Pholisa Nofemele

ST10081751

PROG7311

POE PART 1

Farmer Central

Contents

Introduction

This report aims to show the Farmer Central owner's how the website will be built in order to achieve the desired requirements. Which is to develop a stock management website, the website should be able to track stock that is coming in and going out, and track which farmer an item belongs to.

Non-functional Requirements, Design Patterns, and Architecture Patterns will be discussed. These concepts play a huge role in developing a website that functions properly.

Figure 1: Supplymint, 2022

Non-functional Requirements

Non-functional Requirements represent a description of the ways in which a system works. These requirements include Performance, Scalability, Reliability, Security, and Interoperability (The IIE, 2018).

Performance:

The system will be able to respond to requests within 2 seconds during normal circumstances. Under circumstances of high traffic (which occurs when multiple users are using the website at the same time), the system will not take more than 10 seconds to respond to the user or show results. The ***response time*** should not be too long, the response time is how long it takes for the application to respond to a user's request. In order to ensure that this will be achieved, performance testing and monitoring will be built into the application's delivery chain (Tozzi, 2019).

Scalability:

Scalability is described as the ability of a system to work effectively even when there is an increase in the workload (The IIE, 2018).

Horizontal scaling will be implemented because it is more effective. Horizontal scaling consists of multiple servers. So, if it happens that one server does not work, then another server will be used. Instead of having **vertical scaling** where there is only one server that has a lot space and is very fast but if it crashes there is no way around that.

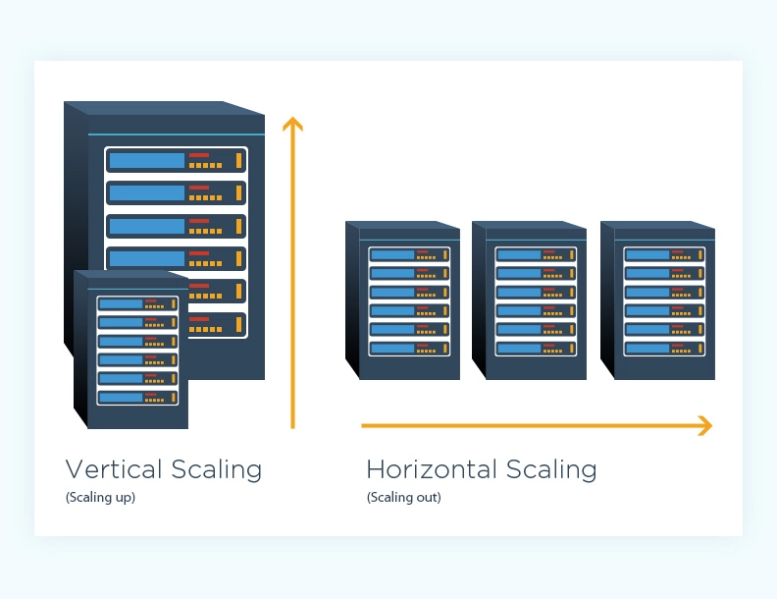

Figure 2: CloudZero, 2021

Horizontal scaling ensures that there is always a way of moving forward, under circumstances in which one server does not work. This will benefit the company very much.

The system will not crash during high traffic, this will be done by ensuring that the website can accommodate high numbers of traffic.

Reliability:

The web application will consist of a maximized ***uptime***. Uptime is the time in which a computer operates. In order to achieve maximized uptime, workloads will be distributed across different servers. An automated failover will be used to locate a workload to a new infrastructure if it fails in one location (Tozzi, 2019).

The uptime will be maximized according to how much uptime is needed or rather sufficient for the website.

The infrastructure will be rightsized, this will be achieved by choosing the right types of cloud instances. Choosing the right size infrastructure will guarantee that the website will have enough resources available, in order for the system to perform adequately.

Maximising the uptime may be costly and can be a negative impact to the plan, but **Cost control** will be taken note of, to ensure that there is no overspending on things that do not add value to the organization/system.

Usability:

Usability will be considered for the workers. The application will allow the workers to navigate documents, architecture information, and the access-control system easily. It is very important to note that making the website too complex is not ideal, what is more ideal is making sure that the system is not too complex for the workers to support it effectively (Tozzi, 2019).

User-experience testing will be done on the workers, to ensure that using the system will not be complex. The knowledge and experience that the workers have when it pertains to computer literacy will be identified by the workers by **testing** the website.

There will be a training program that will train the workers on how things will work, this will ensure that the workers will be on the same page and sort of have an idea of the protocols.

Even though everything will be planned out effectively and a lot of effort will be put into the project, there is still a chance of something going wrong. To make sure that not everything is lost during a disaster etc., there will be a ***backup*** in place and there will be a ***recovery plan***. There will be a backup routine.

Backing up the data may not be enough in some cases, but in order to ensure that an extra mile is taken, the team will know how to ***restore data*** into application instances (an instance is an occurrence of something) that are new.

The system will have a ***schedule*** of how often the data needs to be backed up.

The website will be designed using a website ***pattern*** that is ***modified*** easily, this will assist in ensuring that a website will be ***reliable*** for the long term.

This may impact the plan negatively or positively depending on the workers. Testing can be time-consuming, if the testing is smooth and everything falls into place then there will not be any delays. There might be delays if the workers do not find the website easy to use.

Security is a huge priority. All forms of data should be secured and protected.

This will be done by implementing the following:

- Authentication – this is a process that is responsible for confirming the user's identity.
- Authorisation – where certain people are restricted or granted access to the system or certain materials.
- Encryption – disguising data to prevent access that is not authorized.
- Hash algorithms for passwords –when a password is converted into a form where it is impossible to retrieve the original password.
- Secure Transfer using Secure Sockets Layer etc.- this ensures that there is a secured transfer of data between two parties (The IIE, 2018).

Having a good security system will build a form of loyalty and trust between the business and the customers.

If the security protocols are not implemented effectively then the user's information may be at risk. This will be avoided at all costs by implementing the components mentioned above.

The part of a system that consists of the warehouse data, will be able to communicate, or rather exchange data with other parts of the system. This will be done by using ***interoperability***. This will ensure that the system is constantly being updated, and this will improve the ***accuracy of the data***, due to data being constantly exchanged (The IIE, 2018).

Design Patterns

A **Design Pattern** is description of how a problem will be fixed which can be used repeatedly for many different situations.

As mentioned before, using a pattern that is easily modifiable is very important, the ***Decorator Design Pattern*** will be used to achieve this. Technology is always evolving, and it is very important for businesses to evolve with technology and be able to accommodate or rather provide services that newly introduced, which benefits the business. This design makes sure and tries to make keeping up with new technologies as smooth as possible (The IIE,2018).

In order to ensure that there is constant communication occurring within the system and there is loose coupling, the ***Command Design Pattern*** will be made use of. There is loose coupling when classes that perform different actions are interconnected but are not too dependent on each other. This assists when it comes to updating, ensuring that pieces of code are updated effectively.

Proxy Pattern will be used when payments are made. This pattern is responsible for controlling access to the original object, meaning that an action will only be performed once a request gets through to the original object. This will keep track of transactions and ensure that the transactions only occur once they have been approved (Sourcemaking, 2023).

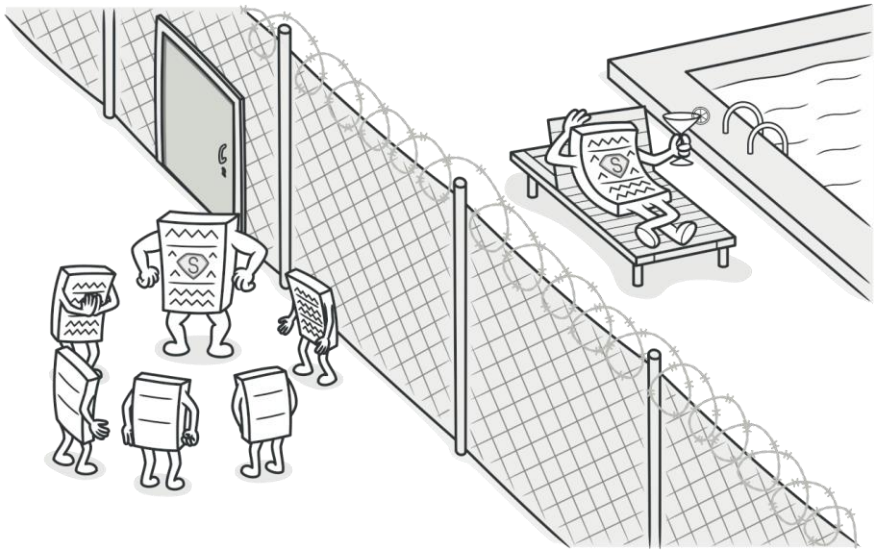

Figure 3:Sourcemaking

Mediator Pattern will be used. This pattern is responsible for restricting direct communication between objects, the objects are forced to collaborate via the use of an objector mediator. The mediator in this project will be the team director, who will tell the team what to do and when to do it etc. At the end of the day, the result will be successful even though there is no communication between the team members. This pattern helps eliminate confusion and misunderstanding and avoids things being clustered (Sourcemaking, 2023).

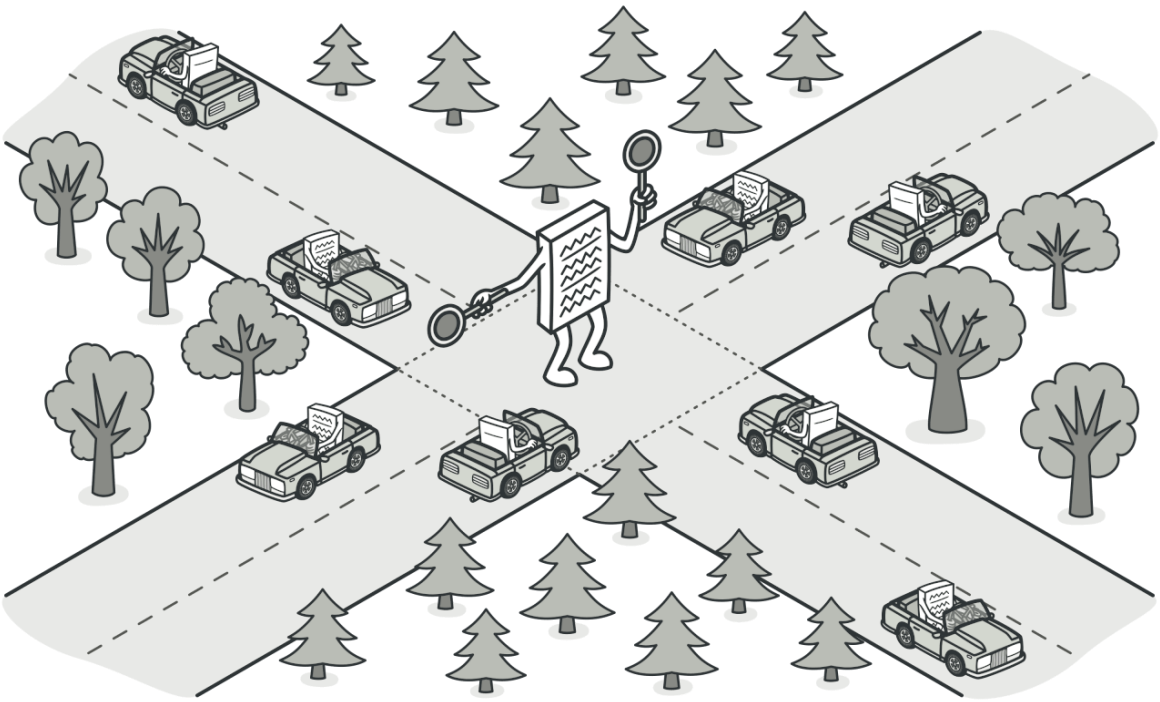

Figure 4:Sourcemaking

Architecture Patterns

Architecture patterns are described as solutions to problems that are common to a wide level application. In simple words, it is a blueprint of building a software (GeeksforGeeks, 2021).

Layered Pattern is also known as the ***'N-tier architecture'***. In this pattern, components are divided into subtasks that play a specific role on the website and these layers are arranged vertically, one above the other.

This layer benefits the concepts mentioned below:

- Maintainability
- Scalability
- Flexibility
- Security

You may be familiar with these concepts because they are part of ***the non-functional requirements*** that were mentioned earlier in this report.

A good example of a company that uses this architectural pattern is Amazon.

This pattern has 4 layers, namely:

- Presentation layer- this is where users see and enter data required into the application.
- Business layer- here the services are provided by the system per user request.
- Application layer- is the medium of communication between the business and data layer.
- Data layer- consists of the database, where data is being managed.

The application layer plays the same role as the mediator pattern.

This architecture pattern is relevant because it benefits the non-functional requirements that were mentioned above.

Microservices Architecture In this pattern everything happens on the server. Applications that are involved with this architecture pattern, handle requests when a business logic is executed, the database is accessed, and messages are exchanged with other systems. A response is also returned.

This architecture pattern is relevant because it consists of the basic structure that is required for the website. It meets the minimum requirements of the website (GeeksforGeeks, 2021).

Conclusion

In conclusion, the concepts discussed will produce data that is accurate, ensure that the website works fast and adequately, and workers will find it easy to make use of the website.

Reference List:

- CloudZero. 2021. They Compare. [Online]. Available at: https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling [Accessed 17 April 2023].
- Design Patterns. [Online]. Available at: https://sourcemaking.com/design_patterns [Accessed 15 April 2023].
- GeeksforGeeks. 2021. Types of software architecture patterns, 27 October 2021. [Online]. Available at: https://www.geeksforgeeks.org/types-of-software-architecture-patterns/ [Accessed 10 April 2023].
- Supplymint. 2022. 5 common inventory management techniques. [Online]. Available at: https://www.supplymint.com/blogs/uncategorized/5-common-inventory-management-techniques/ [Accessed 16 April 2020].
- The IIE. 2018. Programming 3A [PROG7311 Module Manual]. The Independent Institute of Education: Unpublished.
- Tozzi, C. 2019. What it really takes to build a reliable app, DevOps.com, 14 November 2019. [Online]. Available at: https://devops.com/what-it-really-takes-to-build-a-reliable-app/ [Accessed 15 April 2023].