UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Auto Parts Market

Name: Lupou Krisztián-Róbert
Group: 30236

Table of Contents

Deliverable 1

Project Specification

Auto Parts Market is a project for an e-commerce website for auto parts. The goal of this project is to make a platform for an auto parts business for selling auto parts to different customers.

Functional Requirements

Functional requirements help to understand the functions of the system for the client part and the administrator part. The functional requirements for a client are the possibility to create an account and the authentication to that account, to update and change the password for the account, to view the parts, the stock for that part and search for the desired auto part, to put the multiple auto parts in a cart to make an order and to view the list of precedent orders. The functional requirements for an administrator are the possibility to delete an account, to add a new product, to update the information for a product, to remove a product, to view the list of products and users, to view the list of orders from clients and to update the state of each order.

Use Case Model 1

Use Cases Identification

Use-Case: Make an order
Level: User Goal
Primary Actor: Logged Costumer
Main success scenario: An order is sent to the administrator
Extensions:

Use-Case: Add a new product
Level: User Goal
Primary Actor: Administrator
Main success scenario: A new product is added
Extensions:

Use-Case: Account Access
Level: User Goal
Primary Actor: User
Main success scenario: User successfully logged in, registered, and updated his account
Extensions:

UML Use Case Diagrams

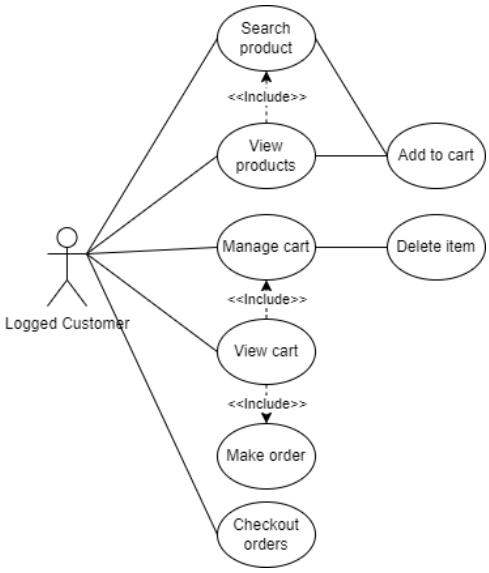

Figure 1. Use-Case: Make an order

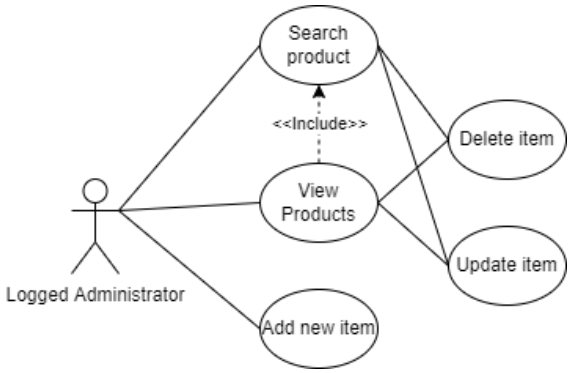

Figure 2. Use-Case: Add a new product

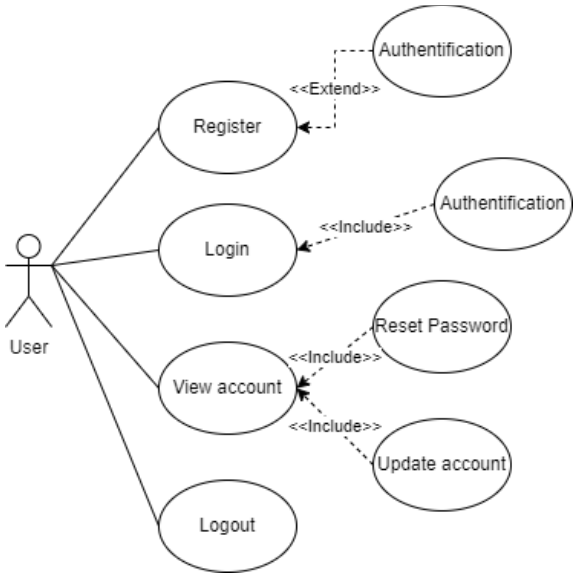

Figure 3. Use-Case: Use-Case: Account Access

Supplementary Specification

Non-functional Requirements

A non-functional requirement is the security of an account. An account is needed to make an order. For each account a password is needed to create and authenticate into account and the administrator will have a separate account. Another requirement is the accessibility of the web application. The application will implement an user interface to easily use the functional features for each user. Another non-functional requirement is the data integrity. All monetary amounts must be accurate to two decimal places and all the stock values must be modified if a product is bought by a costumer. Verifiability is a non-functional requirement which uses tests to prove that the system will function as intended. To prove that the back-end uses tests for the functional requirements for each type of users.

Design Constraints

This project back-end is made using Spring Framework (1) which is written in Java language. For the back-end, the project uses a MySQL database and makes changes on that database. For the interaction with the database, the back-end uses CrudRepository (2) for each entity in the database. In order to not write setters, getters, toString(), equals and constructors, the project use Lombok library. For the testing of the system methods, the application uses JUnit Test Framework (3).

Glossary

(1) The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

(2) CrudRepository is a Spring Data interface for generic CRUD (Create, Read, Update, Delete) operations on a repository of a specific type. It provides several methods out of the box for interacting with a database.

(3) JUnit is a Regression Testing Framework used by developers to implement unit testing in Java and accelerate programming speed and increase the quality of code.

Deliverable 2

Domain Model

[Define the domain model and create the conceptual class diagrams]

Architectural Design

Conceptual Architecture

[Define the system's conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]

Package Design

[Create a package diagram]

Component and Deployment Diagram

[Create the component and deployment diagrams.]

Deliverable 3

Design Model

Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

Data Model

[Create the data model for the system.]

System Testing

[Describe the testing methides and some test cases.]

Future Improvements

[Present some features that apply to the application scope.]

Conclusion

Bibliography