# ABC’s Inventory Management

## Architecture Notebook

## Purpose

The purpose of this document is to describe the architecture philosophy, decisions, constraints, justifications, significant elements, and other overarching aspects of ABC’s Inventory Management System that shape the design and implementation of the system. The main purpose of this document is to propose an architecture for the system to be built which will help fulfil the requirements. This document will firstly describe the goals and philosophy of the architecture. This will be followed by any assumptions and dependencies effecting the architectural decisions. The document will then describe the architecturally significant requirements, decisions, constraints and justifications made. Different architectural mechanisms driving the design and implementation will also be described in this document. Finally, the document will present the key abstractions, architectural framework and architectural views of the system.

## Architectural goals and philosophy

The architecture to build the system will be based on a multi-tiered client server architecture. The main architectural goal of ABC’s Inventory Management System is to provide a minimal graphic inventory management application which satisfies the functional and non-functional requirements for the system. The system needs to be built from scratch since the business is new and has never used a digitized online inventory management system. Moreover, the system should be able to handle at least 1000 requests per hour. The system must also be strong and avoid crashes as much as possible as this might affect the daily business in store and the warehouse. The wait time for users for any action should not be more than 5 seconds. Scanners and printers are essential part of the business hence the system, the system should be able to access these devices to scan barcodes, print documents and so on. Multi-tiered architecture will help achieve all these requirements as it is flexible. The justification for all architectural decisions are described in detail below.

## Assumptions and Dependencies

The following are the assumptions and dependencies that drive the architectural decisions.

* All product and current inventory data will be available during the implementation of the system.
* Store and Warehouse employees have basic knowledge about computers and windows operating system.
* Project team members are familiar with the Unified Process.
* Project team members are familiar with core Java, Java EE, XML, JavaScript and MySQL.
* Computers with enough are available to deploy the system.
* Physical devices like scanner and printers are available.

## Architecturally significant requirements

The following are the architecturally significant requirements of the system:

**Performance**

The system must have a reasonable response time. Lagging of the program should be avoided. Queries should be processed in a reasonable time. Processing time must be less than 3 seconds. During high traffic, the processing time should be 5 seconds at most.

**Security**

The system must have login requirements for the users. The system should only accept new passwords that are at least 8 characters in length and that have digits or special characters in them while. Similarly, the store staff must not have admin functions (high level functionalities like adding new user, delete stock, add stock and so on). The warehouse staff should have more functionalities like adding new user, adding new stock, update stock and so on. The system should logout automatically after 5 minutes of inactivity.

**Availability**

The system should be available 24/7. Since store and warehouse staffs might need to use the system to check stock level, stock availability and status at any time, the system should be always available to deliver. If maintenance needs to be made, a notice must be given to all staff and should be done outside business hours so that the business is not highly affected during system maintenance.

**Reliability**

The system can have at most 100 hours of downtime per year. After breakdowns, the system must be able to restart and continue normal functions.

**Usability**

The system should have a very basic and consistent UI. The system must have a white or gray background while buttons and links should have black font color so that these elements are very clear. The system must be available in English. The system must be displayed full screen when launching the application. However, the system should allow users to change the size.

**Capacity**

The system must be able to handle 1000s of requests per hour. The business still being of small scale, the system will not be swamped with requests from different stores. However, in case of a business growth in the next 3 years, the system should be able to handle at least 1000 requests per hour. Similarly, the system should be able to store data and information of all products, stores and the warehouse.

## Decisions, constraints and justifications

The following are the decisions, constraints, and justification regarding the architectural approaches:

|  |  |
| --- | --- |
| Decisions or Constraints | Justifications |
| Use of minimum UI design and decorations following Nielsen’s 10 usability heuristics | * Users will find it easy to navigate through the application improving the **usability** of the application. * Less time can be spent on beautifying the user interface and more time can be spent on solving the business issues the application is designed to solve. * Minimal display will make improve the application’s **performance**. |
| Using Glass Fish application server | * The glass fish server will run on the business logic layer making it free from the data layer or the user interface layer which improves the **reliability, capacity and availability** of the application |
| Use of multitier architecture | * Three tier application architecture boosts the application’s flexibility since each tier can be managed and maintained independently. * It also improves the application’s availability since each component is separated from each other which allows the system to be available even during some maintenance or if a particular component of the system fails. * Using the layered architecture and following proven security protocol improves the security of the application to the highest of standards. Each layer will be following best practices for **security.** Moreover, the separation of components protects each component even if one gets compromised. * Multitier architecture might affect the **performance** of the system. However, this performance drawback will not be drastic as the system is still small scale which does not need to process massive data. |
| Using MySQL relational database in the Data Layer | * The use of MySQL will allow enough space for the system to store data about the product, stores and the warehouse which increases the capacity of the system. Moreover, the data stored will be independent from other layers therefore even if the interface or business layer is modified, the data will not be affected. |

## Architectural Mechanism

The following are the architectural mechanisms of the system with a brief description:

|  |  |  |
| --- | --- | --- |
| Mechanism | State | Description |
| Security Mechanism | Analysis | All users must not be allowed to make crucial changes to data and the system must be secure from people other than staffs trying to get access to the data. Admin functions like adding new product, deleting product, adding new account and so on is separated from store staff account. Accounts are provided with a username and password. |
| Availability Mechanism | Analysis | System must be available for staff to check stock content in each location, send stock to different location and accept s tock from another location. Classes for each layer must be separated from each other so that if one layer fails, diagnosing the point of failure is easier. Application server should be used for data access and persistence. |
| Persistence Mechanism | Analysis | Product, staff and location data need to be stored in a database. The reading and writing of this data must be done using an application server in the Business Logic Layer. |

## Key Abstractions

The following are the key abstractions of the system:

* Warehouse – This represents the warehouse that distributes the stock to stores.
* Store – This represent the stores where product is sold.
* Product – Items that are sold, requested to be transferred from warehouse and sent through deliveries.
* User – Warehouse and store staff that will be using the system.

## Layers or architectural framework

The following diagram represents the layers of the system:

User Interface Layer

Business Logic Layer

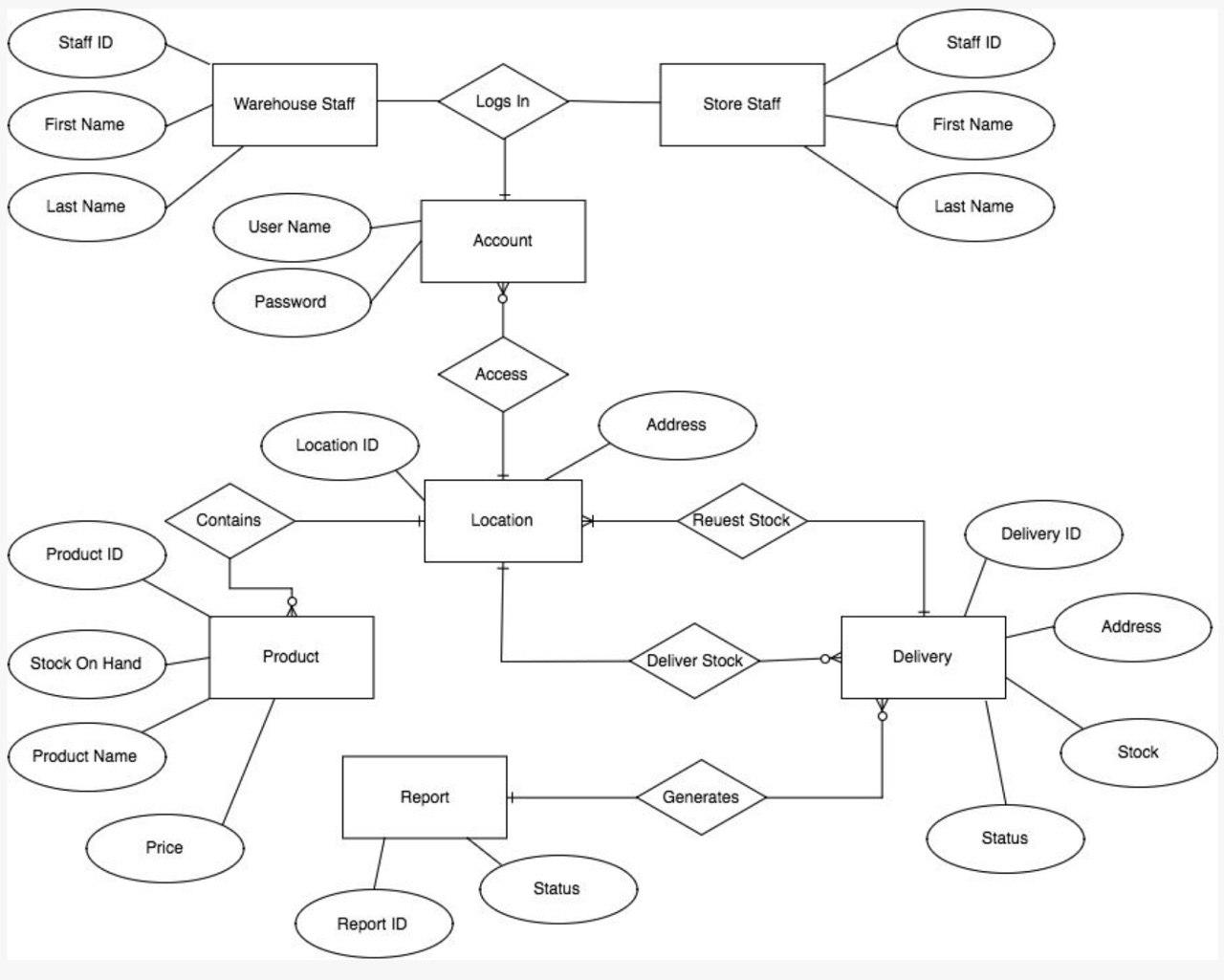
Data Layer

## Architectural Views

The following are the architectural views that will describe the software architecture:

**Logical**

The following is the logical view that describes the software architecture:



**Use Case**

The following is the use case view of the software architecture:

