|  |
| --- |
|  |
| Xpirit - YAKShop MiniProgramming Assignment  Part of the Xpirit assessment for new hires |
| |  |  |  | | --- | --- | --- | | Xpirit |  | Assignment | |

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | State | Change | Affected Part | Author |
| 1.0 | Final | 26-2-2016 |  | Alex Thissen |
| 1.1 | Final | 10-Jan-2017 | Minor changes to improve readability | Mukul Jain |

**Table of Contents**

[Introduction 2](#_Toc410030461)

[The assignment 2](#_Toc410030462)

[First time starting the application 2](#_Toc410030463)

[Continuing to take practice tests 2](#_Toc410030464)

# Introduction

This assignment is part of the Xpirit assessment procedure for hiring new employees.

You are asked to read the information below. It describes a small problem domain that needs a technical implementation. Take some time to think of an approach and solution for the problem. It is intentionally not too specific in certain parts to allow you to choose your own path and make your own choices. It is important to us to learn how you approach this problem, plan a solution strategy and implement the solution in a small program.

You will actually program the solution together with the assessors. During the programming you can explain what, why and how you are doing the implementation. The assessors will also ask you questions and perhaps ask you to do additional tasks.

The total exercise together with the assessors will take approximately 30 minutes.

The following aspects are considered important for the evaluation of your work and solution to this assignment:

* If you have any questions, do ask these. If there are areas of the assignment that are unclear, discuss these.
* Apply and demonstrate as your current software development skills and approach where you think these are applicable.
* The assignment is a minimum of specifications to help get this part of the assessment started. Feel free to discuss any enhancements to the specification.
* Make sure you include essential parts of your implementation to help demonstrate what you think of a high quality solution to a problem.

# The assignment: The YakShop

Your new customer is a Yak shepherd living on the tundra herding a group of Yaks. Every once in a while he gets customers who come in to buy Yak wool or milk from him. However, he decides to open up a shop on the internet so that he can expand his horizon and actually sell his products outside of his regular clientele. He has decided to hire you as a developer for his new webshop. You've had a few meetings with him and together you've thought up a number of user stories so that you can both have a clear focus.

From research on the internet you know that Yaks age like humans, and with age they give less milk until they finally die of old age. Contrary to humans, a standard Yak year consists of 100 days.

The shepherd currently has Yaks who all stem from the "Yaks" tribe. This tribe is known for its consistency in wool quality, milk taste and production rate of said goods. The shepherd gave you the following facts about LabYaks:

* Each day a LabYak produces 50-D\*0.03 liters of milk (D = age in days).
* At most every 8+D\*0.01 days you can again shave a LabYak (D = age in days).
* A yak can be first shaven when he is 1 year.
* A LabYak dies the day he turns 10.

### Assumptions

* The moment you open up the YakShop webshop will be day 0, and all yaks are eligible to be shaven, as the two of you spent quite a lot of time setting up this shop and the shepherd wasn't able to attend much to his herd.
* Each morning the shepherd milks and shaves his yaks. Yaks which aren't eligible to be shaven on the exact day, cannot be shaved today. Example: a yak who started out on day 0 as 4 years, can be shaven again on day 13.

# User stories

These are the user stories that the Yak shepherd has created together with a business analyst.

## User story YAK-1

As a Yak Shepherd, I want to be able to read in a XML file that contains data about my herd so that I can query it.

**Input herd.xml:**

<herd>

    <labyak name="Betty-1" age="4" sex="f"/>

    <labyak name="Betty-2" age="8" sex="f"/>

    <labyak name="Betty-3" age="9.5" sex="f"/>

</herd>

Note: The age is given in standard Yak years

Your program should take 2 parameters:

1. The XML file to read
2. An integer T, representing the elapsed time in days.   
   Note: T=13 means that day 12 has elapsed, but day 13 has yet to begin

**Output for T = 13:**

In Stock:

  1104.480 liters of milk

  3 skins of wool

Herd:

  Betty-1 4.13 years old

  Betty-2 8.13 years old

  Betty-3 9.63 years old

**Output for T = 14:**

In Stock:

  1188.810 liters of milk

  4 skins of wool

Herd:

  Betty-1 4.14 years old

  Betty-2 8.14 years old

  Betty-3 9.64 years old

## User story YAK-2

As a Yak Shepherd I want to be able to query my herd and my current stock using HTTP REST services which output JSON data. The following are the requests you wish to make.

* GET /yak-shop/stock/T   
  Returns a view of your stock after T days
* GET /yak-shop/herd/T  
  Returns a view of your herd after T days

### Samples

#### **Request 1**

GET /yak-shop/stock/13

#### Response

 { "milk" : 1104.48, "skins" : 3 }

#### **Request 2**

GET /yak-shop/herd/13

#### Response

{

"herd" : [

  { "name" : "Betty-1", "age" : 4.13, "age-last-shaved" : 4.0 },

  { "name" : "Betty-2", "age" : 8.13, "age-last-shaved" : 8.0 },

  { "name" : "Betty-3", "age" : 9.63, "age-last-shaved" : 9.5 }

 ]

}

## User story YAK-3

As a Yak Shepherd I want my customers to be able to buy from my stock using my HTTP REST services. You can assume that requests come in ascending order of time. If you cannot fulfill one of the ordered goods of the order because you're out of stock, you deliver the other goods that are fully in stock.

For instance, if your stock contains 4000 liters of milk and 10 yak hides, and your customer orders 4500 liters of milk and 4 hides, you only deliver the 4 hides (and omit the milk from the result) and give a Http status code 206 (partial content). If the full order is not in stock, you only return a HTTP 404 (Not Found) status code. If the order was placed successfully you return HTTP status code 201 (Created) with the resulting order.

#### Request

* POST /yak-shop/order/T  
  where T is the day the customer orders, this means that day T has \_\_not\_\_ elapsed.

### Samples:

#### **Request 1**

POST /yak-shop/order/14

{

  "customer" : "Medvedev",

  "order" : { "milk" : 1100, "skins" : 3 }

}

#### Response

Status = 201

{ "milk" : 1100.0, "skins" : 3 }

#### **Request 2**

POST /yak-shop/order/14

{

  "customer" : "Medvedev",

  "order" : { "milk" : 1200, "skins" : 3 }

}

#### Response

Status = 206

{ "skins" : 3 }

## User story YAK-4

As a Yak Shepherd I want to have a user interface in my browser which I can use to order goods. The user interface should be able to place an order using the exposed REST services, and provide feedback whether the order was placed successfully, partial successfully, or failed. In case of a partial success it should output what you will get delivered.

## User story YAK-5

As a Yak Shepherd I want to be surprised by your ingenuity so that I can show off to my fellow shepherds. You can do anything that we haven't thought of and show some cool stuff on our monthly shepherd meeting.