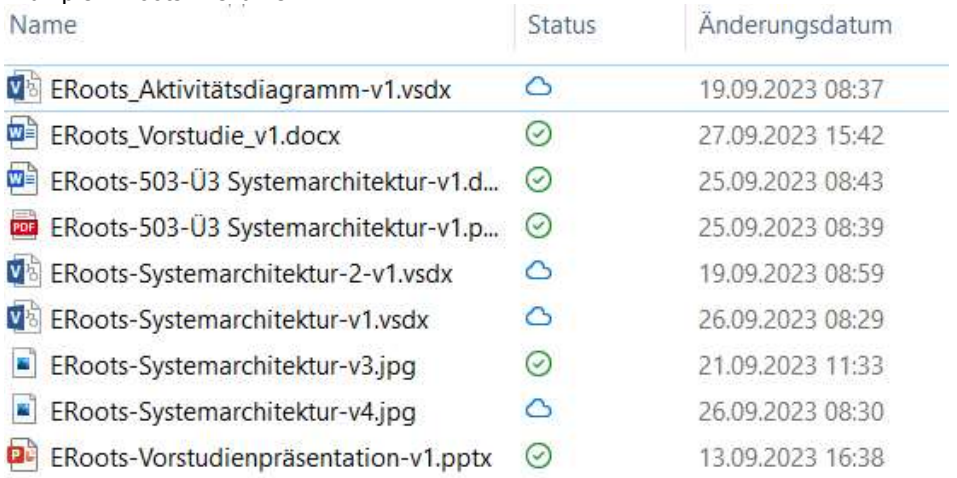
1.15 Project „Rules"

- One should adhere to deadlines and promptly upload files to the cloud.
- Communication among team members:
- When there are open questions or uncertainty, team members should reach out to their colleagues and assist each other.
- Every team member must complete their tasks by the specified deadlines.
- Regular meetings to monitor the project and take action if complications arise.

1.16 Project Risk Analysis

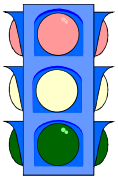
		PROJECT RISK ANALYSIS							
WBS-Code	WBS-description	Risk-description	Priority	Benefit /Cost	Probability	Risk-value	Delay	Avoiding action	Cost for minimizing risk
(Code)	*(Text)*	*(Text)*	*(Range)*	*(hours)*	*(Percentage)*	*(hours)*	*(Days)*	*(Text)*	*(hours)*
1.3.1.1	Plan appointments	Scheduling conflicts with appointments	5	2	20	0.4	0	Make sure conflicts in schedules are solved right away.	10
1.3.1.2	Plan product showcase	Technical issues during product showcase	6	4	30	1.8	0	Check the technical setup before showing the product.	4
1.3.2.1	Plan product management	Issues with product data integration	4	3	25	0.75	0	Test how well product information is put together.	5
1.3.2.2	Plan category management	Categories could not connect to products	6	4	30	1.8	0	While writing the computer code, try out different categories and create sample data to make sure it works.	7
1.3.2.3	Plan security and settings	Security vulnerabilities in the settings	5	3	15	0.75	0	Keep checking and updating the security plans regularly.	3
1.4.2.2	Implement product management	Issues with product implementatio n	4	4	35	1.4	0	Check how well the product works to find and fix any issues.	8
1.4.2.3	Implement category management	Orders are not saved in the database	3	5	40	1.2	0	Connect the code to the database and look for mistakes.	4
1.4.2.4	Implement security and settings	Security vulnerabilities in implementatio n	5	4	30	1.8	0	Regularly go over and update the security plans.	3
1.4.2.5	Implement order management	Issues with order processing and storage	4	3	25	0.75	0	Make sure the system for handling orders and storing information is strong and reliable.	6
Project total									50

1.17 Project Documentation

Area	Description
File	Microsoft Teams integrated with OneDrive, where files are stored in the "Diplomarbeit" folder. This allows every project member to access the project folder and be authorized to upload, edit, and delete files. Within the "Diplomarbeit" folder, there are additional subfolders for different project phases to facilitate organization and structure.  Name / Status / Änderungsdatum / Typ 0. Vorprojektphase — 15.06.2023 08:59 — Dateiordner 1. Projektmanagement — 24.05.2023 15:18 — Dateiordner 2. Analyse — 13.09.2023 15:49 — Dateiordner 3. Implementierung — 30.07.2023 12:45 — Dateiordner 4. Entwurf — 24.05.2023 15:18 — Dateiordner 5. Test — 24.05.2023 15:18 — Dateiordner Infos fürs Dokument — 01.06.2023 13:52 — Dateiordner
Access Authorisation	Each project member has the permission to upload, edit, and delete files in the shared folder.
Naming convention	The filename should be appropriate to the file and should have a naming structure to avoid disorder and confusion. At the beginning of the filename, our abbreviation "ERoots" should be used to identify the file as a project work. Furthermore, there should always be a version number specified. Example: ERoots-Filename-vX  Name / Status / Änderungsdatum ERoots_Aktivitätsdiagramm-v1.vsdx — 19.09.2023 08:37 ERoots_Vorstudie_v1.docx — 27.09.2023 15:42 ERoots-503-Ü3 Systemarchitektur-v1.d... — 25.09.2023 08:43 ERoots-503-Ü3 Systemarchitektur-v1.p... — 25.09.2023 08:39 ERoots-Systemarchitektur-2-v1.vsdx — 19.09.2023 08:59 ERoots-Systemarchitektur-v1.vsdx — 26.09.2023 08:29 ERoots-Systemarchitektur-v3.jpg — 21.09.2023 11:33 ERoots-Systemarchitektur-v4.jpg — 26.09.2023 08:30 ERoots-Vorstudienpräsentation-v1.pptx — 13.09.2023 16:38
Rules	• Every project member needs to have OneDrive installed to ensure that everyone has access to the shared folder. • It's essential to avoid uploading duplicate files. • Adhere to the naming convention. • Folders or files should only be deleted after discussion with the entire team. • Continuously save intermediate results and always label them with a version number.

2 Project Controlling

2.1 Project Status Report

Evergreen Roots 01	PROJECT STATUS REPORT as of 13.11.2023
 Project crisis Project in difficulties Project according to plan ✓	**1) Overall Status:** The Evergreen Roots (Eroots) project is currently in a positive state, with all key objectives and activities progressing as planned. The project team is actively working on client and admin interfaces, and significant milestones, including the completion of Prototype 1, are anticipated to be achieved within the specified timelines. The project's schedule and resources/costs are well-managed, and the overall project context and organizational culture are positive. Regular communication and progress updates contribute to a collaborative and successful project environment.
2) Status Project objectives 1) Client interface: Positive 2) User interface: Positive 3) Admin interface: Positive	**Activities:** 1) Product page with all products created: On track 1) One product page created: On track 2) Products ordered: On track 2) View previous orders with a history view: On track 2) Implement a search functionality: On track 3) Products with data persistence created, edited, and deleted: On track 3) Categories with data persistence created, edited, and deleted: On track 3) Appointments booked (reservations): On track 3) Products, categories, and admin and user login data in the database stored: On track
3) Status Project progress * Overall Progress: On track	**Activities:** * Project proposal signed by the project sponsor (project start): Completed on 07.07.2023 * Product specifications finished: On track * Technical specifications finished: On track * Prototype 1 completed: On track * Project completion protocol signed by the project sponsor: On track, planned for 29.02.2024
4) Status Schedule * Overall Schedule: On track	**Activities:** * Project proposal signed by the project sponsor (project start): Revised to 07.07.2023 * Product specifications finished: On track, planned for 06.11.2023 * Technical specifications finished: On track, planned for 20.12.2023 * Prototype 1 completed: On track, planned for 08.01.2024 * Project completion protocol signed by the project sponsor: On track, planned for 29.02.2024
5) Status Resources/costs * Overall Resources/Costs: None	**Activities:** * No additional resources or costs reported
6) Status Context * Overall Context: Positive	**Activities:** * Project context is well-managed, and communication is effective
7) Status Organisation/culture * Overall Organisation/Culture: Positive	**Activities:** * Weekly progress updates are consistent, and the team is engaged.

3 Project Close Down

To do….

3.1 Project Close Down Report

<Project name> <Project no.>	PROJECT CLOSE DOWN REPORT
1) Overall impression	**2) Reflection: Fulfilment of objectives**
3) Reflection: Deliverables / Schedule	
4) Reflection: Resources / Costs	
5) Reflection: Internal Organisation / Environmental Relationships	
6) Performance appraisal (Project owner, Project manager, Project member)	**7) Lessons learned (Summary of Experiences and suggestions for improvement)**

8) Post-Project Phase Planning, Additional Tasks

To-Do	Owner	Schedule

9) Project Close Down

---------- *<Name>* (Project owner)

---------- *<Name>* (Project manager)

3.2 Minutes – Project Close Down

3.2.1 Project Close Down Workshop

Requirement Specification for Project EVERGREEN ROOTS

Document name	ERoots-RequirementSpecification-v1.docx
Version	1.0
Date	6-Nov-23
State	Finished
Confidentiality	For internal use only
Document No.	ERoots-RequirementSpecification-v1
Distribution	Njeri Everlyn, Paukner Sarah, Braun Valerian, Andrei Elias

Table of contents

Table of contents 2
Summary 2
0. Revision history 3
1. Objectives and usage environment 3
1.1. Description of the actual situation and desired state 3
1.2. Actual situation 3
1.3. Initial situation 3
1.4. Initiative for the project 3
1.5. Desired state 4
1.6. Description of objectives 4
1.6.1. Need-to-Have Objectives: 4
1.6.2. Optional Objectives (Desirable but Not Critical): 4
1.6.3. Not-to-Have Objectives (Not Important for This Project): 5
2. Result 5
3. Benefits 5
4. Use of the product 5
4.1.1. Application areas 5
4.1.2. Target user groups 5
5. Product Overview 6
5.1. Feature list 6
5.2. Use case description for the whole system 8
6. Product Description 9
6.1. Product features 9
6.1.1. Use Case 01 xxxxx 9
6.1.2. Use Case 02 xxxxx 9
6.1.3. Other features 10
7. Non-functional requirements 11
8. Product data 12
9. Layout of system architecture 13
9.1.1. Frontend 13
9.1.2. Backend 14
9.1.3. Hosting 14
10. Interfaces 15
10.1. Homepage 15
10.2. Cart 15
11. Delivery 16
12. Glossary 17

Summary

Evergreen Roots is an eCommerce web application project created to empower a Kenyan Women Self Help Group. The project's main framework is centered around providing these women with the means to upload and sell their agricultural products. It features an admin dashboard for efficient system management, product management to handle product information, and order management for seamless order processing.

One of the project's primary objectives is to prioritize security, implementing encryption, user authentication, and authorization controls to protect sensitive data. For the end-users, a client dashboard is designed for profile management, order history access, and detailed product information, ensuring a smooth and intuitive experience.

The project's database is well-structured, supporting the system's functionality, while a comprehensive testing strategy is in place to maintain a high-quality and reliable product. Evergreen Roots aims to transform the lives of Kenyan women by leveraging technology and eCommerce to foster sustainable economic growth and empower self-sufficiency.

0. Revision history

Version Nr.	Author	Date	Changes, reason for change
0.1	Paukner, Njeri	25.10.2023	Creation, initial version
0.2	Njeri	27.10.223	1.1, 1.2, 1.3, 1.4, 1.5, 1.2.1, 2, 3, 4, 4.1.1, 4.1.2
0.3	Paukner, Njeri	02.11.2023	6. Layout of system architecture
0.4	Njeri, Paukner, Braun	06.11.2023	2.1, 1, Summary, 1.6, 1.6.1, 1.6.2, 1.6.3, 4, 5, 7, 8, 9, 10, 11, 12, 5.2, 6

1. Objectives and usage environment

This part of the document describes the project's objectives and usage environment.

1.1. Description of the actual situation and desired state

Evergreen Roots is a project that offers an eCommerce web-application for a Kenyan Women Self Help Group, which allows them to upload their farmed goods. The Web-Application provides an admin dashboard to manage the products. The front interface offers a wide-range product page, and ordering page, a history view, and a search functionality. The web-application offers the creation of User Accounts, with which they can book appointments to be able to inspect the goods before they are purchased.

1.2. Actual situation

The project sponsor, a Kenyan Self-help Group, is currently operating within the confines of person-to-person sales, which has constrained their capacity to tap into a broader customer demographic. This limitation has, unfortunately, resulted in the emergence of a significant food waste challenge. The restricted reach of their current sales approach has led to excess or unsold food products, contributing to the larger issue of food waste in the region. Addressing this dual concern, expanding their customer outreach, and mitigating food waste, is a pivotal aspect of this project's mission.

1.3. Initial situation

At present, the Women Self Help Group is facing challenges related to the limited sale of their products. Consequently, a substantial portion of their produce goes to waste, leading to adverse environmental impacts and reduced income potential.

1.4. Initiative for the project

In Kiambu, Kenya a Women Self Help Group wants to improve their sales of locally farmed goods in order to expand their consumer base. To achieve this, they need a web-based platform that enables online ordering of their products, with payment to be made upon pick-up.

1.5. Desired state

The e-commerce web application boasts an intuitive admin dashboard for seamless product management, along with a customer dashboard that empowers clients to order products through appointment scheduling. The focused customer acquisition strategy is geared towards broadening the sponsor's customer base, with the overarching goal of making a substantial impact in the battle against food waste.

1.6. Description of objectives

This part of the document describes the project's objectives.

1.6.1. Need-to-Have Objectives:

- **Admin Dashboard**: Develop an admin dashboard with specific features, such as categories management, product management, and order management. Ensure that it allows administrators to efficiently manage the system.
- **Product Management**: Create a product management system that enables the addition, modification, and categorization of products. Ensure it supports essential product attributes like name, description, pricing, and stock availability.
- **Ordering Management**: Implement a robust order management system, enabling order creation, modification, and monitoring.
- **Security**: Prioritize the implementation of comprehensive security measures, including encryption, user authentication, and authorization controls. Address potential vulnerabilities and protect sensitive data.
- **Client Dashboard**: Develop a client dashboard that allows users to view order history, and access product details, ensuring a seamless and intuitive user experience.
- **Database**: Design and implement a well-structured database, including schema, relationships, and data storage requirements, to support the functionality of the system.
- **Testing**: Establish a comprehensive testing strategy, to identify and resolve issues, ensuring a high-quality and reliable product.

1.6.2. Optional Objectives (Desirable but Not Critical):

Implement a **user review** feature for products, enabling customers to share their experiences, with the goal of having it fully functional within three months to evaluate user engagement.

Integrate a **coupon** and **discount code system** at checkout to enhance the shopping experience and assess its impact on sales, ensuring it's operational before the project's final release.

Develop a **wish list feature** that allows users to save products for future reference, aiming to encourage return visits and have it fully operational within two months of the project's launch.

Create a **user registration** option for a newsletter subscription to build a marketing channel for promotional updates and engagement, making it available from the project's launch and actively promoting it.

1.6.3. Not-to-Have Objectives (Not Important for This Project):

Online payment system integration is not a project objective for this phase, as the primary focus is on core functionalities, including admin and client dashboards, product management, ordering management, security, and database implementation.

2. Result

An eCommerce web-application with an admin dashboard to upload new products and a client dashboard to order products and make an appointment for the evaluation of the farmed good getting sold.

3. Benefits

The project is economically beneficial for the reduction of food waste in the local part of Kenia where the project sponsor lives. The web application sets out to reach a higher audience of buyers to reduce the spoilage of unbought goods.

4. Use of the product

This part of the document defines the application area and target user group of the project.

4.1.1. Application areas

The product is set to be an available web application through the internet via the domain name “eroots.store”.

4.1.2. Target user groups

Target users of the product are the Kenian population around the project sponsor that buy the goods that have been planted locally.

5. Product Overview

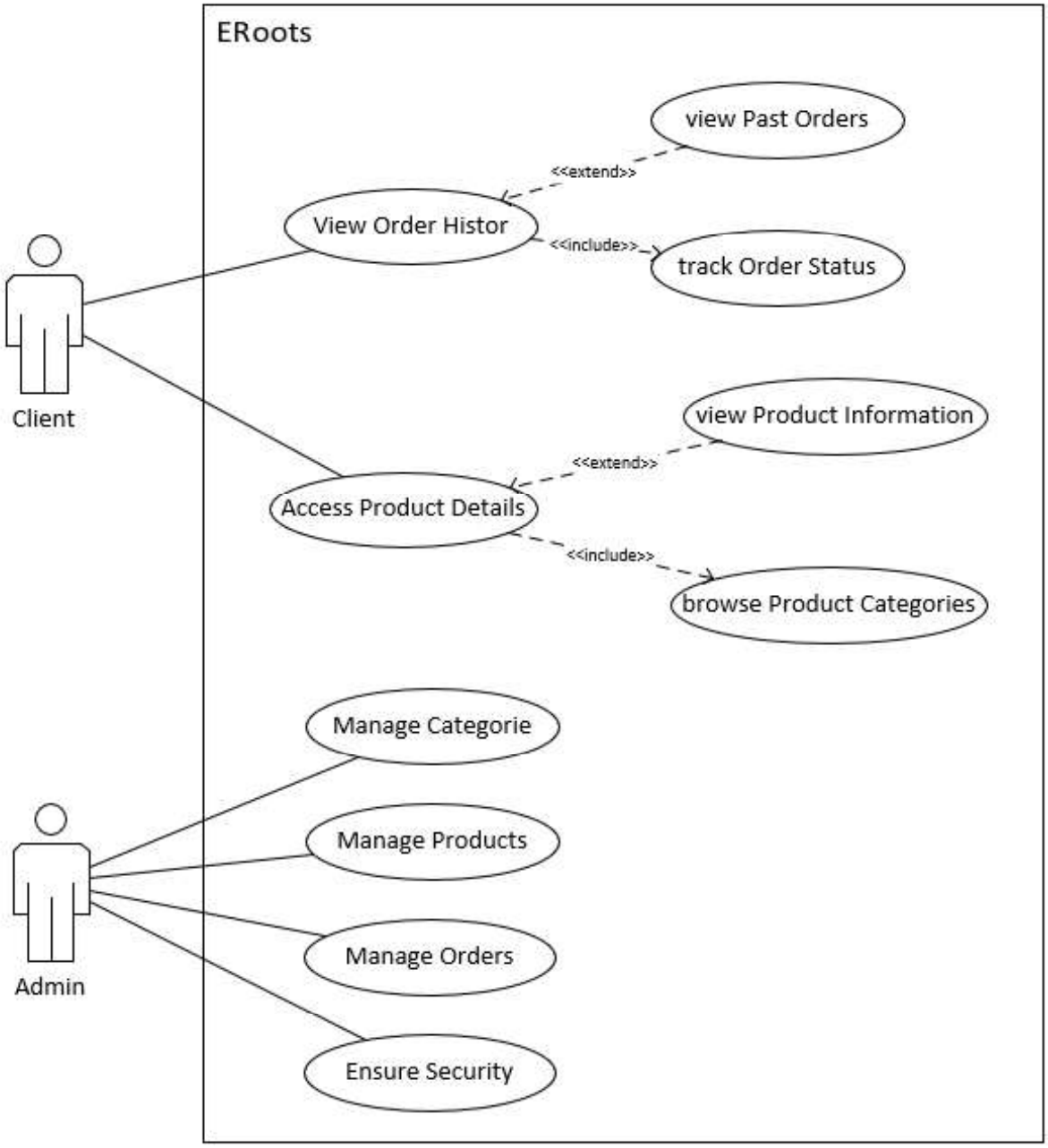
Component	Description
Admin Dashboard	Interface for administrators to manage categories, products, and orders.
Client Dashboard	User-friendly interface for clients to view order history and access product details.
Product Management	System module for adding, modifying, and categorizing products.
Ordering Management	System module for creating, modifying, and monitoring orders.
Security Measures	Comprehensive measures including encryption, user authentication, and authorization controls.
Database	Well-structured database with schema, relationships, and data storage to support system functionality.
Testing Strategy	Comprehensive testing plan to ensure product quality and reliability.
User Review Feature	Optional feature for users to share product reviews.
Coupon System	Optional feature for coupon and discount code integration.
Wish List Feature	Optional feature for saving products for future reference.
Newsletter Subscription	Optional user registration for newsletter subscription.

5.1. Feature list

Category	Feature	Description
Need-to-Have	Admin Dashboard	Develop an admin dashboard for efficient system management, including user, product, and order management.
	Product Management	Create a system for adding, modifying, and categorizing products, with support for key attributes like name, description, pricing, and stock availability.
	Ordering Management	Implement a robust order management system, allowing order creation, modification, and monitoring.
	Security	Prioritize comprehensive security measures, including encryption, user authentication, and authorization controls, to protect sensitive data and address vulnerabilities.
	Client Dashboard	Develop a user-friendly client dashboard for profile management, order history viewing, and product details access.
	Database	Design and implement a well-structured database with schema, relationships, and data storage to support system functionality.
	Testing	Establish a comprehensive testing strategy to ensure a high-quality and reliable product.

Optional	User Review Feature	Implement a user review feature for products, enabling customers to share their experiences, with a three-month evaluation period for user engagement.
	Coupon and Discount Code System	Integrate a coupon and discount code system at checkout to enhance the shopping experience and assess its impact on sales, ensuring functionality before the final release.
	Wish List Feature	Develop a wish list feature allowing users to save products for future reference, with the aim of encouraging return visits and full functionality within two months of the project launch.
	Newsletter Subscription	Create a user registration option for a newsletter subscription to build a marketing channel for promotional updates and engagement, actively promoting it from the project's launch.
Not-to-Have	Online Payment System Integration	Online payment system integration is not a project objective for this phase, as the primary focus is on core functionalities such as admin and client dashboards, product management, ordering management, security, and database implementation.

5.2. Use case description for the whole system



6. Product Description

6.1. Product features

6.1.1. Use Case 01 xxxxx

Use Case 01	
Name	View Past Orders
Short description	Extend "View Past Orders" to include additional functionality.
Actors	Client
Trigger	Client's desire to view and interact with past orders.
Result(s)	Client can view past orders.
Preconditions	Client has accessed the order history.
Incoming data	Order details and actions to be taken (e.g., reordering a past order).
Postconditions	Any additional actions taken by the client are reflected in the system.
Process description (essential action steps)	● Client selects a past order from the order history. ● The system displays the details of the selected order. ● Client has the option to perform additional actions, such as reordering the same products. ● The system validates the actions and processes them accordingly.
Error behavior	Client attempts to reorder an out-of-stock product
Variations	/

6.1.2. Use Case 02

Use Case 2	
Name	Access Product Details
Short description	View detailed information about a product.
Actors	Client
Trigger	Client's desire to explore a product's details.
Result(s)	Client can access and review product information.
Preconditions	Client selects a product to explore.
Incoming data	Product selection.
Postconditions	Client has reviewed the product details.
Process description (essential action steps)	● Client selects a product from the catalog. ● The system displays detailed information about the selected product, including name, description and pricing. ● Client can interact with the product details and perform actions like adding it to the cart or leaving a review.
Error behavior	Client attempts to access product details for a non-existent product
Variations	/

6.1.3. Use Case 03

Use Case 3	
Name	Manage Categories
Short description	Manage the product categories in the system.
Actors	Admin
Trigger	Admin's desire to add, modify, or delete product categories.
Result(s)	The product categories are effectively managed in the system.
Preconditions	Admin is logged in and has access to the admin dashboard.
Incoming data	Category name, description, and optional parent category information.
Postconditions	The product categories are added, modified, or deleted in the database.
Process description (essential action steps)	● Admin selects the "Manage Product Categories" option. ● The system displays a list of existing product categories. ● Admin can add a new category or modify/delete existing categories. ● The system validates the input and updates the database accordingly.
Error behavior	Administrator attempts to delete a category with associated products.
Variations	Administrator assigns parent categories to subcategories

6.1.4. Use Case 04

Use Case 4	
Name	Manage Products
Short description	Admin manages product listings and attributes.
Actors	Admin
Trigger	Admin's desire to add, modify, or remove products from the system.
Result(s)	The product catalog is effectively managed and updated.
Preconditions	Admin is logged in and has access to the admin dashboard.
Incoming data	Product details, including name, description, pricing, and stock availability.
Postconditions	The product catalog is updated and reflected in the database.
Process description (essential action steps)	● Admin selects the "Manage Products" option. ● The system displays a list of existing products. ● Admin can add new products, modify existing product details, or remove products. ● The system validates the input and updates the product catalog accordingly.
Error behavior	Admin attempts to delete a product that is associated with active orders
Variations	/

7. Non-functional requirements

This part of the document describes the project's relevant non-functional requirements for the project:

1. **Usability**: The system should be user-friendly, ensuring accessibility for handicapped users, adherence to corporate identity rules and guidelines, and a responsive and intuitive graphical user interface.

2. **Reliability**: The system should be highly reliable and available, with 24x7 access for both the Self-Help Group and its clients. Correctness of results is paramount to maintain trust in the system.

3. **Performance**: The system must meet specific performance requirements, including response time for user interactions. For instance, it may specify that all response times for user actions should be less than a certain number of seconds to ensure a smooth user experience.

4. **Security**: Robust data protection and backup measures should be in place to safeguard data integrity. Protection against various forms of cyberattacks, including cryptographic requirements, should be enforced to protect sensitive information.

5. **Portability**: The system should be independent of specific requirements, such as particular computer configurations, operating systems, protocols, middleware, and tools. This ensures that the system can be deployed across various environments.

6. **Maintainability**: Good maintainability is crucial, and it involves having a well-structured architecture and comprehensive documentation. This facilitates the incorporation of both functional and technical changes with ease, ensuring the system's adaptability and longevity.

7. **Scalability**: The system should be designed to handle an increase in the number of users over time. Scalability requirements may specify how the system should scale and distribute resources to accommodate more users.

8. **Localization**: The system should support localization by allowing for different languages, currencies, and compliance with local legal frameworks to cater to a broader user base.

9. **Architecture**: The choice of system architecture should be defined, whether it's a server/client model, Service-Oriented Architecture (SOA), or any other architecture that best suits the project's needs.

10. **Product Performance and Functional Constraints**: Detailed performance requirements for various product features and product information should be established. These constraints could include specific time limits, response times, screen resolution requirements, usability guidelines, user management, reliability and security standards, and compatibility with industry standards and interfaces.

These non-functional requirements play a critical role in shaping the design, development, and deployment of the Evergreen Roots system, ensuring it meets the desired performance, security, and usability standards.

8. Product data

In Evergreen Roots, important data that is stored permanently, along with approximate quantities and data storage requirements, is as follows:

1. **Customer Data (LD10)**: Approximately 50,000 customer records should be stored. This data should include customer profiles, contact information, order history, and any relevant preferences. The storage should prioritize reliability to ensure that customer information is always available. Security measures should be in place to protect customer data, including encryption and access controls. Speed of access is crucial for providing a seamless user experience.
2. **Product Data (LD20)**: Up to 10,000 product records need to be stored, including details like product names, descriptions, pricing, stock availability, and categorization. Reliability is essential to maintain the accuracy of product information, and access speed is important for quick retrieval and display on the website. Security measures should be in place to prevent unauthorized modifications to product data.
3. **User Directory (LD30)**: The user directory can contain a maximum of 300 user profiles, which include information about system administrators, vendors, and clients. This data should be securely stored with user authentication and authorization controls to protect sensitive information. Reliability and fast access are also important, especially for administrators who need to manage the system.

Possible Further Data:

1. **Order Data**: This includes information about customer orders, such as order details, order status, payment information, and delivery tracking. The quantity of order data would depend on the number of orders processed, and it should be stored securely and reliably to ensure accurate order processing and tracking.
2. **Session Data**: Session data for logged-in users to maintain user states and preferences while using the web application. This data should be managed for optimal performance and quick access.
3. **Transaction Logs**: Log data related to system activities and user interactions. These logs are essential for system monitoring, troubleshooting, and security purposes.
4. **Backup Data**: Regular backups of customer, product, and order data should be stored securely and with redundancy to ensure data recovery in case of any data loss.
5. **Product Images**: Images of products for display on the website. Image data should be optimized for efficient storage and fast loading on the user interface.

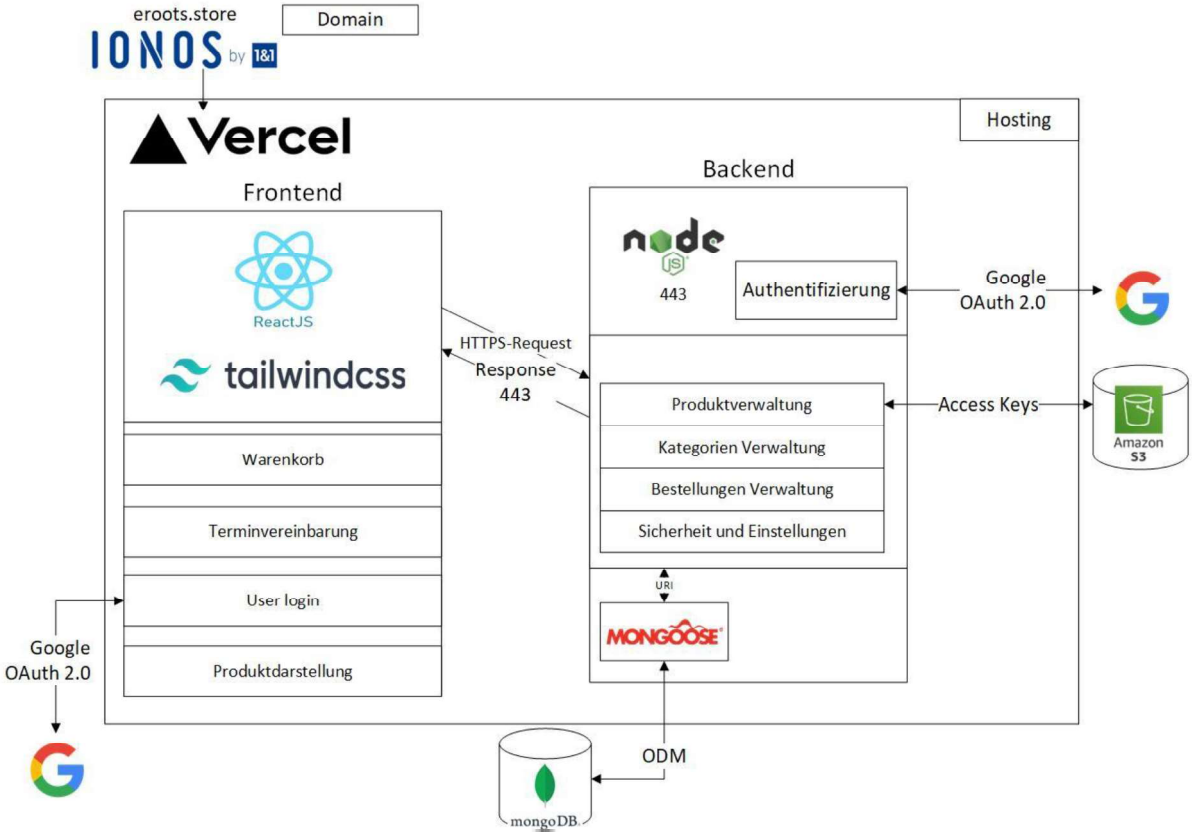
Requirements for Data Storage:

- **Reliability**: Data storage systems should have high reliability to minimize data loss and ensure data integrity. Regular backups and redundancy should be in place.
- **Security**: All stored data, especially customer information, should be well-protected with encryption, access controls, and security best practices to prevent unauthorized access and data breaches.
- **Speed of Access**: The data storage system should provide fast access to support real-time operations and deliver a seamless user experience.
- **Scalability**: The data storage solution should be designed to scale as the quantity of data grows, ensuring that it can accommodate future increases in customer, product, and order data.

- **Data Backup**: Regular data backup procedures should be established to prevent data loss in case of system failures or data corruption.
- **Compliance**: Ensure that data storage and handling comply with any relevant legal and regulatory requirements, especially regarding customer data and privacy.

The data storage requirements are crucial to maintaining the functionality, security, and performance of the Evergreen Roots system, ensuring that it can efficiently manage the permanent storage of important data. The specific choice of a database system should align with these requirements to achieve the desired outcomes.

9. Layout of system architecture



9.1.1. Frontend

The application's user interface is the face of the operation, responsible for creating a user-friendly and visually appealing experience. To make this happen, React.js was picked as framework.

When it comes to styling and aesthetics, the application uses Tailwind CSS. Tailwind is a CSS framework that offers a ton of pre-designed classes, making it super easy to style HTML elements. This simplifies the styling process and contributes to an overall polished user experience.

9.1.2. Backend

Behind the scenes, the backend, built on the Node.js platform, takes care of handling data, logic, and communication with the database. Node.js provides a speedy runtime environment for server-side tasks, making it an excellent choice for building scalable and high-performance backend systems.

For managing the database, Mongoose, an Object Data Modeling (ODM) library tailored for MongoDB is used. Mongoose simplifies the interactions with the MongoDB database by introducing a schema-based modelling system, validation features, and a robust query-building interface.

The application's database of choice is MongoDB, a NoSQL database known for its flexibility and scalability. MongoDB stores data in a JSON-like format, making it perfect for accommodating dynamic data structures.

Connecting via an access key is one of the methods used to access the AWS S3 bucket, where the storage of images is securely provided for the product inventory and management. This access key serves as a means of authentication and authorization, allowing the application to interact with the cloud-based storage service and perform various tasks, such as uploading, retrieving, and managing product images.

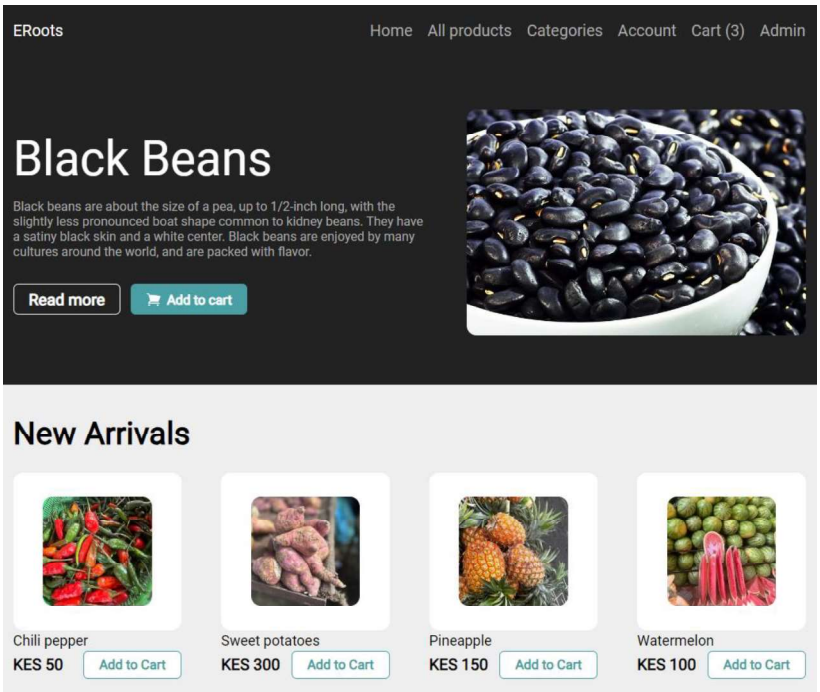
The admin login uses Google OAuth 2.0 to make the security safer and simpler. It lets administrators log in securely with their Google accounts, so there is no need to store passwords. Two-factor authentication can be used for extra security, and it works on different devices. This makes admin access easier and more reliable.

9.1.3. Hosting

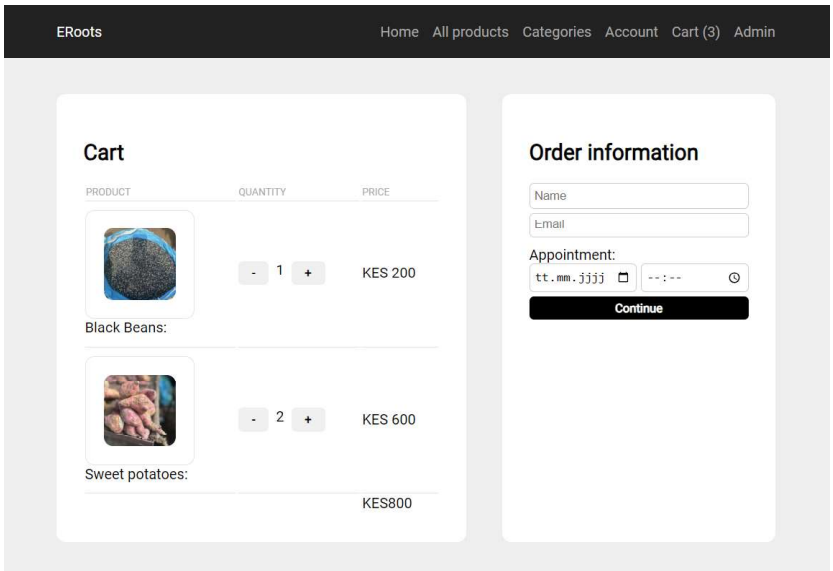
The web application gets hosted with Vercel, a platform known for fast and reliable web deployment. The domain, "eroots.store", is managed by IONOS, a domain registrar. When someone enters the domain in the browser, IONOS directs them to the web application hosted on Vercel. This setup ensures that the site is easily accessible and well-maintained.

10. Interfaces

10.1.Homepage



10.2.Cart



11. Delivery

The Evergreen Roots product will be delivered as a web-based application. It will not be distributed on physical media like CDs or USB sticks. Instead, users will access the application via the internet, making it accessible from any device with a web browser.

The product consists of the following components:

1. **Web Application**: The core application accessible through a web browser, including features like the admin dashboard, client dashboard, product management, and order management.
2. **Support and Training**:

— **User Manual**: A comprehensive user manual will be available as a downloadable resource on the website. It will cover all aspects of the application, from basic navigation to advanced features, troubleshooting, and best practices.

3. **Installation and Setup**:

Since Evergreen Roots is a web-based application, users will not need to install any software on their local devices. Instead, they will access the application by visiting the designated website.

4. **Updates and Maintenance**:

Software updates and maintenance will be handled by the Evergreen Roots development team. Users will not need to perform manual updates or maintenance tasks. Regular updates will be implemented to enhance security and add new features.

5. **Hosting**:

The web application will be hosted on a secure and reliable web server, ensuring 24/7 availability and optimal performance.

6. **Accessibility**:

The web application will be designed with accessibility in mind to accommodate users with disabilities. It will adhere to accessibility standards and guidelines to ensure that all users can interact with the platform.

In summary, Evergreen Roots will be delivered as a web-based application accessible through a web browser. The delivery will include customer support and a comprehensive user manual to assist users in making the most of the platform. Users will not need to install software locally, and updates and maintenance will be managed by the development team. Hosting will be secure and reliable, ensuring accessibility and availability around the clock.

12. Glossary

TERM	EXPLANATION
AWS S3 Bucket	A secure storage service for product images accessed via an access key for the web application.
Accessibility	Designing the app to be user-friendly for individuals with disabilities.
Google OAuth 2.0	Secure admin login with Google accounts, supporting two-factor authentication.
Installation and Setup	No software installation needed; access via the website.
IONOS	Domain registrar managing "eroots.store" for web app accessibility.
Mongoose	Streamlines interactions with the MongoDB database, enhancing efficiency.
MongoDB	Flexible and scalable NoSQL database storing data in a JSON-like format.
Node.js	Backend runtime for handling data, logic, and database communication.
React.js	Frontend framework for creating interactive user interfaces.
Tailwind CSS	CSS framework for frontend styling and user-friendly design.
Updates and Maintenance	Handled by the development team to keep the app current.
User Interface	Responsible for user-friendliness and visual appeal in the app.
User Manual	A comprehensive resource guiding users on app usage and troubleshooting.
Vercel	Hosting platform known for fast and reliable web application deployment.
Web Application	Core component accessible through a web browser.
Web-Based Application	Delivered via the internet, accessible on any device without physical distribution.

System Specification
Evergreen Roots

Document name	ERoots-SystemSpecification-v1
Version	1.0
Date	20.11.2023
State	finished
Confidentiality	RESTRICTED
Document No.	ERoots-SystemSpecification-v1
Distribution	Andrei, Braun, Njeri, Paukner, Schwarz, Bierbamer, Hell, Pirker

HTL Spengergasse, Team ERoots

Table of Contents

0 Revision History 4

4916 4

1 Introduction 6
- 1.1 Objective 6
- 1.2 Scope 6
- 1.3 Definitions, Acronyms and Abbreviations 7
- 1.4 References 9
- 1.5 Overview 9

2 Overall Description 10
- 2.1 Product perspective 10
- 2.2 System interfaces 11
  - 2.2.1 User interfaces 11
  - 2.2.2 Software interfaces 12
  - 2.2.3 Communications interfaces 13
- 2.3 Memory constraints 14
- 2.4 Operations 15
- 2.5 Site adaptation requirements 16
- 2.6 User characteristics 17
- 2.7 Product Functions Summary 18
  - 2.7.1 End User Functions: 18
  - 2.7.2 Admin Functions: 18
  - 2.7.3 Common Functions: 18
  - 2.7.4 Use Case Diagram: 19
  - 2.7.5 Activity diagram: 20
- 2.8 Documentation requirements 24
- 2.9 Constraints 24
- 2.10 Deliverables 25
- 2.11 Assumptions and Dependencies 25
- 2.12 Apportionment of Requirements 26

3 Specific requirements 27
- 3.1 External interfaces 27
  - 3.1.1 Display Products (/01/) 27
  - 3.1.2 Product Details (/02/) 27
  - 3.1.3 Shopping Cart Functionality (/03/) 28
  - 3.1.4 Place Orders (/04/) 28
  - 3.1.5 Product Search (/05/) 28
- 3.2 Admin Interface 29
  - 3.2.1 Create New Products (/06/) 29
  - 3.2.2 Edit Product Details (/07/) 29
  - 3.2.3 Delete Products (/08/) 29
  - 3.2.4 Create New Categories (/09/) 30
  - 3.2.5 Edit Category Details (/10/) 30
  - 3.2.6 Delete Category (/11/) 30
- 3.3 Detailed functionality description 31
  - 3.3.1 Product Page 31
  - 3.3.2 User Account 31
  - 3.3.3 Order Products 31
  - 3.3.4 Order History 31
  - 3.3.5 Search Functionality 31
  - 3.3.6 Product Management 31

3.3.7 Category Management 32
3.4 Classes and modules 33
3.5 Database Requirements 34
3.5.1 Data model and files 34
3.5.2 Table definition 36
3.4.3 Database Model 37
3.6 Performance requirements 39
3.6.1 Static Demands 39
3.6.2 Dynamic Demands 39
3.7 Design constraints 39
3.8 Quality characteristics, Software system attributes 40
3.8.1 Performance 40
3.8.2 Reliability 40
3.8.3 Security 40
3.8.4 Usability 40
3.8.5 Maintainability 40
4 Test cases 41
4.1 4.1 E2E Testing 41
4.2 Component Testing 41
4.3 Automated Testing Workflow 41
4.4 4.4 Test Reporting 41
5 Appendix 42
5.1 Glossary 42

0 Revision History

Version Nr.	Author	Date	Changes, reason for change
0.1	Paukner	14.05.2023	Creation, initial version
0.2	Paukner, Njeri	16.11.2023	Work log, 1.1
0.3	Paukner	18.11.2023	1.2, 2.1, 2.2, 2.2.1, 2.2.2, 2.2.3, 2.3, 2.4, 2.5
0.4	Njeri	19.11.2023	3.1.1,3.1.2
0.5	Andrei, Paukner, Njeri, Braun	20.11.2023	4, 3.5, 3.7, 2.6, 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, 1.3, 3.8, 3.4.1, 3.4.2, 3.1.3-3.1.11, 3.2, 3.3, 5, 5.1
1.0	Andrei, Paukner, Njeri, Braun	20.11.2023	First version completed

Work log (obligatory)

Name (Surname)	**Chapters**	**Pages**	**Words**	**Pictures** (nr / size in pages)	**Time spent** (h)
ANDREI	4, 3.6, 3.1.3 – 3.1.11	26-27, 38, 40	698	fig. 26 predicted hourly visits	2.7 h
BRAUN	2.7.5, 3.1, 3.2	19-22, 26-29	321	fig. 6 admin management for products fig. 7 admin management for orders fig. 8 admin management for categories fig. 9 user view order history	3.4 h
NJERI	3.1.1, 3.1.2 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.5, 5.1	23, 30-31, 42-43	572	fig. 12 Table Products fig. 13 Table Sessions fig. 14 Table Accounts fig. 15 Table Users fig. 16 Table Orders fig. 17 Database Model fig. 18 Database Modell fig. 19 Database fig. 20 Session Modell fig. 21 Product Modells fig. 22 Orders Modells fig. 23 Catagories Model fig. 24 Accounts fig. 25 User Modell	6.7 h
PAUKNER	1.1, 1.2, 1.3, 1.4, 1.5 2.2 + 2.2.1 + 2.2.2 + 2.2.3, 2.3, 2.4, 2.5, 2.6, 2.7,	5-25, 27, 32, 38, 42-43	4916	fig. 1 System Architecture fig. 2 Product Page fig. 3 Cart and Appointment fig. 4 Login Page fig. 5 ERoots Use Case	10.4 h

	2.8, 2.9, 2.10, 2.11, 2.12, 3.4, 3.7, 5, 5.1			fig. 6 Class Diagram Category, Product and Order fig. 7 Class Diagram User and Admin	
TOTAL	0-5.1	5-43	7 929	26 pictures	23.2 h

1 Introduction

This part of the document introduces the system specification of the ERoots project.

1.1 Objective

The Evergreen Roots (ERoots) project is a purposeful eCommerce web application designed to help a Kenyan Women Self Help Group with selling their farmed goods. This document delves into the collective roles of the ERoots team, which contributes collaboratively to the project's success.

One of the tasks the ERoots team focuses on is both development and system administration, playing a pivotal role in defining the project's architecture, framework, and features. This offers insights into creating a platform for seamless agricultural product sales and provides guidance on efficient system management for smooth application operation.

Security and database management are two more requirements in the project. The system specification outlines the project's security measures, including encryption and user authentication, while providing insights into maintaining a well-structured database as a reference for optimization and maintenance.

Additionally, the system specification explores the testing within Evergreen Roots. It outlines a comprehensive testing strategy to ensure a high-quality product. By examining testing requirements, the document ensures the application meets performance standards, contributing to a seamless user experience.

This system specification provides a detailed analysis of the collaborative roles undertaken by the Evergreen Roots (ERoots) team. It serves as a resource for understanding the complexities of the development, system administration, project management, security, database administration, and testing within the project.

1.2 Scope

The software product at the heart of our project, named Evergreen Roots, an eCommerce Platform, is tailored to uplift the Kenyan Women Self Help Group. This platform serves as an online marketplace, simplifying the process for them to sell their agricultural products seamlessly. The project revolves around the development and implementation of this software.

Evergreen Roots has several key features to ensure an effective and user-friendly experience. For users, the platform offers an intuitive interface, allowing easy upload and sale of agricultural products. The admin dashboard enhances system management efficiency, while the product management capabilities and an order management system contribute to smooth operations.

End-users benefit from a client dashboard, providing access to a detailed product information. The platform prioritizes security through encryption, user authentication, and authorization controls to protect sensitive data. The well-structured database supports overall system functionality, while a comprehensive testing strategy is in place to guarantee a high-quality and reliable product.

It's important to note what Evergreen Roots does not do: it does not engage in direct product sales but serves as a facilitator, allowing the Self-Help Group to showcase and sell their

agricultural products.

The special benefit of Evergreen Roots lies in its impact on the lives of the farmers selling their goods in Kenya. The platform fosters sustainable economic growth, empowers self-sufficiency, and provides a broader market reach for the products of the Women Self Help Group.

1.3 Definitions, Acronyms and Abbreviations

These explanations should provide a clear understanding of the technical terms and abbreviations used in the specification.

Abbreviation	Name	Explanation
API	Application Programming Interface	A set of protocols and tools for building software applications, allowing them to communicate with each other.
CDN	Content Delivery Network	A distributed network of servers that work together to deliver web content, like images and videos, to users based on their geographical location.
CSS	Cascading Style Sheets	A style sheet language used for describing the presentation of a document written in HTML or XML, including colours, fonts, and layout.
HTML	Hypertext Markup Language	The standard markup language for creating web pages and web applications. It describes the structure of web pages using markup tags.
OAuth 2.0	Open Authorization 2.0	An authorization framework that enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner or by allowing the third-party application to obtain access on its own behalf.
PDF	Portable Document Format	A file format used to present and exchange documents reliably, independent of software, hardware, or operating systems.
	React.js	A JavaScript library for building user interfaces or UI components, often used for creating single-page applications.
SQL	Structured Query Language	A domain-specific language used in programming and designed for managing and manipulating relational database management systems (RDBMS).
	TailwindCSS	A utility-first CSS framework for rapidly building custom designs. It provides low-level utility classes to create designs directly in the markup.
UI	User Interface	The space where interactions between humans and machines occur. It includes everything designed into a device with which a user can interact.

URL	Uniform Resource Locator	The address used to access resources on the internet. It typically consists of a protocol (such as HTTP or HTTPS), domain name, and path.
UX	User Experience	The overall experience a user has with a product, including its usability, accessibility, and the emotional response it generates.

1.4 References

1. **MongoDB Documentation**
Source: MongoDB
Accessed on: 18.11.2023

2. **React.js Documentation**
Source: React.js
Accessed on: 18.11.2023

3. **TailwindCSS Documentation**
Source: TailwindCSS
Accessed on: 18.11.2023

4. **Google OAuth 2.0 Information**
Source: Stack Overflow
Accessed on: 18.11.2023

1.5 Overview

0. **Revision History**: Lists the revisions made to the document, providing a historical overview of changes and updates.
1. **Introduction**:
   - 1.1 **Objective**: Defines the purpose and goals of the document.
   - 1.2 **Scope**: Describes the boundaries and extent of the document's coverage.
   - 1.3 **Definitions, Acronyms and Abbreviations**: Clarifies the meanings of specific terms, acronyms, and abbreviations used in the document.
   - 1.4 **References**: Lists external sources or documents referred to in the specification.
   - 1.5 **Overview**: Offers a high-level overview of the document's structure and content.
2. **Overall Description**:
   - 2.1 **Product perspective**: Provides an overall context of the Evergreen Roots web application within the larger system or environment.
   - 2.2 **System interfaces**: Describes the interfaces, including user interfaces, software interfaces, and communication interfaces.
   - 2.3 **Memory constraints**: Specifies limitations related to system memory.
   - 2.4 **Operations**: Outlines the main operations and functionalities of the system.
   - 2.5 **Site adaptation requirements**: Details requirements for adapting the system to different sites or environments.
   - 2.6 **User characteristics**: Describes the anticipated characteristics of the end-users.
   - 2.7 **Product Functions Summary**: Summarizes the primary functions and features of the Evergreen Roots application.
   - 2.8 **Documentation requirements**: States the documentation needed for the system.
   - 2.9 **Constraints**: Lists any constraints that may impact the development or operation of the system.
   - 2.10 **Deliverables**: Identifies the expected deliverables from the development process.
   - 2.11 **Assumptions and Dependencies**: Highlights the assumptions made during the project and any dependencies on external factors.
   - 2.12 **Apportionment of Requirements**: Discusses how requirements are allocated and distributed across the system.

3. **Specific requirements**:
    3.1 **External interfaces**: Details external interfaces, including user interfaces.
    3.2 **Detailed functionality description**: Provides a more in-depth description of specific functionalities, such as the product page and product search.
    3.3 **Classes and modules**: Describes the classes and modules that compose the system.
    3.4 **Database Requirements**: Outlines requirements related to the system's database, including data model, ER model, and table definitions.
    3.5 **Performance requirements**: Specifies both static and dynamic demands on system performance.
    3.6 **Design constraints**: Identifies constraints influencing the system design.
    3.7 **Quality characteristics, Software system attributes**: Defines quality characteristics and attributes expected from the software system.
4. **Test cases**:
    4.1 **E2E Testing**: Describes end-to-end testing requirements.
    4.2 **Component Testing**: Outlines testing requirements for individual components.
    4.3 **Automated Testing Workflow**: Details the workflow for automated testing.
    4.4 **Test Reporting**: Discusses the reporting process for test results.
5. **Appendix**:
    5.1 **Glossary**: Provides a glossary of terms used in the document.

2 Overall Description

This part of the document gives a general overview of the project system.

2.1 Product perspective

The Evergreen Roots web application operates as a standalone product, meaning it doesn't rely on or connect to a larger system. It functions independently, providing a dedicated space for the Kenyan Women Self Help Group to sell their agricultural products.

Evergreen Roots' system architecture consists of a user interface for clients and administrators, application logic handling product and order management, a security layer for data protection, and a well-structured database, to ensure a user-friendly and secure experience.

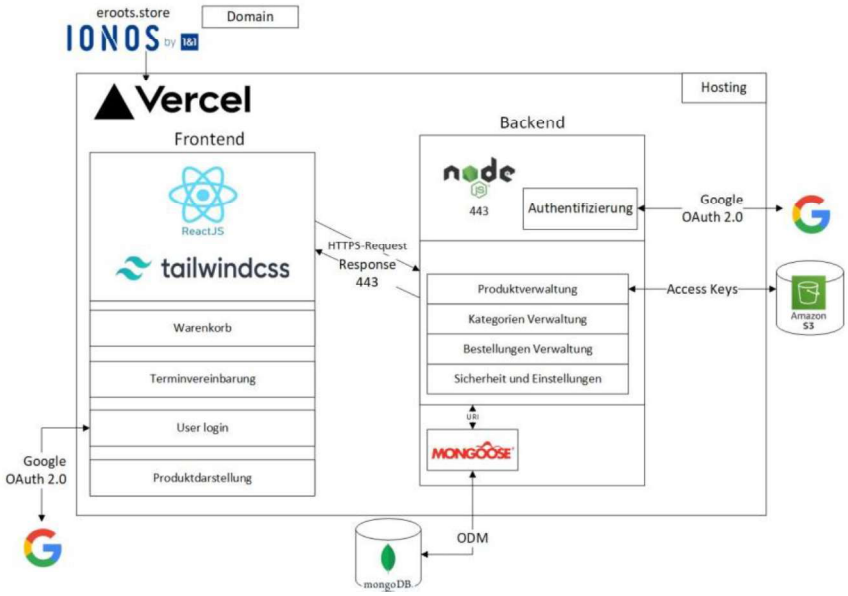


fig. 1 System Architecture

2.2 System interfaces

This part of the document focuses on the system's interfaces which are integral to forming the backbone of the system.

2.2.1 User interfaces

The user interface of the Evergreen Roots web application is designed to be user-friendly. It combines React.js with TailwindCSS styling and incorporates modern design principles to enhance the overall user experience.

Screen Layout:

1. **Cart**: The Cart section provides users with a clear view of their selected items and an easy checkout process.
2. **Appointment**: Users can schedule appointments as this section offers a straightforward layout to input and manage appointments efficiently.
3. **User Login (With Google OAuth2.0)**: The login works with Google OAuth2.0 for a secure and convenient login experience. Users can log in with their Google credentials, streamlining the authentication process.
4. **Product Page**: The product page features a visually appealing layout showcasing products, complete with detailed information to aid users in making informed purchase decisions.

User Qualifications:

ERoots is designed to be user-friendly to speak to a diverse audience. No specific qualifications are required for basic interactions, such as browsing products, adding items to the cart, and viewing appointments.

Illustrations:

1. **Product Page**

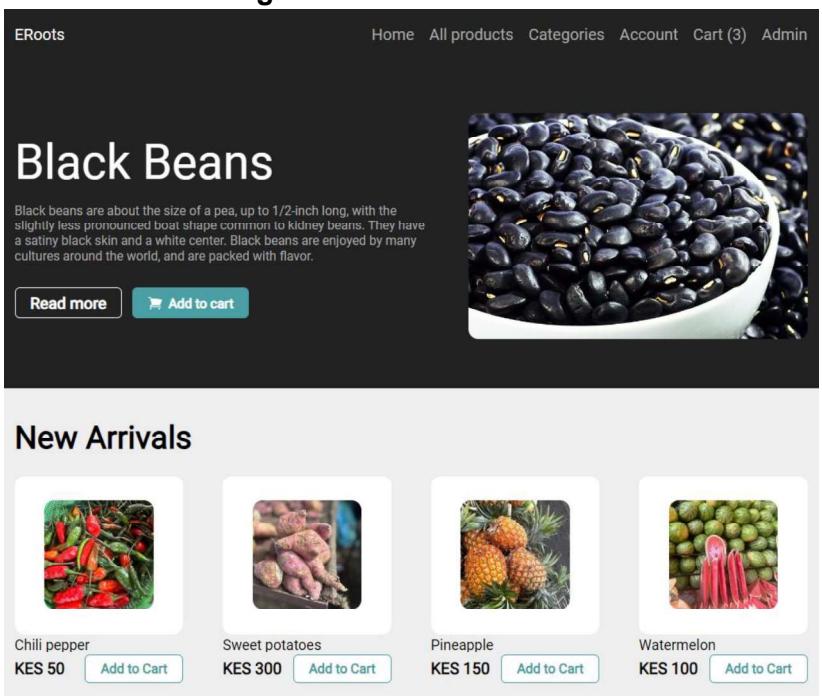


fig. 2 Product Page

2. **Cart + Appointment**

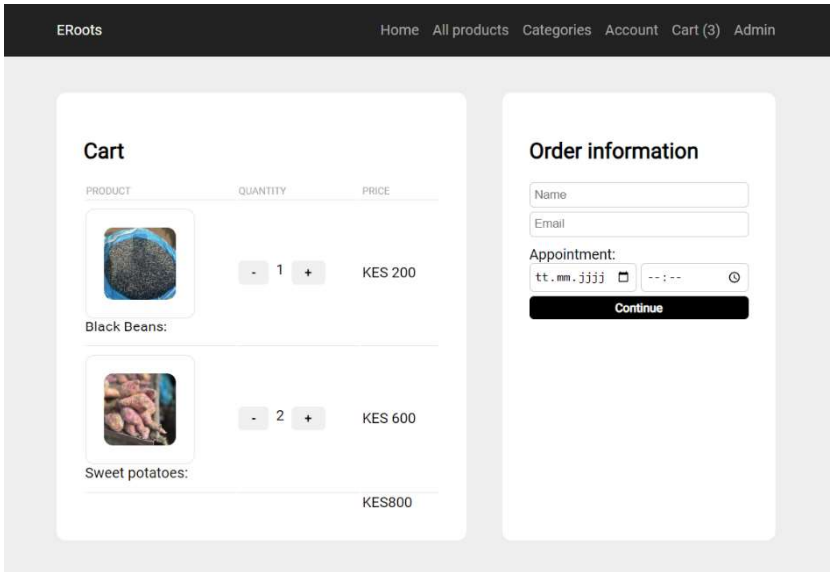


fig. 3 Cart and Appointment

3. **User Login**

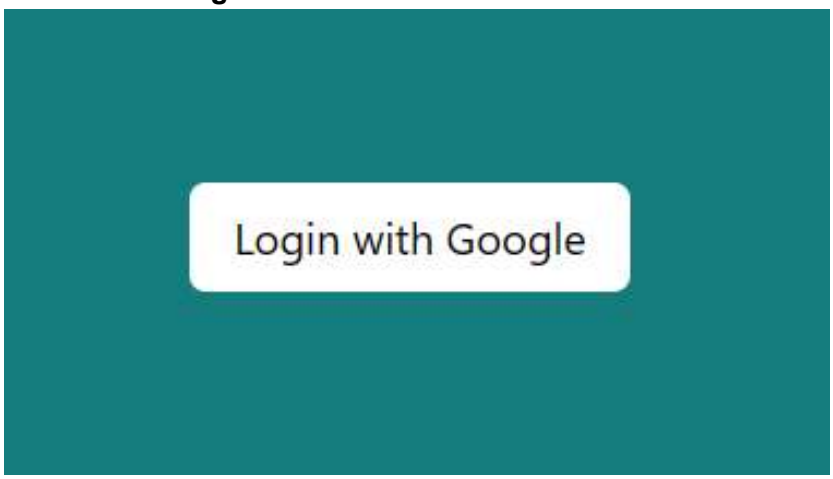


fig. 4 Login Page

2.2.2 Software interfaces

The Evergreen Roots web application utilizes various software components to ensure its functionality:

1. **Database**:
Name: MongoDB
License: Server-Side Public License (SSPL)
Version: 7.0
Manufacturer: MongoDB, Inc.
Objective of Integration: MongoDB serves as the database management system for Evergreen Roots, providing a scalable and flexible structure for storing and retrieving data related to products, orders, and user information.

2. **Development Frameworks**:

React.js:
License: MIT License
Version: 18.2.0
Manufacturer: Facebook
Objective of Integration: React.js is employed for building the user interface of the project, offering a component-based structure for efficient development and seamless UI updates.

TailwindCSS:
License: MIT License
Version: 3
Manufacturer: Tailwind Labs Inc.
Objective of Integration: TailwindCSS is utilized for styling the user interface, providing a utility-first approach for rapid and maintainable styling.

3. **User Authentication**:

Google OAuth 2.0:
License: Dependent on Google's API terms
Version: 3
Manufacturer: Google
Objective of Integration: Google OAuth 2.0 is integrated for secure user authentication, allowing users to log in with their Google credentials.

Data Exchange Interfaces:
Database Interface:
Data Format: JSON (JavaScript Object Notation)
Interface Definition: The database interface is defined within the application's server-side code, specifying how data is stored and retrieved from MongoDB collections.

User Authentication Interface:
Data Format: OAuth 2.0 Tokens
Interface Definition: The user authentication interface is defined through the integration of Google OAuth 2.0 API. The detailed interface specifications are provided in Google's API documentation.

2.2.3 Communications interfaces

The Evergreen Roots web application interacts with various interfaces and protocols to ensure the communication between its components:

1. **HTTP/HTTPS Protocol**:
   Purpose: The project uses the Hypertext Transfer Protocol Secure (HTTPS) to facilitate communication between the client-side (user's browser) and the server-side (web application).
   Format of Data Exchange: Data is exchanged using JSON (JavaScript Object Notation) over HTTP/HTTPS, allowing for lightweight and efficient communication.

2. **RESTful API (Application Programming Interface)**:
   Purpose: The RESTful API is employed for communication between the client-side (user interface) and the server-side (application logic and database).
   Format of Data Exchange: Data is typically exchanged in JSON format, conforming to RESTful principles for resource-based communication.
3. **MongoDB Driver**:
   Purpose: The MongoDB driver interface allows the server-side application to interact with the MongoDB database.
   Format of Data Exchange: Data is exchanged in BSON (Binary JSON), the binary-encoded representation of JSON used by MongoDB.
4. **Google OAuth 2.0 API**:
   Purpose: ERoots integrates with the Google OAuth 2.0 API for user authentication.
   Format of Data Exchange: Data is exchanged using the OAuth 2.0 protocol, which typically involves the exchange of tokens, including access and refresh tokens.
5. **Frontend-Backend Interface (React.js)**:
   Purpose: This interface facilitates communication between the frontend (user interface developed with React.js) and the backend (server-side application).
   Format of Data Exchange: Data is often exchanged in JSON format via HTTP requests, following RESTful principles.
6. **TailwindCSS Stylesheets**:
   Purpose: TailwindCSS stylesheets provide a standardized and utility-first approach to styling the user interface.
   Format of Data Exchange: Stylesheets are written in CSS, and the styling information is applied to the HTML structure of the user interface.

2.3 Memory constraints

Primary Storage Constraints:

1. **Memory Capacity**: The available RAM (Random Access Memory) capacity influences the system's ability to handle concurrent user requests and efficiently manage application data.
2. **Data Access Speed**: Constraints on data access speed may affect the overall performance of real-time operations, such as the responsiveness of the application, or dynamic content loading and user interactions.
3. **Volatile Nature**: Volatile memory loses its contents when the system is powered off, that means it is necessary to have a robust data persistence mechanism, in order to ensure that critical data is stored in non-volatile storage for recovery and continuity.

Secondary Storage Constraints:

1. **Disk Space**: The secondary storage imposes constraints on the amount of data that can be stored over the long term. The disk space limitations may impact the scalability of the platform and require effective data management strategies to handle growing datasets.

2. **Data Retrieval Speed**: The constraints on data retrieval speed may lead to delays in loading product information and other data-intensive operations.
3. **I/O Throughput**: The input/output (I/O) throughput of secondary storage devices, such as hard drives or solid-state drives, places constraints on the speed at which data can be read from and written to storage.
4. **Backup and Recovery Time**: Secondary storage constraints could impact the time required for regular backups and the recovery of data in case of system failures. Through efficient backup and recovery processes the minimizing of the downtime and ensuring data integrity are provided.

2.4 Operations

Operation of Backup and Recovery:

1. **Backup Configuration**:
   General Operation: Regular backups are crucial for data integrity and recovery. Backup settings must be configured, including the frequency of backups, types of data to be backed up, and the destination for storing backup files.
   Specific Steps: Defining backup schedules, specify whether full or incremental backups are needed, and set up a secure location for storing backup files.
2. **Recovery Process**:
   General Operation: In the event of data loss or system failure, the admin must initiate the recovery process to restore the system to a previous state. This involves accessing backup files and reinstating the necessary data.
   Specific Steps: Identify the backup version to restore, initiate the recovery process, and follow step-by-step instructions to ensure a successful recovery.

Operation of General System Configuration:

1. **Database Migration**:
   General Operation: When transitioning to a new version of the application or migrating to a different database system, the admin may need to perform database migration.
   Specific Steps: Execute migration scripts, update database configurations, and verify data integrity post-migration.
2. **Configuration Updates**:
   General Operation: Configuration changes may be necessary due to evolving business requirements or system enhancements.
   Specific Steps: Access the system configuration interface, make necessary adjustments, and validate the impact of changes on the system's functionality.

Operation of Routine Maintenance:

1. **Cache Management**:
   General Operation: Caches may be employed to enhance performance. Users may need to manage and clear caches regularly.
   Specific Steps: Access cache management tools, clear unnecessary caches, and monitor cache performance.

2. **Data Archiving**:
   General Operation: In order to optimize database performance, the users may need to archive historical data periodically.
   Specific Steps: Identify and archive historical data, update database references, and ensure that archived data is stored securely.

Operation of User Account Management:

1. **User Registration and Authentication**:
   General Operation: In order to ensure secure access, the admin need to authenticate themselves.
   Specific Steps: New admins have to be approved and then provide the required information, and existing users authenticate using Google OAuth 2.0.
2. **User Role Configuration**:
   General Operation: System administrators configure user roles to manage access levels.
   Specific Steps: Define user roles (e.g., admin, regular user) with specific permissions, assign roles to users, and update role configurations as needed.

2.5 Site adaptation requirements

Adjustment Costs and Constraints for Specific Installations:

1. **System Configuration for Local Deployments**:
   Adjustment Costs: If the users choose to deploy ERoots locally, they may incur costs associated with configuring the system to match the specific requirements of their local infrastructure.
   Constraints: Local installations may face constraints related to hardware compatibility, network configurations, and security policies specific to the user's environment.
2. **Integration with Local Databases or Services**:
   Adjustment Costs: Users integrating Evergreen Roots with local databases or services may incur costs related to data migration, configuration changes, and potential development efforts.
   Constraints: Compatibility issues, data format disparities, and access restrictions may pose constraints during the integration process.
3. **Scalability Adjustments**:
   Adjustment Costs: Scaling the system to accommodate increasing user loads or growing datasets may involve costs related to infrastructure upgrades, optimization efforts, and performance testing.
   Constraints: Infrastructure constraints, budget limitations, and technical considerations may impact the extent to which the system can scale.
4. **Third-Party Software Integration**:
   Adjustment Costs: Integrating Evergreen Roots with third-party software may incur costs related to licensing, development efforts, and compatibility checks.
   Constraints: Compatibility issues, API limitations, and dependency management may pose constraints during third-party integrations.

2.6 User characteristics

User Personas:

1. **End User**
   Background: Represents the diverse community of consumers in Kenya, with a focus on supporting local entrepreneurs in agriculture.
   Education: Varied, ranging from basic to higher education levels.
   Experience: Familiar with local agricultural practices, sustainability, and community-driven initiatives.
   Technical Knowledge: Basic to moderate technical proficiency, comfortable using smartphones for everyday tasks.
   Goals: To effortlessly discover, explore, and express interest in high-quality agricultural products from local women entrepreneurs. Values authenticity, community support, and sustainable practices.

2. **Admin User**
   Background: Manages the administrative functions of the Evergreen Roots eCommerce Platform.
   Education: Moderate to high educational qualifications.
   Experience: How to oparate and manage a secure web application and project management skills.
   Technical Knowledge: Basic computer literacy, adept at using administrative tools provided by the platform.
   Goals: Efficiently manage products and orders through the admin dashboard, ensuring the empowerment of women entrepreneurs.

Relevant Factors in the Use Case Model:

1. **User-Friendly Interface**:
   End User: Ensure an enjoyable browsing experience for expressing interest in products.
   Admin User: Design an efficient admin dashboard for streamlined product and order management.

2. **Product Information and Transparency**:
   End User: Provide comprehensive product information, emphasizing transparency in pricing and origin.
   Admin User: Access to detailed product information for effective platform management.

3. **Mobile Accessibility**:
   End User: Optimize the platform for mobile accessibility.
   Admin User: Ensure the admin dashboard is responsive across different devices.

4. **Person-to-Person Payment System**:
   End User: Facilitate secure person-to-person payments.
   Admin User: Monitor and manage payment transactions securely.

5. **Appointment Scheduling**:
   End User: Allow users to schedule appointments for product purchases.
   Admin User: Manage and coordinate appointments efficiently.

2.7 Product Functions Summary

Evergreen Roots' web application helps Kenyan people in selling their farmed goods and gives the users a great experience.

2.7.1 End User Functions:

Discover Products

Description: End users can explore a variety of high-quality agricultural products.

Relationships: Linked to "View Product Information" and connected to "Schedule Appointment"

2.7.2 Admin Functions:

Manage Products

Description: Administrators can oversee the platform's product listings and information.

Relationships: Associated with "Update Product Information"

Coordinate Appointments

Description: Admins can schedule and manage appointments for product viewings and pickups.

Relationships: Connected to "View Scheduled Appointments".

2.7.3 Common Functions:

View Product Information

Description: Both end users and admins can access detailed information about listed products.

Relationships: Linked to "Discover Products"

Schedule Appointment

Description: Users, as well as administrators, can arrange appointments related to product interactions.

Relationships: Linked to "Coordinate Appointments" and "View Scheduled Appointments."

2.7.4 Use Case Diagram:

The use case diagram visually represents the relationships and interactions between the key functions of the project.

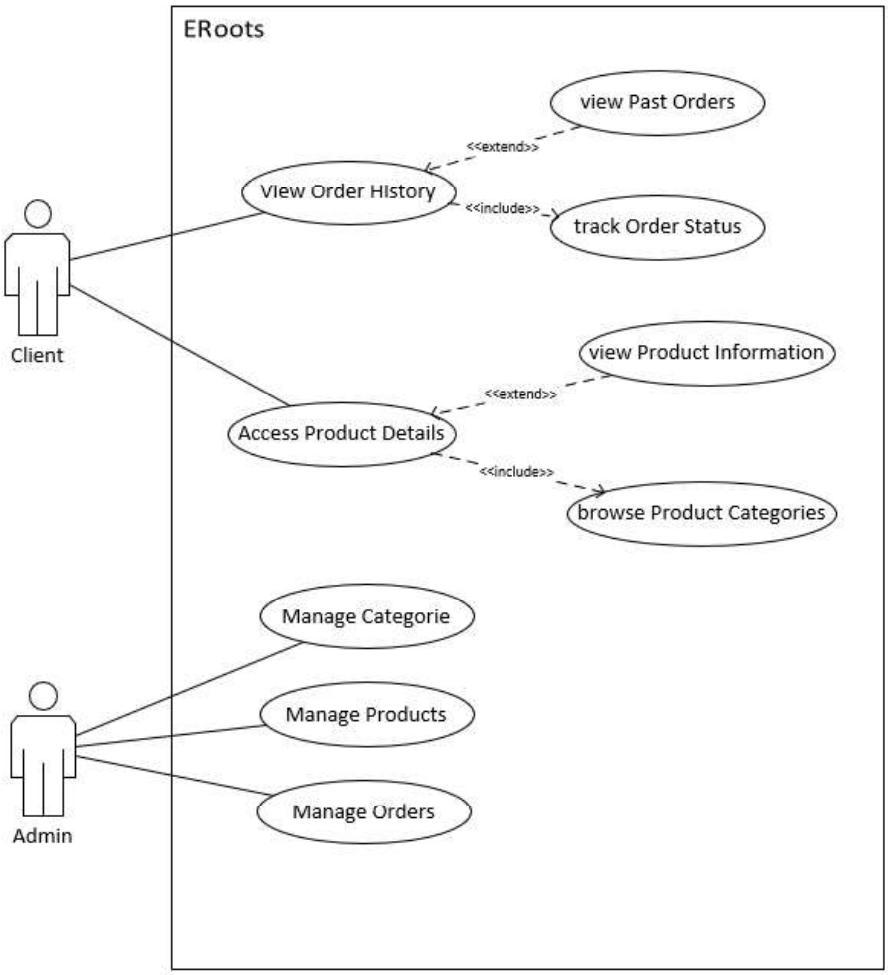


fig. 5 ERoots UseCase

2.7.5 Activity diagram:

This section is used to show the sequence of activities or actions within this system.

2.7.5.1 Admin management for products

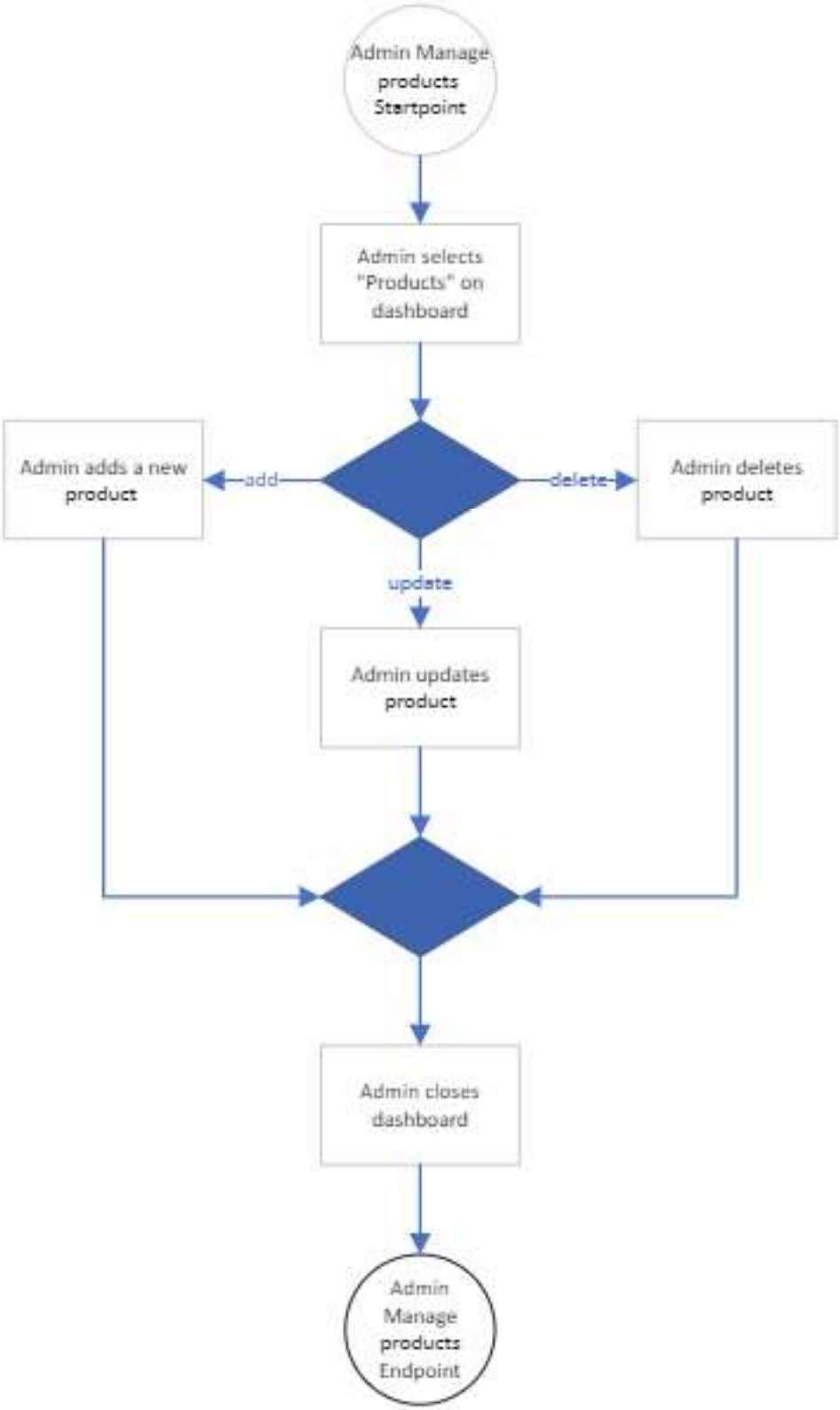


fig. 6 admin management for products

2.7.5.2 Admin management for orders

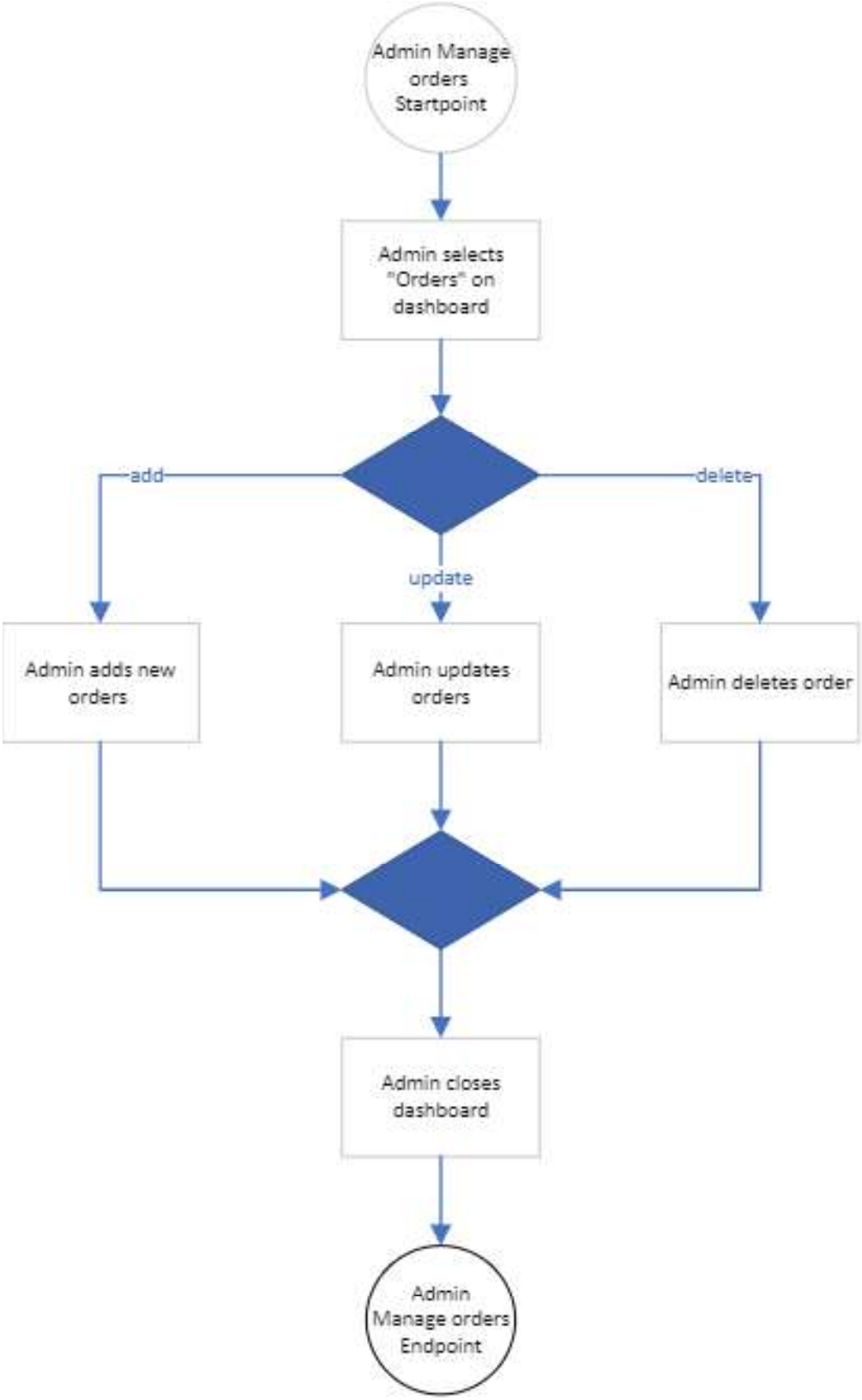


fig. 7 admin management for orders

2.7.5.3 Admin management for categories

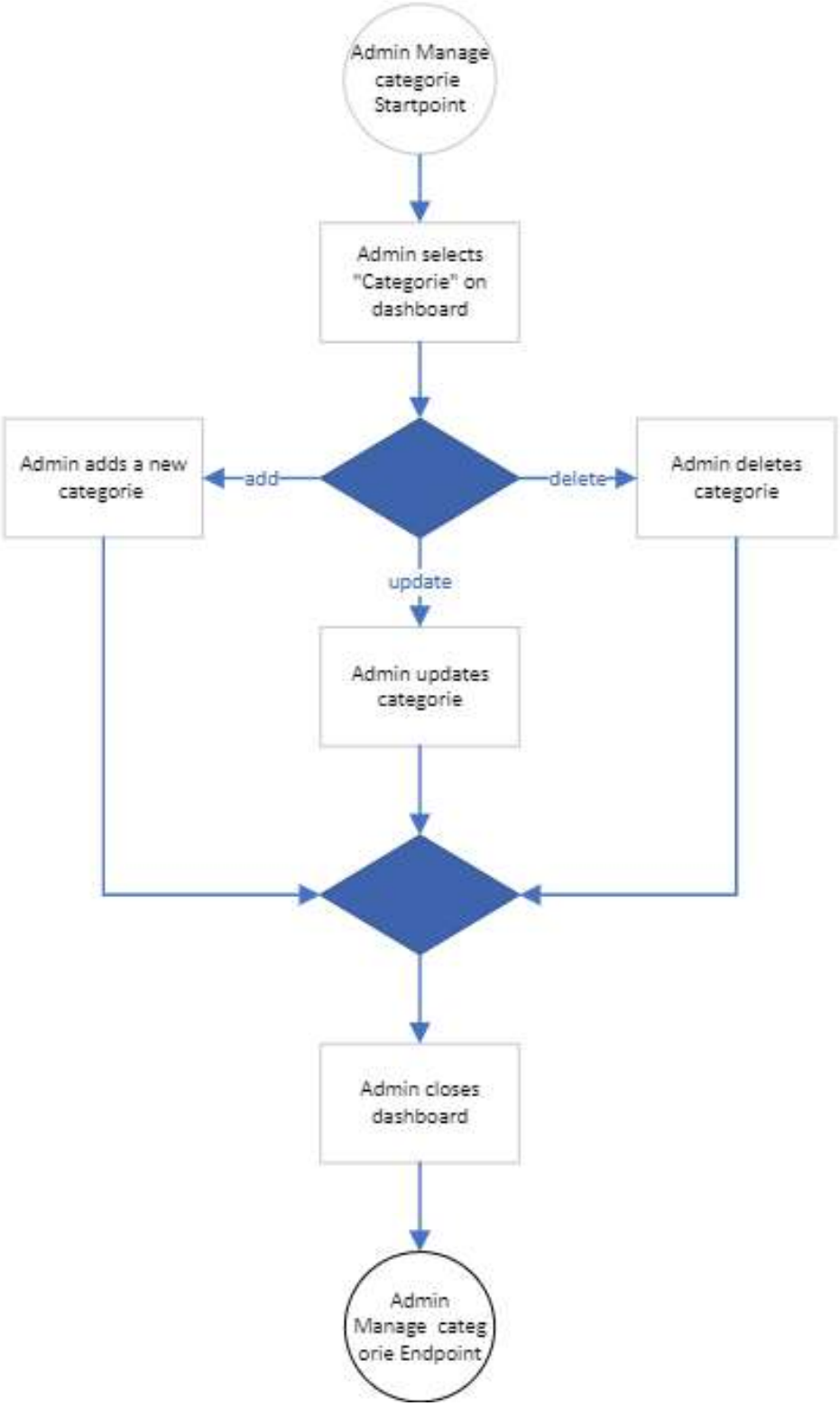


fig. 8 Admin management for categories

2.7.5.4 User View Order History

fig. 9 User view order history

2.8 Documentation requirements

In this part of the document a list of essential documents that should be created for the Evergreen Roots project, outlining their contents, intended audience, size, and format can be viewed.

1. **Requirement Specification Document**
   Contents: Detailed specifications of the project's functional and non-functional requirements, user stories, and use cases.
   Intended Audience: Development team, stakeholders, and quality assurance team.
   Size: Comprehensive, covering all aspects of the project requirements.
   Format: PDF or Word document with sections for easy navigation.

2. **Project Manual**
   Contents: Guidelines on project organization, roles and responsibilities, project timelines, and communication protocols.
   Intended Audience: Project team members, stakeholders, and project managers.
   Size: Medium-sized, providing sufficient detail for effective project management.
   Format: PDF or Word document.

2.9 Constraints

This part of the document lists some of the project’s constraints that should be followed.

1. **Language Constraints**:
   Description: The web application should primarily support English for consistency and broad accessibility.
   Control Rule: All user interfaces, product information, and communications should be in English.

2. **Security Aspects**:
   Description: Ensuring the confidentiality, integrity, and availability of user and transaction data.
   Control Rule: Implement encryption protocols, secure authentication mechanisms, and regular security audits.

3. **Network Connectivity**:
   Description: Users may have inconsistent or limited internet access.
   Control Rule: Optimize the platform for offline use where possible, allowing users to view products or express interest without a continuous internet connection.

4. **Regulatory Compliance**:
   Description: Adherence to local and international regulations related to eCommerce, data protection, and consumer rights.
   Control Rule: Regularly update the platform to comply with evolving legal requirements and standards.

5. **User Authentication**:
   Description: Secure authentication mechanisms to protect user accounts and data.
   Control Rule: Utilize multi-factor authentication and secure login protocols.

6. **Data Privacy**:
   Description: Safeguarding user data and ensuring compliance with data protection laws.
   Control Rule: Implement robust data privacy measures, including anonymizing or pseudonymizing sensitive information.
7. **Scalability**:
   Description: Preparing the platform for potential growth in user base and transaction volume.
   Control Rule: Design the platform architecture to handle increased loads and scale resources as needed.
8. **Mobile Accessibility**:
   Description: Ensuring that the platform is accessible and user-friendly on various mobile devices.
   Control Rule: Test and optimize the platform for mobile responsiveness and usability.

2.10 Deliverables

The components of the Evergreen Roots project will be delivered to the project sponsor in a digital format, eliminating the need for physical distribution, e.g. via CDs or USB sticks.

1. **Web Application**
   Delivery Form: Accessible through any standard web browser.
   Details: The central application, including the admin dashboard, client dashboard, and product/order management tools, will be available online.
2. **Installation and Setup**
   Delivery Form: No local software installation required.
   Details: Users can seamlessly access the application by visiting the designated website, eliminating the need for physical installation.
3. **Updates and Maintenance**
   Delivery Form: Handled by the Evergreen Roots development team.
   Details: Software updates and maintenance tasks will be managed by the development team, ensuring users do not need to perform manual updates.
4. **Hosting**
   Delivery Form: Web application hosted on a secure and reliable server.
   Details: The web application will be hosted on a secure server, ensuring 24/7 availability and optimal performance for the customer.

2.11 Assumptions and Dependencies

1. **Internet Connectivity**
   Assumption: Users have reliable internet access.
   Impact of Change: If there are significant changes in internet availability, the platform may need to adapt to accommodate users with varying connectivity.
2. **Device Compatibility**
   Assumption: Users access the platform through devices with standard web browsers.
   Impact of Change: Introduction of new devices or changes in device preferences may require adjustments to ensure continued compatibility.

3. **Regulatory Environment**
   Assumption: Compliance with existing eCommerce and data protection regulations in Kenya.
   Impact of Change: Changes in regulatory requirements may necessitate updates to the platform to ensure legal compliance.

4. **Security Protocols**
   Assumption: Robust security protocols are in place.
   Impact of Change: Evolving security threats may require enhancements to the system's security measures to maintain data integrity and user trust.

5. **Web Browser Standards**
   Assumption: Adherence to current web browser standards.
   Impact of Change: Updates or changes to web browser standards may require adjustments to the platform for optimal performance and user experience.

6. **User Base Growth**
   Assumption: Gradual growth in the user base.
   Impact of Change: A sudden surge in users may necessitate scaling considerations and potential adjustments to handle increased loads.

7. **Technological Advancements**
   Assumption: Use of contemporary web development technologies.
   Impact of Change: Introduction of new technologies may offer opportunities for system enhancements, but it may also require adaptation to remain current and competitive.

8. **Accessibility Guidelines**
   Assumption: Adherence to current accessibility standards.
   Impact of Change: Changes in accessibility guidelines may require updates to ensure the platform remains inclusive for users with disabilities.

9. **Third-Party Integrations**
   Assumption: Smooth integration with existing third-party services.
   Impact of Change: Changes in third-party APIs or services may require adjustments to maintain seamless integrations.

10. **User Expectations**
    Assumption: User expectations align with the current platform features.
    Impact of Change: Shifting user expectations or market trends may influence the need for new features or modifications to existing ones.

2.12 Apportionment of Requirements

1. **User-Created and Edited Content**
   Postponement Reason: If the primary focus is on showcasing products initially, extensive user-generated content features, such as user-created and edited content, can be deferred to future versions.

2. **Product Reviews by Users**
   Postponement Reason: While user reviews are valuable, if the initial release prioritizes core features, the ability for users to provide reviews for products can be considered for a subsequent version.

3. **Coupon and Discount Code Usage**
   Postponement Reason: Integrating coupon and discount code functionality may require additional development effort. If the initial goal is to establish the basic eCommerce framework, this feature can be deferred to later releases.
4. **Wish List Functionality**
   Postponement Reason: Adding products to a wish list is a valuable but non-essential feature. If the core product management and ordering functionalities are prioritized, wish list functionality can be considered for future releases.
5. **User Newsletter Registration**
   Postponement Reason: While user engagement through newsletters is important, if the immediate focus is on core eCommerce functionality, user registration for newsletters can be delayed to later versions.

3 Specific requirements

This section provides an overview of the project, outlining the roles of the team members and the scope of the software product.

3.1 External interfaces

3.1.1 Display Products (/01/)

Inputs:
- User request to view products

Outputs:
- Displayed product list with titles, images, prices

Data Flow:
- User interaction triggers a query to the database
- Database sends product details to the user interface

Formal Aspects:
- Information Flow: User request → Database query → Displayed product list
- Data Format: JSON, XML, or HTML format for product details
- Timing: Real-time on user request

3.1.2 Product Details (/02/)

Inputs:
- User selection of a specific product

Outputs:
- Detailed product information (specifications, pricing, images)

Data Flow:
- User selection triggers a query for specific product details
- Database sends detailed product information to the user interface

Formal Aspects:
- Information Flow: User selection → Database query → Displayed detailed product information
- Data Format: JSON, XML, or HTML format for detailed product information
- Timing: Real-time on user selection

3.1.3 Shopping Cart Functionality (/03/)

Inputs:

- User actions to add, increase, decrease, remove, or view items in the cart

Outputs:

- Updated cart content or cart summary

Data Flow:

- User interacts with the cart interface (add, increase, decrease, remove, view)
- Backend updates the cart content in real-time and sends the updated information to the UI

Formal Aspects:

- Information Flow: User actions → Backend processing → Updated cart information
- Data Format: JSON or HTML for cart details
- Timing: Real-time interaction

3.1.4 Place Orders (/04/)

Inputs:

- User-initiated action to confirm and place an order

Outputs:

- Confirmation of order placement or order summary

Data Flow:

- User completes checkout process, providing order details (name, email, date and time).
- Backend processes order, updates database, and confirms successful order placement
- Backend sends confirmation email to User and Admin

Formal Aspects:

- Information Flow: User order details → Backend processing → Confirmation or order summary
- Data Format: Order details (JSON or form-encoded data)
- Timing: Immediate upon successful order submission

3.1.5 Product Search (/05/)

Inputs:

- User-entered search query or category selection

Outputs:

- List of products matching the search query or category

Data Flow:

- User enters search keywords or selects a category
- Backend queries the database based on the input and sends search results to the UI

Formal Aspects:

- Information Flow: User search query → Backend processing → Displayed search results
- Data Format: JSON or HTML format for search results
- Timing: Real-time on user query

3.2 Admin Interface

3.2.1 Create New Products (/06/)

Inputs:
- Admin-provided details for new product creation

Outputs:
- Confirmation of successful product addition

Data Flow:
- Admin fills product creation form and submits
- Backend processes form data, adds new product details to the database, and sends a success message

Formal Aspects:
- Information Flow: Admin-provided data → Backend processing → Confirmation message
- Data Format: Form data (JSON, XML, or form-encoded data)
- Timing: Immediate upon successful submission

3.2.2 Edit Product Details (/07/)

Inputs:
- Admin-requested modifications to existing product information

Outputs:
- Updated product information or confirmation of changes

Data Flow:
- Admin accesses product details, modifies information, and saves changes
- Backend updates product data in the database and provides confirmation

Formal Aspects:
- Information Flow: Admin-modified data → Backend processing → Confirmation message or updated data
- Data Format: Form data (JSON, XML, or form-encoded data)
- Timing: Immediate upon successful update

3.2.3 Delete Products (/08/)

Inputs:
- Admin-initiated action to remove products from the system

Outputs:
- Confirmation of successful product deletion

Data Flow:
- Admin selects products for deletion
- Backend removes product data from the database and confirms successful deletion

Formal Aspects:
- Information Flow: Admin action → Backend processing → Confirmation message
- Data Format: Form data (JSON, XML, or form-encoded data)
- Timing: Immediate upon successful deletion

3.2.4 Create New Categories (/09/)

Inputs:
- Admin-provided details for new category creation

Outputs:
- Confirmation of successful category addition

Data Flow:
- Admin fills category creation form and submits
- Backend processes form data, adds new category details to the database, and sends a success message

Formal Aspects:
- Information Flow: Admin-provided data → Backend processing → Confirmation message
- Data Format: Form data (JSON, XML, or form-encoded data)
- Timing: Immediate upon successful submission

3.2.5 Edit Category Details (/10/)

Inputs:
- Admin-requested modifications to existing category information

Outputs:
- Updated category information or confirmation of changes

Data Flow:
- Admin accesses category details, modifies information, and saves changes
- Backend updates category data in the database and provides confirmation

Formal Aspects:
- Information Flow: Admin-modified data → Backend processing → Confirmation message or updated data
- Data Format: Form data (JSON, XML, or form-encoded data)
- Timing: Immediate upon successful update

3.2.6 Delete Category (/11/)

Inputs:
- Admin-initiated action to remove categories from the system

Outputs:
- Confirmation of successful category deletion

Data Flow:
- Admin selects categories for deletion
- Backend removes category data from the database and confirms successful deletion

Formal Aspects:
- Information Flow: Admin action → Backend processing → Confirmation message
- Data Format: Form data (JSON, XML, or form-encoded data)
- Timing: Immediate upon successful deletion

3.3 Detailed functionality description

This part of the document describes the project's functionalities in more detail.

3.3.1 Product Page

3.3.1.1 Function /01/: Displaying all products on a page:

Layout: Grid or list view showing product thumbnails, titles, and prices.
Interaction: Clickable elements for each product to navigate to the detailed product page.

3.3.1.2 Function /02/: Showing detailed information about a selected product when clicked:

Detailed View: Displaying the selected product's title, description, price, images, and additional details.
Interaction: Add to cart button to include the product in the shopping cart.

3.3.2 User Account

3.3.2.1 Function /04/: Allowing users to register using their Google account:

Google OAuth Integration: Permitting users to log in or register using their Google account credentials.
Account Association: Linking the Google account information with the user's account in the system.

3.3.3 Order Products

3.3.3.1 Function /05/: Implementing a shopping cart functionality to add products:

Cart Management: Allowing users to add, remove, and update quantities of selected products in the shopping cart.
Persistence: Saving cart data to ensure continuity across user sessions.

3.3.3.2 Function /06/: Enabling users to place orders through the system:

Checkout Process: Guiding users through the steps to finalize an order, including address entry, payment method selection, and order confirmation.

3.3.4 Order History

3.3.4.1 Function /07/: Providing a view for users to see their previous order history:

Order Summary: Displaying a list of past orders with details like order date, items purchased, and order status.
Order Details: Accessing more detailed information about each past order.

3.3.5 Search Functionality

3.3.5.1 Function /08/: Implementing a search feature to find products based on keywords or categories:

Search Bar: Allowing users to enter keywords or select categories to find specific products.
Filtering Mechanism: Displaying search results based on the entered keywords or selected categories.

3.3.6 Product Management

3.3.6.1 Function /09/: Allowing administrators to create new products:

Admin Interface: Providing a form to input product details such as title, description, price, images, and category.

3.3.6.2 Function /10/: Enabling administrators to edit existing product details:

Edit Functionality: Allowing admins to modify product information and update details as needed.

3.3.6.3 Function /11/: Allowing administrators to delete products from the system:

Deletion Process: Confirming admin action to delete a product, including handling associated data (like orders referencing the product).

3.3.7 Category Management

3.3.7.1 Function /12/: Permitting administrators to create new product categories:

Category Creation: Allowing admins to define new categories and assign products to these categories.

3.3.7.2 Function /13/: Allowing administrators to edit existing category details:

Edit Functionality: Allowing modifications to category names or properties.

3.3.7.3 Function /14/: Enabling administrators to delete categories from the system:

Category Deletion: Handling category deletions and reassignment of products to other categories if necessary.

3.4 Classes and modules

Class Diagram for eCommerce Application:

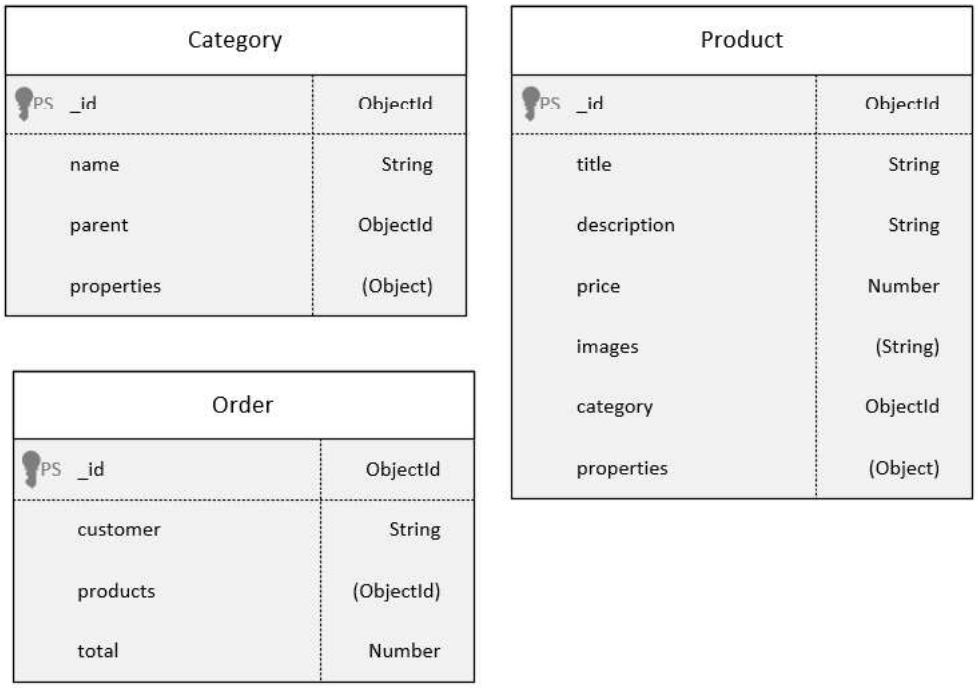


fig. 10 Class Diagram Category, Product and Order

Class Descriptions:

1. **Category**:

1.1 Attributes

`_id`: Unique identifier.
`name`: Category name.
`parent`: Parent category reference.
`properties`: List of category-specific properties.

1.2 Methods

`fetchCategories()`: Fetches categories from the server.
`saveCategory()`: Saves or updates a category.
`editCategory()`: Edits a category.
`deleteCategory()`: Deletes a category.

2. **Product**:

2.1 Attributes

`_id`: Unique identifier.
`title`: Product title.
`description`: Product description.
`price`: Product price.
`images`: List of image URLs.
`category`: Category reference.
`properties`: List of product-specific properties.

2.2 Methods
`fetchProducts()`: Fetches products from the server.
`saveProduct()`: Saves or updates a product.
`editProduct()`: Edits a product.
`deleteProduct()`: Deletes a product.

3. **Order**:
3.1 Attributes
`_id`: Unique identifier.
`customer`: Customer's email.
`products`: List of product references in the order.
`total`: Total order value.
3.2 Methods
`placeOrder()`: Places a new order.
`fetchOrders()`: Fetches orders from the server.

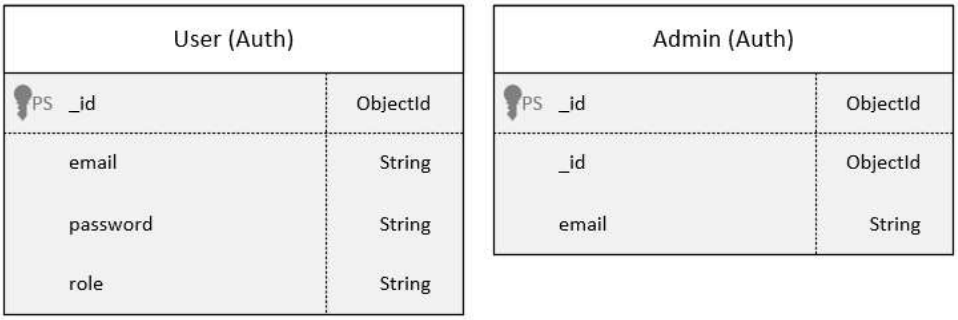


fig. 11 Class Diagram User and Admin

4. **User (Auth) & Admin (Auth)**:
4.1 Attributes:
`_id`: Unique identifier.
`email`: User/admin email.
`password`: User/admin password (hashed).
`role`: Role of the user/admin (user or admin).

3.5 Database Requirements

In this section the database and models are explained in more detail.

3.5.1 Data model and files

3.5.1.1 Users:

1.1 Attributes:
`_id`: Unique identifier for a user.
`name`: User's full name.
`email`: User's email address.
`image`: URL or path to the user's profile image.
`emailVerified`: Boolean indicating whether the user's email is verified.
Relationships: No direct relationships in this model; however, other collections such as Accounts reference the user through the userID field.

3.5.1.2 Accounts:

Attributes:

_id: Unique identifier for an account.
provider: Name or identifier of the account provider (e.g., Google, Facebook).
type: Type of account (e.g., OAuth, local account).
providerAccountID: Account ID provided by the account's service.
access_token: Token for accessing the account's resources.
expires_at: Expiry timestamp for the access token.
scope: Scope of access for the token.
token_type: Type of the token.
id_token: Identity token associated with the account.
userID: Reference to a user in the Users collection.
Relationships: References the userID field to associate an account with a specific user in the Users collection.

3.5.1.3 Categories:

Attributes:

_id: Unique identifier for a category.
name: Name of the category.
properties: Array containing objects with name and values fields describing category-specific properties.
Relationships: No direct relationships in this model. Categories are standalone entities that can be referenced by products.

3.5.1.4 Orders:

Attributes:

_id: Unique identifier for an order.
line_items: Array of objects containing details of items in the order.
email: Email associated with the order.
createdAt: Date indicating when the order was created.
updatedAt: Date indicating the last update to the order.
Relationships: Contains order details (line_items) that reference products in the Products collection. Each line_item contains details such as product information, quantity, etc.

3.5.1.5 Products:

Attributes:

_id: Unique identifier for a product.
title: Title or name of the product.
description: Description providing details about the product.
price: Price of the product.
images: Array of URLs pointing to images of the product.
category: Reference to a Category in Categories collection.
properties: Object containing product-specific properties.
Relationships: References a category from the Categories collection to categorize the product. Each product can belong to a single category.

3.5.2 Table definition

Orders								
_id	ObjectId							
line_items	Array							
	line_items							
name	String	Object	quantity	Int32				
email	String		price_data	Object	price_data			
createdAt	Date				currency	String		
updatedAt	Date				product_data	Object	product_data	
							name	String
					unit_amount	Int32		

fig. 16 Table Orders

Users	
_id	ObjectId
name	String
email	String
image	String
emailVerified	Boolean

fig. 15 Table Users

Sessions	
_id	ObjectId
sessionToken	String
userId	ObjectId
expires	Date

fig. 13 Table Sessions

Accounts	
_id	ObjectId
provider	String
type	String
providerAccountID	String
access_token	String
expires_at	Int32
scope	String
token_type	String
id_token	String
userID	ObjectId

fig. 14 Table Accounts

Products			
_id	ObjectId		
titel	String		
description	String		
price	Int32		
Images	Array		
	Images		
catagory	ObkectId	0	String
properties	Object		
	properties		
	amount	String	

fig. 12 Table Products

3.4.3 Database Model

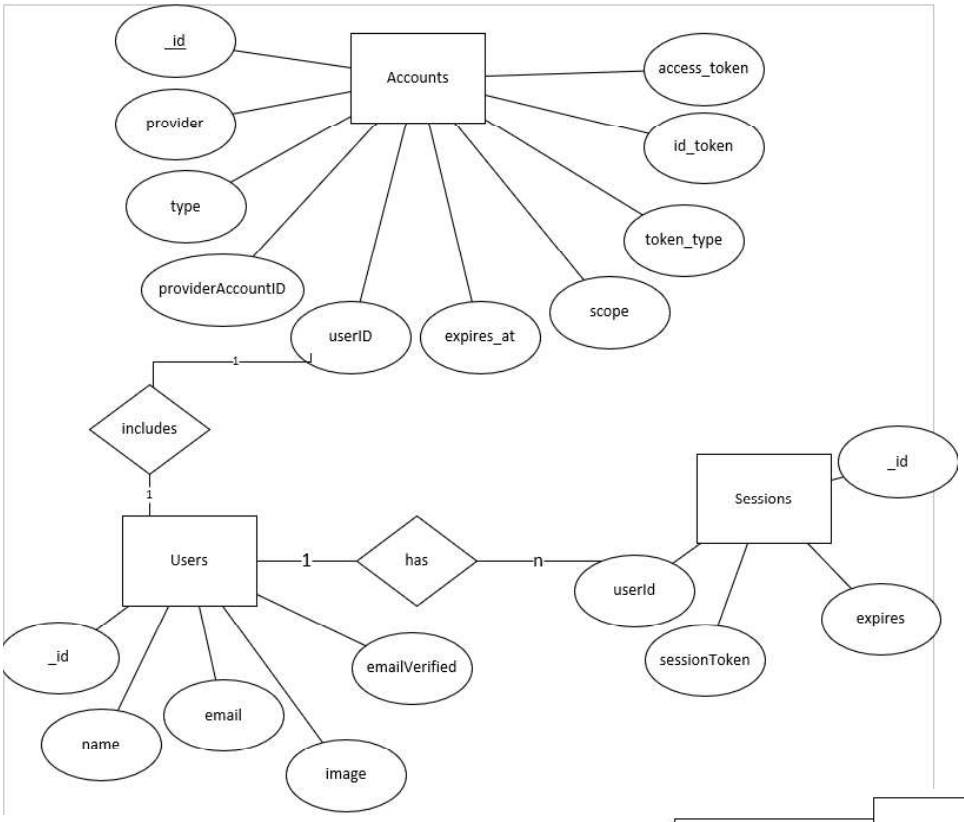


fig. 18 Database Modell

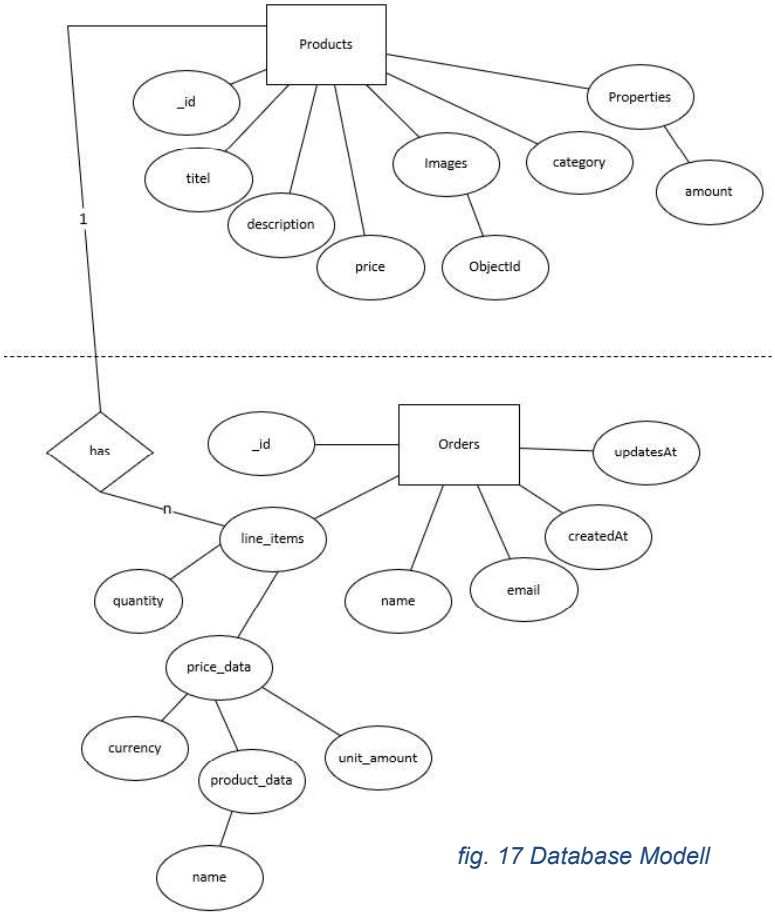


fig. 17 Database Modell

test

- accounts
- categories
- orders
- products
- sessions
- users

fig. 19 Database

```
_id: ObjectId('64786c1be6fd507b56b86f13')
provider: "google"
type: "oauth"
providerAccountId: "110819554338172568481"
access_token: "ya29.a0AWY7Cklfb51gnBxnFhJ5iUjzmvlBO5TR3PwCOJQ0W5EMYU8ouNczIMIUNLAXghm…"
expires_at: 1685617194
scope: "https://www.googleapis.com/auth/userinfo.profile https://www.googleapi…"
token_type: "Bearer"
id_token: "eyJhbGciOiJSUzI1NiIsImtpZCI6IjYwODNkZDU5ODE2NzNmNjYxZmRlOWRhZTY0NmI2Zj…"
userId: ObjectId('64786c1be6fd507b56b86f12')
```

fig. 24 Accounts Modell

```
__v: 0
properties: Array
  0: Object
    name: "amount"
    values: Array
      0: "500 grams"
      1: " 1 kilogram"
      2: " 1.5 kilograms"
      3: " 2 kilograms"
      4: " other"
```

fig. 23 Catagories Modell

```
_id: ObjectId('65492ff3c0a42e707de6e4da')
line_items: Array
  0: Object
    quantity: 2
    price_data: Object
      currency: "KES"
      product_Data: Object
      unit_amount: 600
  1: Object
name: "Ann Wambui"
email: "everlynnjerii@gmail.com"
createdAt: 2023-11-06T18:26:59.087+00:00
updatedAt: 2023-11-06T18:26:59.087+00:00
__v: 0
```

fig. 22 Orders Modells

```
_id: ObjectId('64b13f569917efbf4c3e0054')
titel: "Black turtle bean"
description: "The black turtle bean is a small, shiny variety of the common bean esp…"
price: 3009
images: Array
  0: "https://eroots-next-ecommerce.s3.amazonaws.com/1697463452384.jpg"
__v: 0
category: ObjectId('64b12339655ab945d92b5212')
properties: Object
  amount: " 1 kilogram"
```

fig. 21 Product Modells

```
_id: ObjectId('6474f1e2a63d6ac70e653c99')
sessionToken: "62e833c9-9f25-4b59-ac24-a5169e019f18"
userId: ObjectId('6474f1e2a63d6ac70e653c97')
expires: 2023-06-29T18:45:53.329+00:00
```

fig. 20 Sessions Modell

```
_id: ObjectId('64786c1be6fd507b56b86f12')
name: "Everlyn Njeri"
email: "everlynnjerii@gmail.com"
image: "https://lh3.googleusercontent.com/a/AAcHTtcDCbhXw_bUsxy-q_IYbb28dA3PcM…"
emailVerified: null
```

fig. 25 User Modell

3.6 Performance requirements

This part covers the performance requirements of the project.

3.6.1 Static Demands

Product Catalogue size: ca. 75

3.6.2 Dynamic Demands

Concurrent Users: ca. 10 customers/hr at peak hours

Transaction Volume: ca. 10 transactions/hr at peak hours
Page Load Times: < 1 sec.

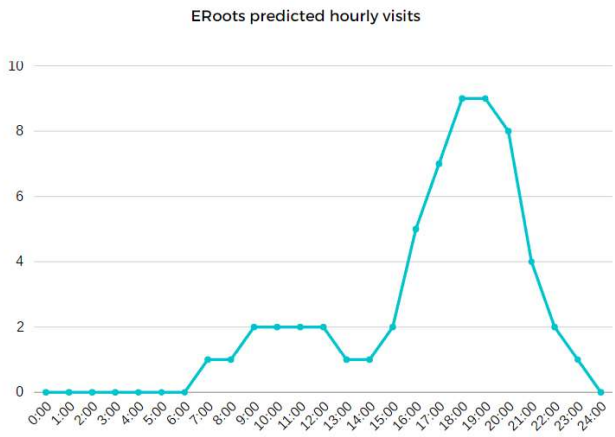


fig. 26 predicted hourly visits

3.7 Design constraints

Typical standards and constraints:

1. **Naming Conventions**:
   Design Restriction: Adherence to specific naming conventions for variables, functions, classes, and other elements.
2. **Reporting Formats**:
   Design Restriction: Compliance with standard reporting formats for consistency and interoperability.
   Example: Using a standardized format BSON for generating and exchanging reports.
3. **Protocols**:
   Design Restriction: Incorporating protocols for communication between system components or external systems.
   Example: Implementing RESTful API standards for communication between the front-end and back-end.
4. **Functions**:
   Design Restriction: Ensuring that certain functions align with established best practices or industry standards.
   Example: Adhering to SOLID principles for designing functions and classes.
5. **Regulatory Compliance**:
   Design Restriction: Meeting legal or regulatory requirements that impact the design of the system.
   Example: Implementing data privacy measures in compliance with relevant data protection regulations.
6. **Accessibility Standards**:
   Design Restriction: Ensuring that the user interface design complies with accessibility standards.
   Example: Designing web interfaces that adhere to WCAG guidelines for accessibility.
7. **Security Standards**:
   Design Restriction: Incorporating security measures and following best practices to protect against common vulnerabilities.
   Example: Implementing encryption protocols for secure data transmission.

3.8 Quality characteristics, Software system attributes

This section deals with the quality features and characteristics of the project's software system.

3.8.1 Performance

- **Responsiveness:** The system should respond to user interactions within 1-2 seconds, ensuring a seamless and engaging user experience. This includes rapid loading of pages and quick feedback for user actions.
- **Throughput:** The system's capacity to handle a specific number of concurrent transactions (see 3.5.2) will be defined and implemented, ensuring optimal performance during peak usage periods.
- **Scalability:** The system will be set up to scale horizontally and vertically, accommodating growing user bases and increased data loads. Load balancing strategies for effective resource distribution will be implemented.

3.8.2 Reliability

- **Availability:** High level of system availability to ensure users can access the application reliably will be provided.
- **Fault Tolerance:** The system will handle errors and unexpected failures, preventing widespread disruptions and providing a robust user experience.

3.8.3 Security

- **Authentication and Authorization:** Secure user authentication and authorization mechanisms to control access to sensitive features and data.

3.8.4 Usability

- **User Interface (UI) Design**: Consistent and intuitive UI design following user experience (UX) principles, enhancing user satisfaction and engagement.

3.8.5 Maintainability

- **Code Maintainability:** Coding standards and best practices to ensure code maintainability, making it easier for development teams to understand, update, and extend the codebase.
- **Modularity:** the system will be designed with a modular structure, promoting ease of updates and enhancements while minimizing the risk of unintended side effects.

4 Test cases

Cypress is crucial for ensuring our system works reliably. It'll conduct automated end-to-end and component tests with each commit, continuously validating our system's functionality.

4.1 4.1 E2E Testing

End-to-end tests simulate real user interactions with the system, covering the entire application flow. These tests ensure that various components and modules work seamlessly together.

4.1.1 Add to Basket

Objective: User Adds 3 Products to Basket

Scenario:

- Visit Homepage
- Add Products to Basket
- Visit Basket
- Check if all Products are in the Basket

4.2 Component Testing

Component tests focus on individual components or modules in isolation, ensuring they function correctly by themselves.

4.2.1 Navigation Bar

Objective: Validate the functionality of the navigation bar

Navigation Links:

- Click each navigation link
- Confirm user is redirected to correct page

Responsive Design:

- Test the navigation bar's behavior on various screen sizes.
- Ensure it remains functional and visually appealing.

4.3 Automated Testing Workflow

All the aforementioned test cases should be integrated into the automated testing workflow, triggered with every commit. Continuous Integration (CI) tools should be configured to execute Cypress tests automatically.

4.4 4.4 Test Reporting

Test Reports will be seen by members of the project on the cypress project overview site (https://cloud.cypress.io/). There detailed test results and reports will be shown after every commit or separate test run.

5 Appendix

This part of the document closes the system specification of the project.

5.1 Glossary

TERM	EXPLANATION
Accessibility	Designing the app to be user-friendly for individuals with disabilities.
Array	An array stores a bunch of items together, allowing easy access and manipulation of multiple elements within one variable
AWS S3 Bucket	A secure storage service for product images accessed via an access key for the web application.
BSON	A powerful testing framework for web applications, allowing automated tests for different parts of the app, like end-to-end and unit testing.
Continuous Integration (CI)	Frequently integrating code changes to a shared repository automates testing and helps catch issues early in software development.
Cypress	A powerful testing framework for web applications, allowing automated tests for different parts of the app, like end-to-end and unit testing.
Google OAuth 2.0	Secure admin login with Google accounts, supporting two-factor authentication.
Installation and Setup	No software installation needed; access via the website.
IONOS	Domain registrar managing "eroots.store" for web app accessibility.
JSON	JSON, short for JavaScript Object Notation, is a format used for easy data interchange between systems.
Mongoose	Streamlines interactions with the MongoDB database, enhancing efficiency.
MongoDB	Flexible and scalable NoSQL database storing data in a JSON-like format.
Node.js	Backend runtime for handling data, logic, and database communication.
Object	An object stores data in key-value pair e.g.: name: 'Alice', age: 30,
React.js	Frontend framework for creating interactive user interfaces.

RESTful API	A web interface that enables systems to communicate using standard web protocols like HTTP, following clear rules for requesting, modifying, and deleting information.
SOLID (Single, Open, Liskov, Interface, Dependency Principle)	SOLID principles are five guidelines in object-oriented programming for creating more maintainable and flexible software.
Tags	Tags label and categorize content, helping organize and identify information within documents or data structures.
Tailwind CSS	CSS framework for frontend styling and user-friendly design.
Updates and Maintenance	Handled by the development team to keep the app current.
User Interface	Responsible for user-friendliness and visual appeal in the app.
User Manual	A comprehensive resource guiding users on app usage and troubleshooting.
Vercel	Hosting platform known for fast and reliable web application deployment.
Web Application	Core component accessible through a web browser.
Web-Based Application	Delivered via the internet, accessible on any device without physical distribution.
XML	XML is a versatile text-based format used for storing and sharing data. It's like HTML but allows custom tags for defining data structures.

Project acceptance

SPENGERGASSE
ausbildung mit zukunft

Projectname:	Project Nr.:
Evergreen Roots	1

Client (represented by):	Contractor (represented by):
Pennina Njuguna	Njeri Everlyn, Paukner Sarah, Andrei Elias, Braun Valerian
Date of acceptance: 22.05.2023	Accepted version (optional): Version 1

During today's joint acceptance, it was determined that the delivered performance was in accordance with the contractually required condition, except for the defects identified below (according to the error classification description, see attachment).

Functions	Accepted	Remediation Period	Class
If 'Account' is selected in the navigation bar the user will be redirected to the Account page.	✓		
If the user is on the Account page, they can click the Login button and will be redirected to the Login page. There, they can log in to the website using an existing Google account.	✓		
The user can browse the homepage or All Products page and select the 'add to cart' function to place an order in the cart.	✓		
On the cart page, the user can schedule an appointment to view their ordered goods by providing their email address (erootsstore@gmail.com), name (Max Mustermann), and a future date and time.	✓		
If the user is logged in, his or her email address and name will be automatically filled into the order details.	✓		
When the user enters 'water' in the search bar, an option for 'watermelon' appears. By selecting it and pressing enter, the user is directed to the watermelon product page.	✓		
When logged into the admin dashboard, the user can create a new product called 'Black Beans'. They can then add a description, pictures, and categorise it as a 'legume' and 'black beans'. Once the product has been saved, the user can delete it by clicking the delete button.	✓		

When logged into the admin dashboard, the administrator can create a new category named e.g. ' Brassicaceae'. After clicking save, the catagorie is saved into the database.	✓		
After saving the category, the administrator can delete it by clicking the delete button.	✓		
A user can access the site by searching for eroots.store in the top search bar of any browser.	✓		
By searching for admin.eroots.store in the top search bar of any browser, an admin can access the administration website.	✓		
An unauthorized person cannot log in to admin.eroots.store due to a security measure in place.	✓		
If 'All Products' is selected in the navigation bar the user will be redirected to the All Products page.	✓		
If 'Cart' is selected in the navigation bar the user will be redirected to the Cart page.	✓		
If 'Home' is selected in the navigation bar the user will be redirected to the Home page.	✓		
If the name ERoots on the left side of the navigation bar is selected the user will be redirected to the Account page.	✓		

Comments and reservations:

Evelyn Ujeri Faulkner Sarah Jessica

Valerian Brann Andrei Elias

Date & Signature (Client)

Date & Signature (Contractor)

Appendix - Error Classes

During the acceptance process, existing defects may be categorized into error classes.

Error Class 1:
The practical use (economically viable use) is not possible, unreasonably restricted, or impossible due to such errors.

Error Class 2:
The practical use is not so significantly affected that the system could not still be used, possibly with interim error workaround routines. These errors are addressed as much as possible during the acceptance test.

Error Class 3:
The practical use is not or only minimally restricted by this error.

The assignment of errors to one of the above error classes is agreed upon between the contracting parties.

Errors of error Class 1 are considered "significant deviations", while errors of error Class 2 and 3 are considered "insignificant deviations". In case of significant deviations, the test will be conducted immediately after the deviation is corrected.

An insignificant deviation does not entitle the client to refuse acceptance. Remaining errors of all error classes after acceptance will be rectified within the warranty period according to a jointly established schedule.

If no significant errors (Error Class 1) occur during the functional test, the project to be accepted is considered accepted.

In case of significant defects (Error Class 1), the functional test is to be repeated after the correction.