ST10114295

Programming 3A

.

requirements and design patterns report

Farm Central

REQUIREMENTS AND DESIGN PATTERNS REPORT

Prepared for

The Accountant

Farm Central

Prepared for

The Head of Marketing

Farm Central

Prepared for

The Store Manager

Farm Central

Prepared by

Nicole Willemse

17th April 2023

Table of Contents

[INTRODUCTION 3](#_Toc138412962)

[NON-FUNCTIONAL REQUIREMENTS 4](#_Toc138412963)

[Important Non-functional requirements and Why 4](#_Toc138412964)

[Addressing the Non-Functional Requirements 5](#_Toc138412965)

[The Impact of The Requirements on the Planning of the Software Development 5](#_Toc138412966)

[DESIGN PATTERNS AND ARCHITECTURE PATTERNS 6](#_Toc138412967)

[The Revelancy of Design and Architecture Patterns 6](#_Toc138412968)

[Applying the design and architecture patterns 7](#_Toc138412969)

[REFERENCES 8](#_Toc138412970)

# INTRODUCTION

Inventory tracking is important, because it can affect the efficiency of the operation of any business. Therefore, we have produced an idea for a protype website that will track incoming and outcoming stock, as well as the tracking of farmers items.

In this proposal, I have outlined the importance of the non-functional requirements that will be implemented into the prototype website, as well as the design and architecture patterns that will be implanted.

# NON-FUNCTIONAL REQUIREMENTS

## Important Non-functional requirements and Why

#### Security

To ensure that the system is secure from unauthorized access, one requirement is **security**. According to (Altexsoft, 2022), security assures that data stays inside the system.

#### Performance

For the system to handle the required number of users, without the system crashing; **performance** is a requirement.

#### Usability

For the system to be easy to use and understand, **usability** is a requirement.

#### Reliability

The system must meet the users’ requirements; therefore, the system must be **reliable.**

#### Maintenance

In order to improve performance and provide a secure environment for the users, **maintainability** is of high importance.

Figure 1: Non-Functional Requirements

## Addressing the Non-Functional Requirements

#### Security

* The user will be required to provide two pieces of information to log into their account (Two-step authorization).
* As requested by the Store Manager, employees will be trained on how to use the system.

#### Performance

The system will handle an unexpected increase in the workload.

#### Usability

A manual will be provided to guide the users through the system.

#### Reliability

To ensure accurate data, as required by the Accountant; data of incoming and outgoing stock, names of employees, and times will be recorded.

#### Maintainability

A recent backup of the site will be available, to prevent data loss and ensure the restoration of the site.

## The Impact of The Requirements on the Planning of the Software Development

The requirements make me want this software to have a 0% failure rate.

The user should be able to learn and use the system with ease.

The system must perform all required functions and be able to accept changes if need be.

According to (Au-Yeung, 2020), accuracy of output is important; so, I plan to ensure that the software outputs the correct results.

Hackers will not be able to access unauthorized resources.

# DESIGN PATTERNS AND ARCHITECTURE PATTERNS

## The Relevancy of Design and Architecture Patterns

* A design pattern, is a general repeatable solution to a commonly occurring problem.
* It speeds up the development process by providing tested, proven development paradigms.
* By using it, you will learn to solve problems using principles of object-oriented design and prevent subtle issues that causes issues.

Four essential elements include **pattern name, problem, solution, and consequences.**

|  |  |  |
| --- | --- | --- |
| **Design Patterns** | | |
| **Creational** | **Behavioural** | **Structural** |
|  | | |
| * **Abstract Factory** | * **Observer** | * **Composite** |
| * **Builder** | * **State** | * **Decorator** |
| * **Singleton** | * **Visitor** | * **Proxy** |

Figure 2: Design Patterns

* Architecture patterns is a universal solution to a commonly occurring problem within a given project.
* It enhances project development lifecycle, improve user experiences and eases software maintenance.

Types include **Layered, Event-driven, Microkernel, Microservices and Client-Server**

In conclusion, yes, they are relevant.

## Applying the design and architecture patterns

The architecture pattern will be applied, using the **n-tier,** **3-tier architecture.**

#### Presentation Layer

The user interface is the top-most level of the application; the main functions is to send content to browsers, translate the results for the user to understand. It will be developed using a web application via Visual Studio.

#### Business Logic Layer

The application gets coordinated and commands are processed. Data gets moved and processed between the two surrounding layers. C# classes and operations will be performed.

#### Database Layer

The business layer gets connected to the database. The farmers and employee names and product details will be stored,

This pattern is beneficial, because the tiers can be developed simultaneously, without impacting one another.

Figure 3: N-Tier Architecture Pattern

# REFERENCES

Altexsoft, 2022. *Non-functional Requirements: Examples, Types, How to Approach.* [Online]   
Available at: https://www.altexsoft.com/blog/non-functional-requirements/#:~:text=Security%20is%20a%20non%2Dfunctional,malware%20attacks%20or%20unauthorized%20access.  
[Accessed 12 April 2023].

Au-Yeung, J., 2020. *What Makes Software Good?.* [Online]   
Available at: https://levelup.gitconnected.com/what-makes-software-good-920ab5732862  
[Accessed 11 April 2023].