E-Shop System

Author1 (Yad Ghazi Othman, y\_othman@utb.cz)

Author2 (Mohammed Khalid Jamal, m\_jamal@utb.cz)

Author3 (Milad Sulaiman Safari Barzani, m\_barzani@utb.cz)

Author4 (Ahmed Hamid Taha Al-Doori, a\_al\_doori@utb.cz)

Content

[1. Introduction 4](#_Toc120040744)

[1.1. Overview of the proposed systems 4](#_Toc120040745)

[1.1.1. System architecture and design 4](#_Toc120040746)

[1.1.2. Clean architecutre pros 5](#_Toc120040747)

[1.1.3. Clean archirecutre cons 5](#_Toc120040748)

[1.2. System limitations 5](#_Toc120040749)

[1.2.1. Genral problem architecutre 5](#_Toc120040750)

[2. Functional analysis 6](#_Toc120040751)

[2.1. Functional requirements 6](#_Toc120040752)

[2.2. Non-functional requirements 6](#_Toc120040753)

[2.3. Actors 7](#_Toc120040754)

[2.4. Use case model 7](#_Toc120040755)

[2.5. Specification of use cases 7](#_Toc120040756)

[2.6. Implementation of requirements 8](#_Toc120040757)

[3. System architecture 9](#_Toc120040758)

[3.1. Class model 9](#_Toc120040759)

[3.2. Data model 9](#_Toc120040760)

[3.3. Realization of UC 10](#_Toc120040761)

[4. Description of the proposed app 11](#_Toc120040762)

[4.1. Wireframe 11](#_Toc120040763)

[4.2. Application description (if app exists) 11](#_Toc120040764)

# Introduction

## Overview of the proposed systems

* The system will be used as an online shop, the system should not be aimed to one specific category, the system can contain one or more category such as (Electronics, clothes, headsets and more)
* The system can make huge benefits for people that want to start their own business or company, they can use the system as their online E-shop to show their available products, the system should be adaptable to any product categories, the system also can be used in all different fields (Pharmacy, Food and supplements, Electronics, …etc)

### System architecture and design

The system should use the Server-side architecture where the whole system is divided into

* Client side
  + Client side can be any front-end framwork or vanila javascript with HTML, CSS
* Server side
  + While in the server side we will be using the **clean architecture design**, where we have several layers (Presentation layer, Domain layer, Application layer, Infrastructure layer)
  + The main beneift of the clean architecture desing is that the system later can adapt to any client application, in other word By employing Clean Architecture, you can design applications with very low connection and independent of technical implementation details, such as databases and frameworks. That way, the application becomes easy to maintain and flexible to change.
  + The clean architecutre design flow works as nested refrences where the outside can access the inside (meaning we can only go forward, but not backward), for example the domain layer can not access the outside layer, while the outside layers can access the inner layers
* Database side:
  + The system should be adaptable to any database type, since our system in the server side we will use object relational mapper as a layer between the server side and database side

### Clean architecutre pros

Clean architecture has a number of principles that can be summarized here:

* Independent of Frameworks
* Testable
* Independent of UI
* Independent of Database
* Independent of any external agency
* Greate for long live projects
* Parllel teams
* Greate for big teams
* Scalability
* Maintainability

### Clean archirecutre cons

Clean architecutre also has some down sides but they are too little compared to the pros of this architecture:

* Boilerplate code
  + sections of code that are repeated in multiple places with little to no variation.
* Mutiple ways to implement

Diagram

Description automatically generated

## System limitations

### Genral problem architecutre

The only main problem in this general architecutre is the when the application grows the folder architecutre and folder nesting grows hugly and navigation between the folders of the system will become harder, otherwise this architecture with the use of MeditoR and CQRS the application and adopt to many changes with no problem.

# Functional analysis

## Functional requirements

A picture containing diagram

Description automatically generated

## Non-functional requirements

Graphical user interface, text, application

Description automatically generated

## Actors

* Diagram of actors.
* Overview of actors, generalization if needed.
* The actors are user roles, external systems or time.
* Eventual inheritance will also be captured here.
* For actors who do not describe user roles, we choose rectangular notation.

## Use case model

* It contains actors of use cases.
* The SEQUENCE of actions is not captured in the UC model.
* The primary actors will be in the upper left part of the model.
* The use cases included are located to the right of the UC caller.
* Use cases that are extended are located below the caller UC.

## Specification of use cases

* Scenarios will be written in MS Word; they can be inserted into EA.
* In scenarios, the relationships <<include>> and <<extend>> must be visible (where needed).
* Alternative scenario IDs are numbered with a lowercase appendix, i.e. UC001a, UC002b, etc.

Table 1: Template for primary scenario (main flow)

|  |  |  |
| --- | --- | --- |
| Title: | | |
| ID: UC001 | | |
| Characteristics:  Capture a sample use case | | |
| Primary actor:  Actor A | | |
| Secondary actors:  Are not | | |
| Entry conditions:  A condition must be met before the use case can be triggered. | | |
| Output conditions:  A condition must be met after the scenario is executed. | | |
| Main scenario: | | |
| Step | Actor/System | Description |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
|  | | |
| Alternative scenarios:  UC001a – Alternative scenario | | |

Table 2: Template for alternative scenarios

|  |  |  |
| --- | --- | --- |
| Name - Alternative Scenario: | | |
| ID: UC001a | | |
| Characteristics:  The capture of the alternative flow of a use case | | |
| Alternative scenario: | | |
| Step | Actor/System | Description |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
|  | | |

## Implementation of requirements

* An overview of how use cases cover the requirements.
* Diagram from EA for each requirements package.
* Verbal description.

# System architecture

* In the division into modules and subsystems, it is possible to use the component model, deployment diagram, etc.

## Class model

* Capture the static structure of the system.
* They capture the abstraction of objects and their basic characteristics.
* Software classes (difference from the domain model).
* Class model from EA.
* Verbal description – characteristics of the class.
* Class names, types of associations, multiplicity, directionality and other elements must be used.
* Use camel script, class names in the singular, and name associations.
* Describe the responsibilities of classes.
* Describe attributes and their meaning.
* Describe operations and their meaning.

## Data model

* ERD:

Diagram, schematic

Description automatically generated

* Description of entities, data types, meaning. (hama this your taks describe each class and what is the relationshipes means)

## Realization of UC

* Sequence diagrams
* For each use case

Diagram

Description automatically generated

- Here, an actor plays the role of a user who opens our website, browses the products, and adds an item to their shopping cart.Diagram

Description automatically generated

- The user wishes to buy the item and go to the checkout.

If the user is already registered and selects the proper payment option, he receives a payment confirmation and the order is confirmed.Diagram

Description automatically generated

-To receive payment confirmation and have the order completed, the user must enter correct and valid information if the payment method they used was incorrect.

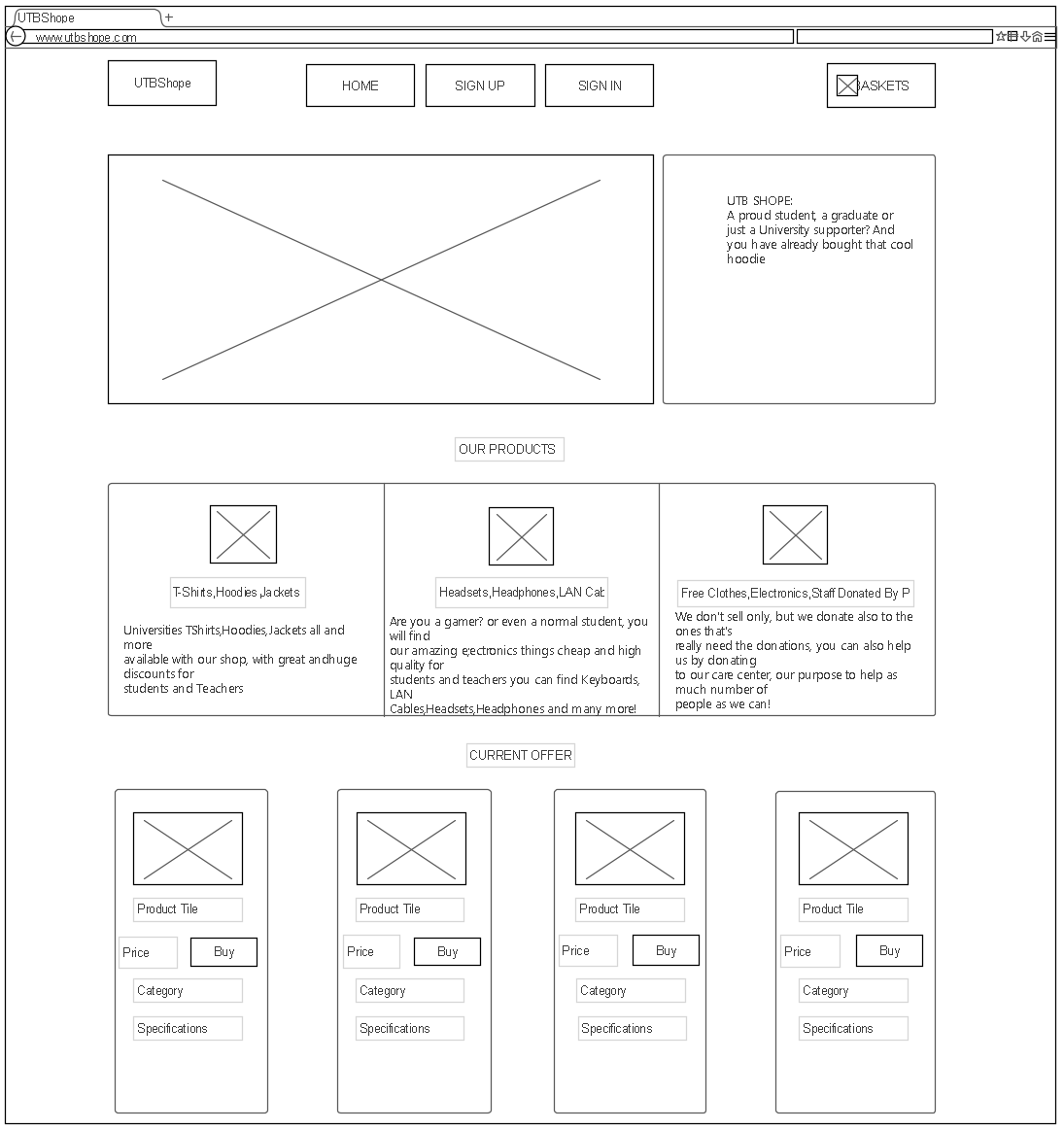
If the user is not already registered, he will not be able to checkout until he does so. Once registered, he will be able to check out using the proper payment option, receive payment confirmation, and confirmation of the order.

# Description of the proposed app

## Wireframe

* A sequence of screens.
* Wireframes for UC (they may not be all).
* It is possible to use the design in EA or another tool (Balsamiq etc.).
* The basic concept of interface.

- Here is a sketch of the system that will be built.

The drawing will be used later by the web developers to determine exactly what the customer had in mind, and the wireframe offers them an idea of the scale of various elements.

## Application description (if app exists)

* A brief description of the application.