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](#_Toc113867785)

[1.1. Overview of the proposed systems 4](#_Toc113867786)

[1.2. System limitations 4](#_Toc113867787)

[2. Functional analysis 5](#_Toc113867788)

[2.1. Functional requirements 5](#_Toc113867789)

[2.2. Non-functional requirements 5](#_Toc113867790)

[2.3. Actors 5](#_Toc113867791)

[2.4. Use case model 5](#_Toc113867792)

[2.5. Specification of use cases 5](#_Toc113867793)

[2.6. Implementation of requirements 6](#_Toc113867794)

[3. System architecture 7](#_Toc113867795)

[3.1. Class model 7](#_Toc113867796)

[3.2. Data model 7](#_Toc113867797)

[3.3. Realization of UC 7](#_Toc113867798)

[4. Description of the proposed app 8](#_Toc113867799)

[4.1. Wireframe 8](#_Toc113867800)

[4.2. Application description (if app exists) 8](#_Toc113867801)

# Introduction

## Overview of the proposed systems

* Describe the purpose of the system, what the system will do, and what it will not do.
* Describe the benefits and meaning of the solution.
* It is possible to add a context diagram incorporating the system (in any form). What are the surroundings of the system?
* The aim is to define the scope of the system.

## System limitations

* All limitations (both technical and from the problem domain). Include everything that can affect design, implementation, and deployment.

# Functional analysis

## Functional requirements

* Organize requests into packages.
* Specify package characteristics.
* Diagram of packages.
* Requirements diagram, aggregate relationships can be used only where it matters.

## Non-functional requirements

* These are limitations that affect functional requirements. Typically, the properties of the systems – how fast they should work, efficiency, simplicity, regulatory requirements, standards, etc. Furthermore, extensibility and scalability.
* System non-functional requirements – technology and standards, operating system, commercial applications necessary for running, hardware, etc.
* Usability – How easy the user can use the system, what experiences are required
* Performance requirements include speed (time to complete the system operation) and safety (relation to possible damage to equipment, people, etc.) during use.
* Accuracy (quantification achieved by processing accuracy).
* Availability – the time between failures, recovery time, etc.
* Regulations and regulations – definition of what applies to the proposed system, what legislation restricts it, etc.
* Interface requirements – what is needed for the operation of the system.

## 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
* Logical data model
* Description of entities, data types, meaning.

## Realization of UC

* Sequence diagrams
* For each use case

# 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.

## Application description (if app exists)

* A brief description of the application.