



Contents

[Introduction](#_gjdgxs)

[Purpose](#_30j0zll)

[Project Scope](#_1fob9te)

[Intended Audience](#_3znysh7)

[Design Considerations](#_6tnd17msaolv)

[Assumptions and Dependencies](#_tyjcwt)

[Constraints](#_3dy6vkm)

[Goals](#_1t3h5sf)

[Operating Environment](#_4d34og8)

[Overall Description](#_2s8eyo1)

[Detailed Product Description](#_17dp8vu)

[Architectural Strategies](#_3rdcrjn)

[The Data-Centred Architecture](#_26in1rg)

[The Object-Oriented Architecture](#_lnxbz9)

[The Layered Architecture](#_35nkun2)

[System Architecture](#_1ksv4uv)

[Requirements](#_44sinio)

[Functional Requirements](#_2jxsxqh)

[Non-functional Requirements](#_z337ya)

[Glossary](#_wyrtsun4vhqo)

[Bibliography](#_1y810tw)

# Introduction

## Purpose

This document will serve to describe the features and functionality of the “claims” application forms used by the Namibian University of Science and Technology’s (NUST) Human Resources’ department to approve or reject the logging of overtime hours, application for part time employment and lastly, the financial advice document offered to staff members which is used to ensure that all necessary benefits and payouts are handled correctly. Additionally, this document will outline the mechanisms which will be used to achieve these features and functionality, it will include a development timeline, as well as design specifications.

## Project Scope

The Human Resources department at the Namibia University of Science and Technology (NUST) would like to digitize their current paper based systems. The system will be linked to the universities’ kiosk and will be linked to the employee database in order to streamline the application process. When an employee accesses an application form, their credentials should be autofilled with their personal information which will be obtained from the employee database. There are three(3) groups and four(4) forms which fall under “claims'':

## Intended Audience

The digitized “claims” application forms are intended for use by staff at the NUST HR department and employees of the university. The HR department will use the system to digitize a currently paper based system and automate administrative duties. The rest of the employeest at the university will use the system to complete certain application forms in a more efficient way.

# Design Considerations

## Assumptions and Dependencies

The system will be deployed to the Universities’ kiosk and will be accessible by any employee. It will automate the application process for logging overtime hours, applying for part time employment and completing financial advice forms.

The project is dependent on the NUST ITS system so it will require access to fully integrate the Claims System with the ITS system and work coherently. The four types of forms would then need to be digitized in order to be added to the system. and finally a database needs to be added to the back-end of the system for storage.

## Constraints

Constraints are the limitations a project can and has gone through. One of the major constraints we have is the limited allocation of time given to complete the requirements document and the project in whole. This will affect the quality of the system as it may not meet the expectations of the clients. Another constraint is the platform to be used. Uncertain of the tool to be used, this gives the developers little to no time or opportunity for them to learn and be adjusted to the platform that they may be given to use.

## Goals

Our aim as a team is to build a system that successfully transfers the current paper-based claims system into a digitised version, which makes it easier and more efficient to use.

## Operating Environment

N/A

# Overall Description

## Detailed Product Description

The proposed system will consist of 4 digitalised claim forms, namely the Overtime Claim form, the Advice Claim form, the Part-time Form for Lecturers/Academic Staff and the Part-time Claim form for Administrative Staff.

The 4 claim forms to be created:

1. Overtime Claim

This form will be used to calculate the amount of time someone works beyond normal working hours and the pay received during this time. Features to be included are as follows;

* Will have built-in calculations
* Automated fill-in of information once the user enters the Personnel Number
* Signatures will be required by the Claimant, Line Manager and then sent to payroll.

1. Part-time Claim form for Lecturers/Academic Staff

This form will be used to calculate the total compensation received by lecturers. This includes not only the budget allocation, but also lectures, tutorials, consultations, and board meetings. Features to be included are as follows;

* Will have built-in calculations
* Automated fill-in of information once the user enters the Personnel Number
* Signatures will be required by the Claimant, Line Manager and then sent to payroll.

1. Part-time Claim form for Administrative Staff

This form will be used to calculate the total compensation received by the admin staff. This includes the amount of hours the particular admin staff claimed. Features to be included are the same as the features for the part time claim form for lecturers.

1. Advice Form

This form will be used when someone is hired where their salary is established, which includes their benefits, tax reductions, bonuses and allowances. This form is also for people who are resigning, where the resignation letter is required, as well as payment of benefit forms. Features to be included are as follows;

* Automated fill-in of information once the user enters the Personnel Number or Advice Number.
* Uploading of documents to be attached to the form

## Architectural Strategies

Architectural strategies up for consideration are as follows:

* + - The Data-Centred architecture, as the system requirements call for a database at the center of the system. This method would be complementary to the system design.
    - Object-Oriented Architecture, as this method is best suited for a system that needs to be scalable, high modularity provides high scalability.
    - Layered Architecture is the final model for consideration as this would provide the most robust and secure system at the risk of the project potentially becoming too large and difficult to properly maintain.

### The Data-Centred Architecture

A data-centred architecture lends itself well to the proposed system as this method employs the use of a robust database being the most integral module of the system. In order to meet the requirements, set by the client, a large database will be needed regardless of which architectural strategy is chosen in the end.

However, potential drawbacks of employing this strategy include having an underdeveloped front-end application as the development of the application being centred around the back-end may result in the front-end being neglected. Additionally, this strategy requires a thorough understanding of the data in the system (Harding, 2018). The development/maintenance team in charge of this system not having an adequate understanding of the data would result in shortcomings on the most integral module in the system.

### The Object-Oriented Architecture

Object-Oriented Architecture (OOA) employs modularity to a system and divides each of the system's functions into reusable and self-sufficient entities known as “objects” (Object Oriented Architecture, n.d.). OOA provides highly understandable, reusable, maintainable, and testable modules to a system at a reduced development time and cost.

However, This paradigm requires an extensive understanding of the modules required in the system as any oversights can result in modules not working well together, hence the system may require a rework before it is even completed, resulting in the project going over budget or taking too long to complete.

### The Layered Architecture

Layered Architecture is “the organization of the project structure into four main categories: presentation, application, domain and infrastructure.” (Ziemoński, 2018). This paradigm is well suited for medium to large, front and back-end systems that have a tight deadline as there is a clear separation between each layer and the development team can plan each stage of development very clearly. The separation of each layer also results in domain logic being localized to a layer, so in the event of a rework, only that layer needs to be changed while the others can be left alone. The concept is also easy to understand in the event that someone unfamiliar with the system has to perform maintenance.

There are substantial drawbacks to a layered architecture. The code is often messy because related objects will be scattered across different layers, meaning that cleaning messy functions can take a long time. If the project grows too much, it will become difficult to organize. Additionally, due to the dependency of the top layers on the lower layers, reworks on the lower-level infrastructure almost always propagate to the higher levels and result in entire system reworks for simple procedures such as code optimization. Finally, due to the often-robust nature of the source code in applications that use this paradigm, it can be difficult to tell what an application is doing simply by looking at the structure of the code. A developer unfamiliar with the system would have to spend a lot of time simply figuring out what the different functions do.

## System Architecture

TBC

# Requirements

NOTE: The current process may change during the requirements verification activities and design activities. Additional business process steps and flows may be added as a result of the requirements verification activities and design activities

## Functional Requirements

1. System shall store employee data.
2. The digitized “claims” application shall return the corresponding First Name, Last Name, eeee, and eeee from the database table, if available, for the user entered.
3. The digitized “claims” application shall not allow a future date to be entered in the Document Date field.
4. Users shall have the ability to select a form from a dropdown list - system shall return the corresponding form from the list table.
5. An HR Restricted Verify queue shall be added to the system.

## Non-functional Requirements

1. Work with the business areas to identify and capture all detailed requirements

2. Review the existing detailed requirements to capture all existing detailed requirements.

3. Capture all recommended changes along with the reasons for implementing or not implementing the recommended changes to the detailed requirements.

4.Technology-the software used to create this system should be easily deployable to the Namibia University of Science and Technology kiosk .

5.Robustness-the system should be able to handle internet connection problems,hardware malfunctions and any other kinds of errors.

6.Scalabilty-the system should be designed in a manner that allows for ease of use on a mobile phone,ensuring the experience of using the system on different devices is good.

7.Integrability- the system must fit in as part of the system of claim application as well.

8.Authentification-the system should be able to verify that users are who they claim to be.

# Glossary

* **Architectural** **strategy** - translates business strategy into objectives for building, enhancing, or replacing business and system capabilities.
* **Automate** - convert (a process or facility) to be operated by largely automatic equipment.
* **Deploy** - includes all of the steps, processes, and activities that are required to make a software system or update.
* **Digitise** - is the process of converting information from a physical format into a digital one.
* **Entity** - an instance which is different from another instance and they are independent of each other.
* **Features** - a distinctive attribute or aspect of something.
* **Functionality** - The range of operations that can be run on a computer or other electronic system.
* **Timeline** - a graphical representation of a period of time, on which important events are marked.
* **Substantial** - concerning the essentials of something.
* **Log file** - file that records either events that occur in an operating system or other software runs, or messages between different users of a communication software.
* **Logging** - is the act of keeping a log.
* **Mechanism** - a natural or established process by which something takes place or is brought about.
* **Model** - abstractions used to represent and communicate what is important.
* **Modularity** - Is the degree to which a system's components may be separated and recombined.
* **Streamline** - the improvement of the efficiency of a certain process within an organization. It can be done by automation.

# Bibliography

* + Harding, C. (2018, March 12). *Data-Centered Architecture - First Thoughts*. LinkedIn. <https://www.linkedin.com/pulse/data-centered-architecture-first-thoughts-chris-harding>
  + *Object Oriented Architecture*. (n.d.). Tutorialride. Retrieved September 9, 2021, from <https://www.tutorialride.com/software-architecture-and-design/object-oriented-architecture.htm>
  + Ziemoński, G. (2018, April 10). *Layered Architecture Is Good*. Dzone.Com. <https://dzone.com/articles/layered-architecture-is-good>