*Faculty of Natural Science*

*Department of Computer Science*

*Kristianstad University*

*Course: DA122B*

*Program: TGDU1*

*Authors: Joel Cohen, Jonny Lay, Isak Zayane, Emma Johnsson*

The Store

Table of Contents

[1. Introduction 3](#_Toc478369144)

[2. Requirements 4](#_Toc478369145)

[2.1 Use Case Overview 4](#_Toc478369146)

[2.1.1 <Name of Use Case> 4](#_Toc478369147)

[2.1.2 < Name of Use Case > 4](#_Toc478369148)

[3. Design and Implementation 5](#_Toc478369149)

[3.1 Classes 5](#_Toc478369150)

[3.1.1 <Name of Class> 5](#_Toc478369151)

[3.1.2 <Name of Class> 5](#_Toc478369152)

[3.2 Class Interactions and Use Case mappings 5](#_Toc478369153)

[3.2.1 <Name of Use Case> 5](#_Toc478369154)

[3.3 Database 5](#_Toc478369155)

[4. Test Results 6](#_Toc478369156)

[5. Summary and Conclusion 7](#_Toc478369157)

[5.1 Weekly Progress 7](#_Toc478369158)

[5.1.1 Week 1 7](#_Toc478369159)

[5.1.2 Week 2 7](#_Toc478369160)

[5.1.3 Week 3 7](#_Toc478369161)

[5.1.4 Week 4 7](#_Toc478369162)

[5.1.5 Week 5 7](#_Toc478369163)

[5.1.6 Week 6 7](#_Toc478369164)

[5.1.7 Week 7 7](#_Toc478369165)

[5.2 Difficulties and challenges 7](#_Toc478369166)

[5.2.1 <Name of Challenge/Difficulty 1> 7](#_Toc478369167)

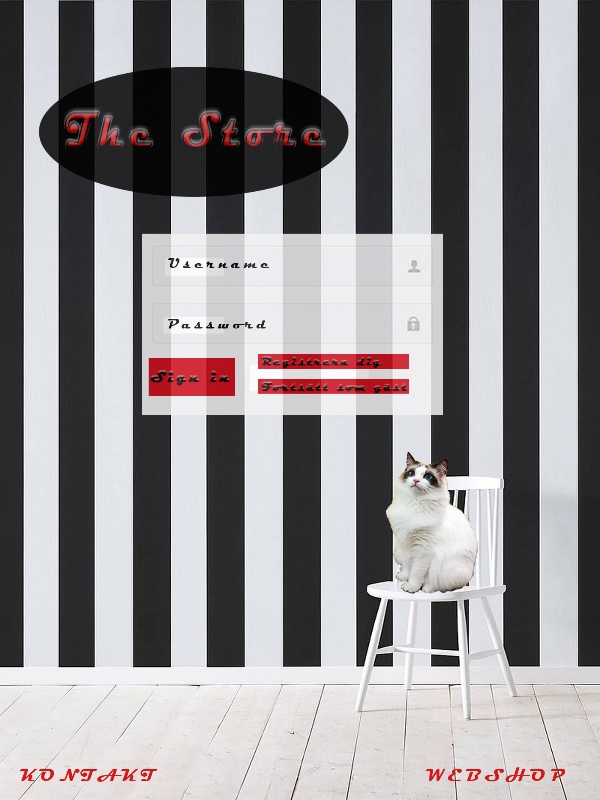
[5.3 Correctness of time estimates 7](#_Toc478369168)

[5.4 Priority decisions 8](#_Toc478369169)

[5.5 Conclusion 8](#_Toc478369170)

[6 References 9](#_Toc478369171)

# Introduction

We have decided to create an application for a webshop called “The Store” which sells upscale cat food and luxury essentials for the discerning cat’s well-being. We feel that this is a fully realistic market that still leaves us a lot of room to have fun with design options.

When entering the webshop, the customer finds the above page offering three alternatives; to log in (for the registered user), to register a new account or to continue to browse the webshop as guest. If the customer decides to continue as guest, they can browse the webshop and put items in the cart but may not proceed to checkout without first creating an account.

Any customer placing an order receives an e-mail with a PDF order specification.

# Requirements

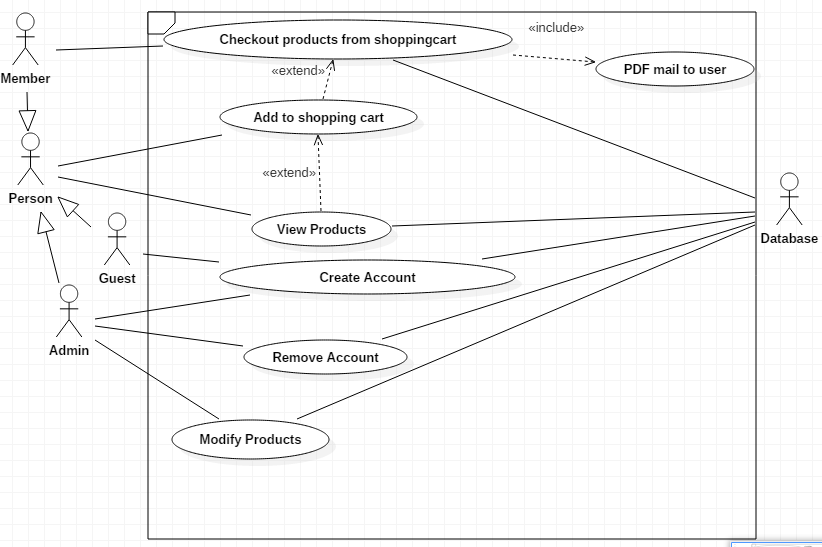
<Instructions: The table below shall contain the requirements for your application. They shall be numbered from 1..N. Each requirement has a short name and a longer description. Hour estimates for tasks and their priority shall be put into the Gantt chart project file. In this case, the priority values are between 0-1000 where 0 is lowest priority, 500 for medium priority (default), 1000 for highest priority. >

Table 1 below lists all the requirements for this project.

Table 1 - Requirements

|  |  |
| --- | --- |
| **Req.**  **No** | **Req. Name: Description** |
| 1 | MySQL: Database over products, customers etc |
| 2 | Main menu: Create the different pages of the webshop and link them |
| 3 | Webshop: Add products, set prices |
| 4 | PDF receipts: Ensure each customer receives a PDF order specification |
| 5 | Accounts: Create possibilities for registering and using an account |
| 6 | Admin account: Create the admin account and ensure usability |

## Use Case Overview



### 2.1.1 <Name of Use Case>

<Instructions: Describe each use case in detail, i.e. the typical user interaction for a use case. Note that this use case section numbering is most likely not mapped to the exact requirement numbers in Table 1, but when feasible, you should refer to the corresponding requirement number(s) from this description text, e.g. *“See requirement numbers 1, 3 and 4 in Table 1.”* >

Actor “person” is an abstract class with the subclasses “user”, “guest” and “admin” (see requirements number 5 and 6) the only interaction that all the actors can do is “View products” and “Add to shopping cart”, which must interact with the Database to see if the product is in stock.

A Guest can also view products and add to checkout but cannot checkout the product since you must be a user or an admin to do so. A guest can create an account to become a user and this will also require connection to the database (see requirements number 1 in table 1).

Just like the actor actors that has inheritance from person, a user can view products and add to checkout. The user can checkout the product from the shopping chart and this will also interact with the database to update the current stock of the item. When the user have checked out her/his products a pdf will be sent to the users e-mail.

Admin can do everything that the other actors can but also modify products and remove user account if there is a connection to the database.

### 2.1.2 < Name of Use Case >

<Instructions: …>

# Design and Implementation

< Instructions: Describe your design in this chapter. >

## Classes

<Description of classes. List one class per sub chapter and add some class diagrams to illustrate relations (inheritance and/or associations) between the main classes. The UML does not need to be extremely detailed, but the most important attributes and methods shall be shown.>

### 3.1.1 <Name of Class>

< Description of this class, including UML. >

### 3.1.2 <Name of Class>

…

## Class Interactions and Use Case mappings

<Instructions: the sub chapters here shall correspond to the use cases in chapter 2, and each use case shall contain a UML **sequence diagram** of the classes that are involved in that use case, and how they interact to implement the use case, including method calls. >

### 3.2.1 <Name of Use Case>

<A sequence diagram of the classes involved in this use case, and how they interact. You may write some explaining text here, and/or you may use notes in the diagram itself.>

## Database

<Show your database design with ER diagram(s). >

# 

# Test Results

Table 2 below contains the current status of implemented and tested requirements.

<Instructions: This table shall map 1-1 to the table in Chapter 2. The test result for each requirement shall be one of the following: NOT IMPLEMENTED, PASSED or FAILED.>

Table 2 - Test Results

|  |  |  |
| --- | --- | --- |
| **Req.**  **No** | **Req. Name** | **Test Result** |
| 1 | <Requirement 1 name> | <NOT IMPLEMENTED/PASSED/FAILED> |
| 2 | <Requirement 2 name> | … |
| .. | .. | .. |

# Summary and Conclusion

This chapter contains a summary and conclusion of the work that was carried out in this project as well as reflections and thoughts about working methods and challenges.

## Weekly Progress

Below is a short summary of what was done each week.

### Week 1

During our initial meeting we quickly agreed that we wanted to make a web shop. We quickly went on to explore git and making sure that the entire team could access the project. After that, the Database was created and linked to the project, a rough mock-up of the web shop was created and we decision was reached to sell fancy cat food and accessories. A Gantt chart has also been produced (see attached file) .

The team was some what challenged by the git process as we overused merge and accidentally worked in master branch when we should have been working in individual branches.

<Instructions: Describe what you did this week. You can see it as a developer’s weekly diary. Try to answer the following questions: What did you do this week? Did you meet any challenges? What was difficult? Did you get stuck with something? What went well and what went bad? What have you learned during this week?>

### 5.1.2 Week 2

TODO

### 5.1.3 Week 3

TODO.

### 5.1.4 Week 4

TODO.

### 5.1.5 Week 5

TODO.

### 5.1.6 Week 6

TODO.

### 5.1.7 Week 7

TODO.

## Difficulties and challenges

Below is a list of notable challenges that came up during this project and that took a long time to solve.

### 5.2.1 <Name of Challenge/Difficulty 1>

<Instructions: List the most difficult tasks in this project and describe why they were difficult. Did you learn something, e.g. how to handle very difficult programming problems?>

## Correctness of time estimates

<Instructions: Look back on your time estimates and discuss your results. How accurate were they? What have you learned about time estimates and how can you get better in next project?>

## Priority decisions

<Instructions: Look back on your feature priority settings. Did you prioritize the right features? Did you succeed to deliver the highest prioritized features? Have you learned anything about setting priorities>

## Conclusion

<Instructions: Look back on the whole project. Here you can write a bit more freely about your thoughts on this project. What was your overall experience? How was the teamwork? What did you learn? Can you list some points that you will do better in next project? Other thoughts. >

# References

<Instructions: In this chapter, you shall list references to external sources, books, web sites, etc. In this document we use the **Vancouver Referencing System**, also called the author-number system. >