Floris Steverink

551330@student.saxion.nl

Flofone website

FRONTEND AND BACKEND DOCUMENTATION

Contents

[Introduction about the project 2](#_Toc169572549)

[Frontend Documentation 3](#_Toc169572550)

[System Description 3](#_Toc169572551)

[Functionality 3](#_Toc169572552)

[User Authentication 3](#_Toc169572553)

[Product Management 3](#_Toc169572554)

[Shopping Cart 3](#_Toc169572555)

[Order Management 3](#_Toc169572556)

[Wireframes 4](#_Toc169572557)

[Product page (index.html) 4](#_Toc169572558)

[Cart page (cart.html) 4](#_Toc169572559)

[Login page (login.html) 5](#_Toc169572560)

[Register page (register.html) 5](#_Toc169572561)

[Dashboard main page / Order page (dashboard.html) 6](#_Toc169572562)

[Users page (users.html) 6](#_Toc169572563)

[User info page (userDetails.html) 7](#_Toc169572564)

[Order details page (orderdetails.html) 7](#_Toc169572565)

[Backend Documentation 8](#_Toc169572566)

[System Description 8](#_Toc169572567)

[Key Features 8](#_Toc169572568)

[REST Specification 8](#_Toc169572569)

[The REST Specification can be found in the Frontend folder. REST-Specification.html 8](#_Toc169572570)

[Sequence diagram 9](#_Toc169572571)

[User Login Process 9](#_Toc169572572)

[Add Product to Cart 9](#_Toc169572573)

[User Registration Process 10](#_Toc169572574)

[Fetching Products with Optional Category Filter 10](#_Toc169572575)

[Class Diagram 11](#_Toc169572576)

[Implementation Choices 12](#_Toc169572577)

[Frameworks and Libraries 12](#_Toc169572578)

[Architecture Choices 12](#_Toc169572579)

[Implementation Details 12](#_Toc169572580)

[Conclusion 13](#_Toc169572581)

[Testing 14](#_Toc169572582)

[General Testing Strategy 14](#_Toc169572583)

[Detailed Testing Plan 14](#_Toc169572584)

# Introduction about the project

This project concerns the development of a website for the company "FloFone". FloFone is a company that was previously conceived during the course "Organization and IT". The company focuses on selling phones and related accessories specifically for the elderly. The goal of FloFone is to make technology more accessible to the older generation by offering user-friendly phones and useful accessories.

The website is designed to allow potential customers to easily navigate through the product range, add products to their shopping cart and place orders. Since the company is Dutch, the website is completely in Dutch. The focus is on a simple and intuitive user experience, so that elderly people can use the online store without any hassle.

In addition, the website offers an administration panel that allows administrators to manage product information, view orders and perform user management. This project combines user-friendly frontend functionality with a robust backend to provide a seamless shopping experience for both customers and administrators of FloFone.

# Frontend Documentation

## System Description

The frontend of this e-commerce application is designed to provide a user-friendly interface for browsing products, managing a shopping cart, and placing orders. It interacts with the backend via RESTful API endpoints to perform various operations such as user authentication, product retrieval, and order processing. The frontend is built using HTML, CSS, and JavaScript.

## Functionality

The frontend provides the following key functionalities:

### User Authentication

* **Login**: Allows users to log in with their email and password. On successful login, a JWT token is stored in the browser's local storage.
* **Registration**: Allows new users to register by providing their name, email, and password.
* **Logout**: Allows users to log out by removing the JWT token from local storage.

### Product Management

* **Product Listing**: Displays a list of products retrieved from the backend. Users can filter products by category.
* **Product Details**: Provides detailed information about a selected product, including its name, description, price, and image.

### Shopping Cart

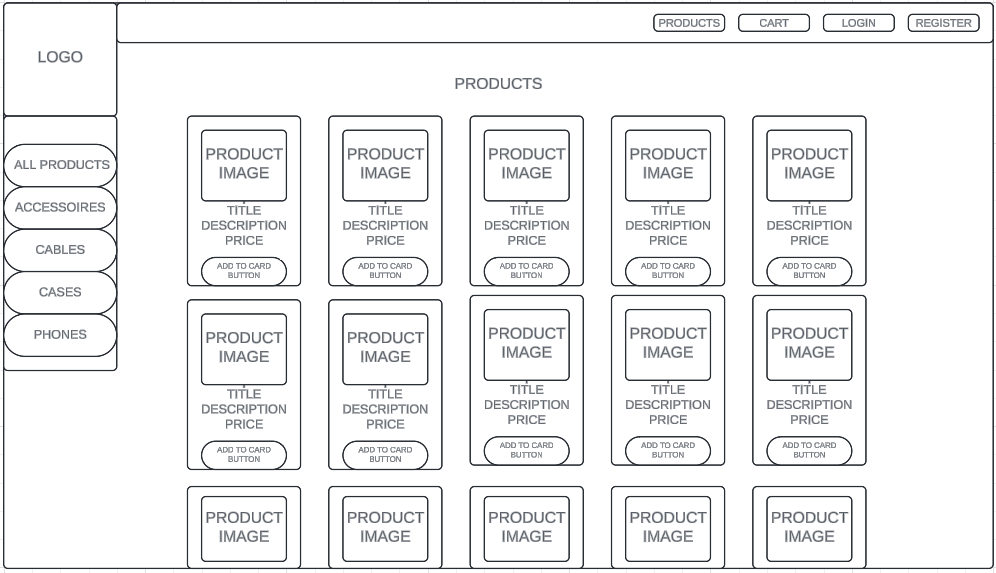
* **Add to Cart**: Allows users to add products to their shopping cart. If a product is already in the cart, the quantity is updated.
* **View Cart**: Displays the items in the user's shopping cart, including product details and quantities.
* **Remove from Cart**: Allows users to remove products from their cart.

### Order Management

* **Place Order**: Allows users to place an order by providing their shipping details and confirming the items in their cart.
* **View Orders**: Displays a list of orders placed by the user, including order details and status.

## Wireframes

### Product page (index.html)



### Cart page (cart.html)

A screenshot of a computer screen

Description automatically generated

### Login page (login.html)

A screenshot of a login form

Description automatically generated

### Register page (register.html)

A screen shot of a computer screen

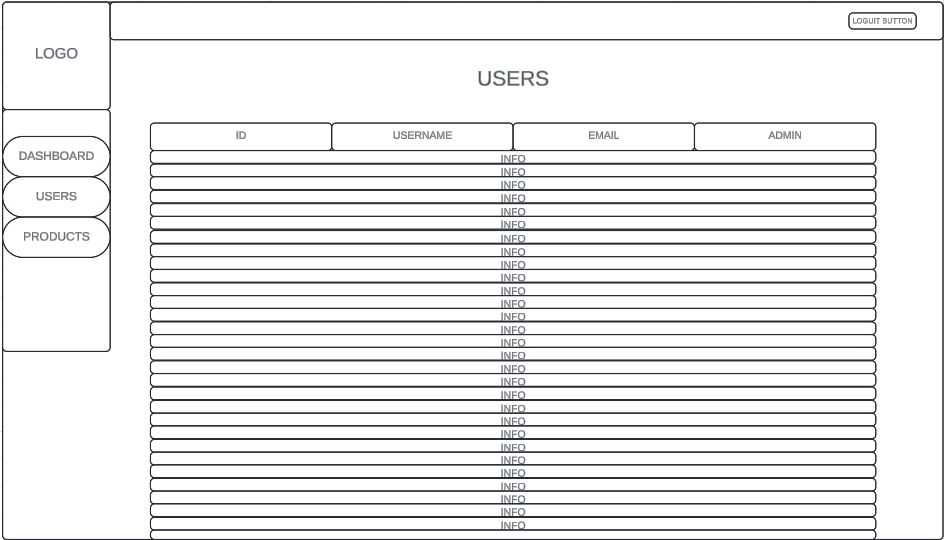
Description automatically generated

### Dashboard main page / Order page (dashboard.html)

A screenshot of a dashboard

Description automatically generated

## Users page (users.html)



### User info page (userDetails.html)

A screenshot of a form

Description automatically generated

### Order details page (orderdetails.html)

A screenshot of a computer

Description automatically generated

# Backend Documentation

## System Description

This backend system is designed to support an e-commerce web application. The backend is responsible for user authentication, product management, shopping cart operations, and order processing. It is built using Node.js with the Express framework and uses SQLite for data storage. The system employs JSON Web Tokens (JWT) for secure user authentication and Bcrypt for password hashing.

## Key Features

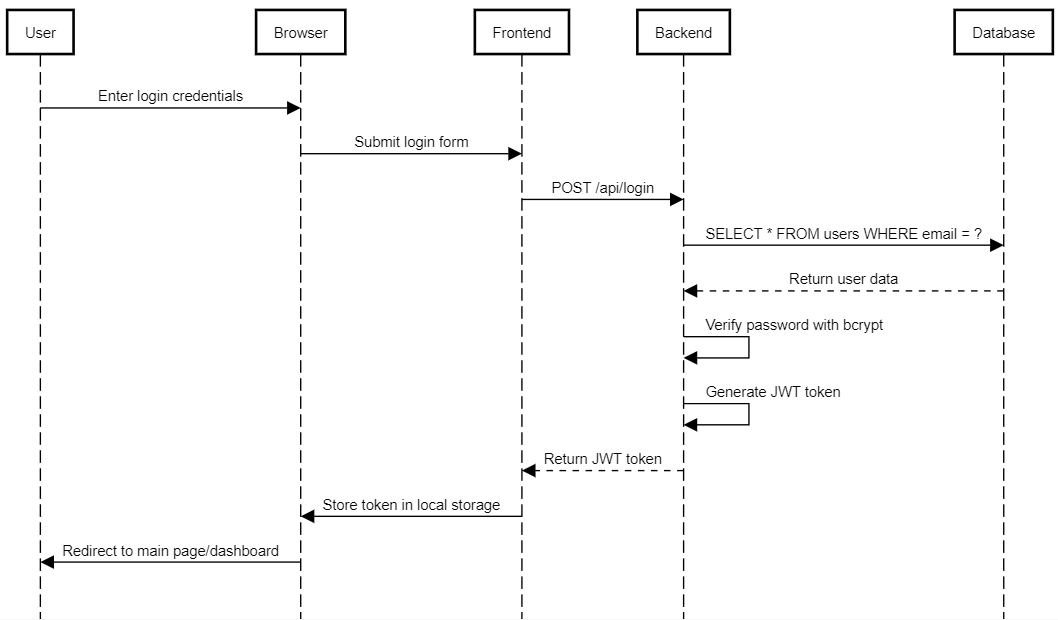
* User registration and login
* JWT-based authentication
* Product management with optional category filtering
* Shopping cart operations
* Order processing

## REST Specification

## The REST Specification can be found in the Frontend folder. REST-Specification.html

## Sequence diagram

### User Login Process

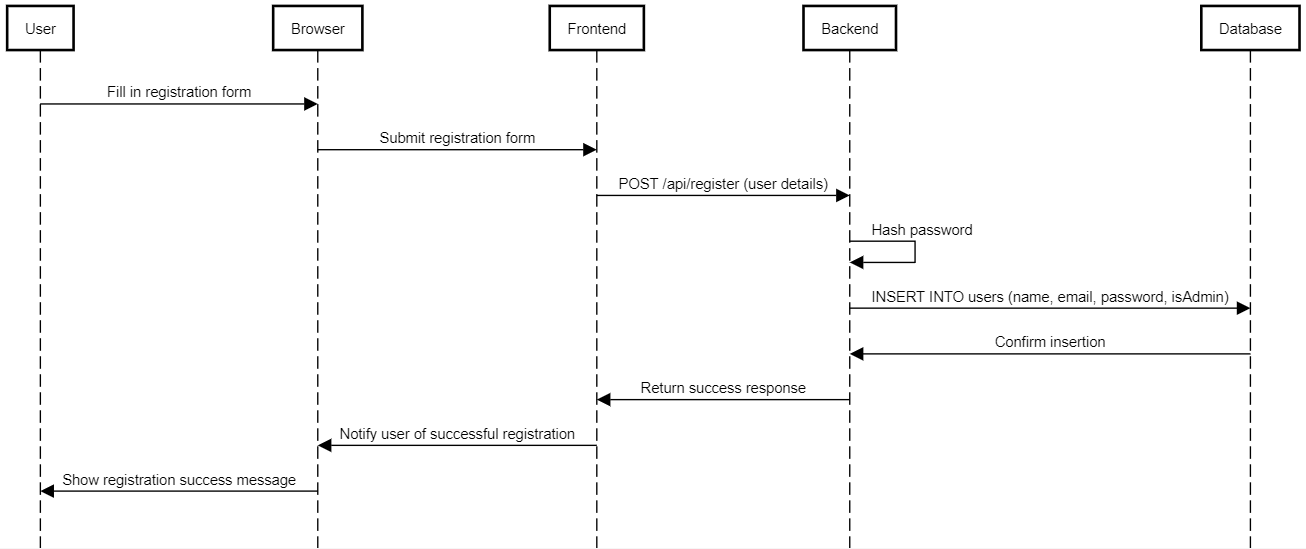


### Add Product to Cart

A diagram of a software project

Description automatically generated with medium confidence

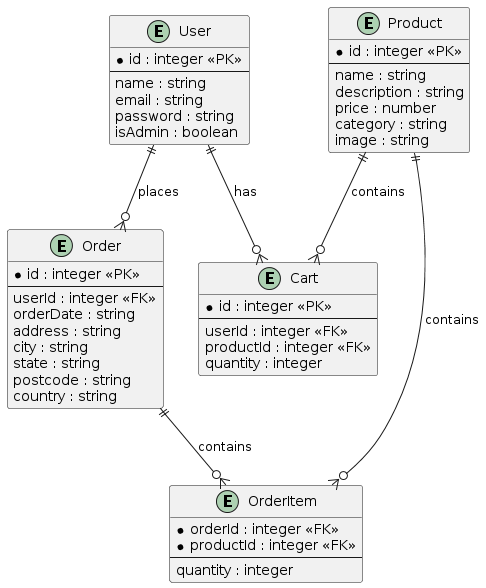
### User Registration Process



### Fetching Products with Optional Category Filter

A diagram of a company

Description automatically generated

Class Diagram

## Implementation Choices

### Frameworks and Libraries

* Express.js: A minimal and flexible Node.js web application framework for creating APIs and web server functionality.
* SQLite: A lightweight, file-based relational database for simple setup and use, ideal for small to medium-sized applications.
* JWT (JSON Web Token): Used for secure transmission of information between parties as a JSON object. It is compact, self-contained, and easy to use for authentication and authorization.
* Bcrypt: A password-hashing function that adds secure hashing and salting to protect passwords.
* Cors: Middleware to enable Cross-Origin Resource Sharing, allowing the frontend to make API requests to the backend.

### Architecture Choices

* MVC (Model-View-Controller) Architecture: Separates the application into three main components: Model, View, and Controller, promoting organized, maintainable, and scalable code.
  + Model: Handles database logic and interactions.
  + View: Contains frontend files such as HTML, CSS, and JavaScript.
  + Controller: Contains logic for processing user requests and updating the view.
* RESTful API Design: Uses HTTP requests to perform CRUD (Create, Read, Update, Delete) operations, providing a uniform interface and stateless interactions, enhancing scalability and flexibility.

### Implementation Details

1. User Registration and Login:
   * Technology: Express.js, JWT, Bcrypt, SQLite.
   * Process: Handles user registration and login, securely storing passwords and generating JWT tokens for authentication.
2. Product Management:
   * Technology: Express.js, SQLite.
   * Process: Manages product data, providing API endpoints for fetching products with optional category filtering.
3. Shopping Cart Functionality:
   * Technology: Express.js, SQLite.
   * Process: Manages shopping cart items, allowing users to add products to their cart, update quantities, and view cart contents.
4. Order Processing:
   * Technology: Express.js, SQLite.
   * Process: Handles order submission, storing order details and associated products in the database.

### Conclusion

The chosen frameworks and architecture decisions provide a robust, scalable, and maintainable backend system. By utilizing reliable tools such as Express.js, SQLite, JWT, and Bcrypt, the system ensures secure authentication, efficient data management, and a smooth user experience. The MVC pattern and RESTful API design promote a structured and organized codebase, facilitating future expansion and maintenance.

# Testing

Testing the frontend of this e-commerce application involves performing a series of actions on all pages of the site to ensure that each component works as intended. Given the limited number of pages and the relative simplicity of the actions, manual testing is deemed sufficient and effective. Here is an expanded testing methodology that includes detailed actions, expected outcomes, and considerations for each page.

### General Testing Strategy

1. **Consistency Checks**: Ensure that the UI elements are consistent across different pages, including navigation links, headers, footers, and overall layout.
2. **Functionality Checks**: Verify that each interactive element (buttons, forms, links) performs the expected action.
3. **Boundary Testing**: Test with edge cases, such as empty inputs and boundary values for numerical inputs.
4. **Error Handling**: Ensure that the application handles errors gracefully and provides meaningful feedback to the user.
5. **Responsive Design**: Test the application on different devices and screen sizes to ensure responsiveness.

### Detailed Testing Plan

#### Home Page / Product Listing

* + **Actions**:
    - Navigate to the home page.
    - Verify that the list of products is displayed.
    - Click on each product category in the sidebar and verify that the products are filtered correctly.
    - Click on the "Add to Cart" button for several products.
  + **Expected Outcomes**:
    - The home page loads with a list of products.
    - Products are correctly filtered based on the selected category.
    - Products are added to the cart, and the cart icon or counter is updated.
  + **Considerations**:
    - Check for proper loading of images and text.

#### Product Details Page

* + **Actions**:
    - Click on a product to navigate to the product details page.
    - Verify that the product details (image, name, description, price) are displayed correctly.
    - Click the "Add to Cart" button.
  + **Expected Outcomes**:
    - The product details page loads with accurate information about the selected product.
    - The product is added to the cart, and the cart icon or counter is updated.

#### Shopping Cart Page

* + **Actions**:
    - Navigate to the cart page.
    - Verify that the products in the cart are displayed with correct details and quantities.
    - Update the quantity of a product.
    - Remove a product from the cart.
    - Click the "Checkout" button to proceed to the order placement page.
  + **Expected Outcomes**:
    - The cart page displays the correct products and quantities.
    - Updating the quantity reflects the correct total price.
    - Removing a product updates the cart correctly.
    - Proceeding to checkout navigates to the order placement page.
  + **Considerations**:
    - Check for accurate calculation of total price.
    - Verify that the cart persists between page reloads.

#### Login Page

* + **Actions**:
    - Navigate to the login page.
    - Enter valid credentials and click the "Login" button.
    - Enter invalid credentials and attempt to login.
  + **Expected Outcomes**:
    - Valid credentials log the user in and redirect to the home page.
    - Invalid credentials display an appropriate error message.
  + **Considerations**:
    - Ensure the login form validates input (for example empty fields).

#### Registration Page

* + **Actions**:
    - Navigate to the registration page.
    - Enter valid details and click the "Register" button.
    - Enter invalid details (e.g., mismatched passwords) and attempt to register.
  + **Expected Outcomes**:
    - Valid details register the user and redirect to the login page.
    - Invalid details display appropriate error messages.
  + **Considerations**:
    - Check for input validation (for example empty fields).
    - Verify that existing email addresses are not accepted.

#### Order Placement Page

* + **Actions**:
    - Navigate to the order placement page from the cart.
    - Enter valid shipping details and place an order.
    - Enter invalid shipping details (for example empty fields) and attempt to place an order.
  + **Expected Outcomes**:
    - Valid details successfully place the order and display a confirmation message.
    - Invalid details display appropriate error messages.
  + **Considerations**:
    - Ensure the order summary reflects the correct items and prices.
    - Verify that the order is recorded in the backend (can be checked in dashboard).