

**CS 491(A)**

**High level Architecture**

**P04: Dairy Farm Management System**

| **Student ID** | **Name** |
| --- | --- |
| 22100125 | Abdullah Saleem |
| 22100072 | Khawaja Junaid |
| 22100070 | Furqan Athar |
| 22100209 | Saad Qadeer |

**Submitted to**

**Sir Waqar Ahmad**

**Table of Contents**

[1.](#_heading=h.gjdgxs) Introduction 3

[2.](#_heading=h.30j0zll) System Architecture 4

[2.1 Architecture Diagram 4](#_heading=h.1fob9te)

[2.2 Architecture Description 5](#_heading=h.3znysh7)

[2.3 Justification of the Architecture 6](#_heading=h.2et92p0)

[3.](#_heading=h.tyjcwt) Risk Management 7

[3.1 Potential Risks and Mitigation Strategies 7](#_heading=h.3dy6vkm)

[4.](#_heading=h.1t3h5sf) Tools and Technologies 8

[5.](#_heading=h.4d34og8) Hardware Requirements 9

[6.](#_heading=h.2s8eyo1) Who Did What? 10

[7.](#_heading=h.17dp8vu) Review checklist 10

# Introduction

In attempts to modernize the dairy industry in Pakistan, which is one of the greatest producers and consumers of milk and other dairy products in the region, our proposed and developed system will present a solution to better manage the dairy farms and optimize sales and help in better record keeping.

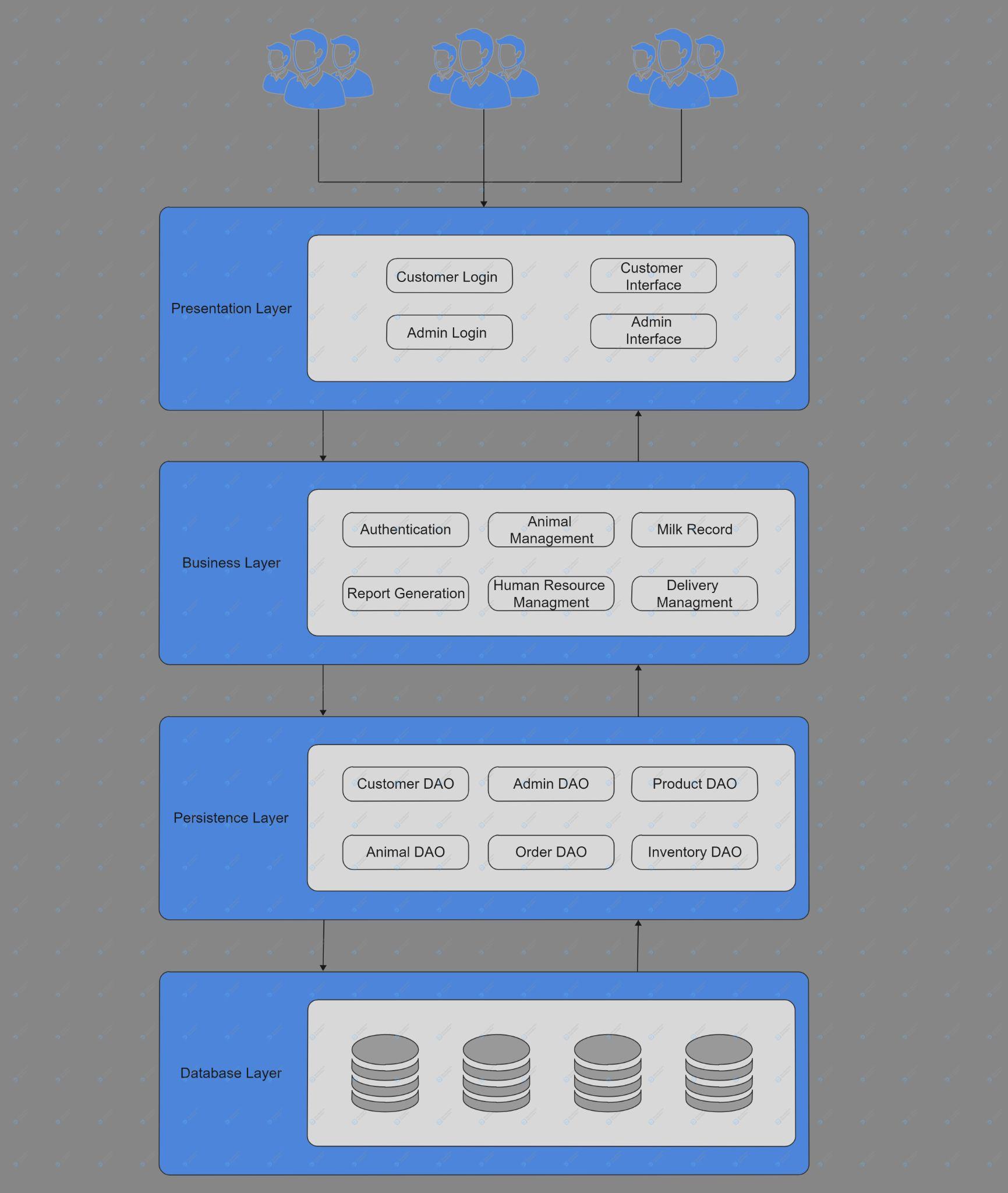
Furthermore, multiple dairy farm owners will have the opportunity to set up their accounts on our management system, as we aim to develop a multi-tenant software, which will aid them in getting more customers and for the better management and record keeping of their farms. The system will allow the customer to keep track of daily milk production, sales, expenses and maintenance. Moreover, the system will also provide an interface for the customers of the dairy farm to check their delivery and monthly invoice.

Potential users of our management system include dairy farm owners (who wish to digitize their records and keep an updated track of their farms) and the customers who wish to order dairy products, be it for domestic or commercial use.

The main purpose of this product will be to provide multiple tenants a way to see various trends in their milk production, expenses and income streams and to make better decisions for the future.

# System Architecture

## Architecture Diagram



## Architecture Description

Our system is based on a layered architecture pattern which includes a presentation layer, business layer, persistence layer and a database layer. As our system is a multi-tenant system hence multiple dairy farms interact with the presentation layer. The multi-tenant system needed to be made in such a way that users have access to only their own data and their data remain safe and secure from other users. That is why a layered architecture was used so that the data is kept safe under multiple layers and all the layers have checks in place to allow only the right users access to their data. Details of different layers is given below:

1. **Presentation Layer:**

Presentation layer is composed of the components that the user directly interacts with which are the user interface components. Our system has four basic user interfaces: Customer Login Portal, Admin Login Portal, Customer’s Interface and Admin’s Interface. Both the login portals have almost the same architecture and design and communicate with the Authentication component in the business layer to allow customers and admins access to the software. Customer’s user interface is user-friendly and has links to access the customer’s monthly/annual reports/invoices. It encapsulates all the customer’s requirements and is an independent component which can be modified and altered on its own without having an affect on other components. The admin user interface envelopes all the administration requirements and lets the managers and the owners of the dairy farm perform their administrative tasks online such as generating monthly and annual reports and updating delivery and animal records.

1. **Business Layer**:

The business layers consist of the logical components of the software which are used to implement the logic of the system. Our business layer includes six basic components which are further divided into sub components. The six basic components are: authentication, animal management, milk records, report generation, human resource management and delivery management. Authentication communicates with the login components in the presentation layers to authenticate users to grant them access to the portal. Animal management, milk records and human resource management components communicate with the admin interface component in the presentation layer and only the admin of a dairy farm can access these features to manage the resources of his/her dairy farm. The remaining two components, reports generation and delivery management, communicate with both, the customer’s interface and the admin’s interface. The admin can generate reports for the entire dairy farm for a year or month or day. On the other hand, the customer can generate reports for deliveries made to that specific customer in a month or year (basically the invoice for that time period). Delivery management interacts with the admin portal who updates the deliveries made to a customer on a daily basis and the customer can view the deliveries however the customer wouldn’t be able to make changes to deliveries.

1. **Persistence Layer:**

Persistence layer consists of the components that encapsulate the methods used to access the data from the database. There are basically six data access objects (DAO) which are used to access the data from the database. Customer DAO is used to access all the data related to the customer, admin for admin data, product for product data (milk in our case), animal for animal data, order for order details and inventory DAO fetches data from the inventory in our database. These data access objects contain methods/functions to fetch the relevant data from the database.

1. **Database Layer:**

Database is the final layer in our architecture which contains all the data of the dairy farm. As our software is a multi-tenant system hence there was a need to secure the data so that data of one dairy farm remains safe from the other users and data theft. That is why the database layer is the last layer in our system so that all the layers above have checks in place to safely store the data and only allow access to the right user to the data.

## Justification of the Architecture

The software that we aim to develop is multi-tenant software which means that multiple clients can use our software for their dairy farms. In order to satisfy our customers' needs, we needed to make sure that the data of our customers is safe and secure. The data should not only be secure from intruders but also from other clients. Our system manages not only the dairy farm but also keeps track of the expenses and the income from the dairy farm, hence there was a dire need to keep the data of the clients safe. In order to achieve this, we decided to build our system on the layered architecture pattern.

Not only this, our choice of architecture helps in the implementation of the non-functional requirements. First of all, it guarantees security and reliability which is the main concern of the users of any software these days. It also achieves performance trademarks. The software is maintainable as it is divided into layers which are further divided into components. If modifications are made to one layer or component, it does not adversely affect other layers or components. This not only ensures maintainability but also helps when upgrades need to be made to the system.

Layered architecture has many advantages such as the guarantee of security, changeability and isolated layers which help in the upgradation of the software but it also has some drawbacks. Scalability is difficult to achieve with layered architecture. This is due to the fact that layered apps tend to take on monolithic properties which means that if we need to scale the app, we’ll need to scale up the entire application. Also, the cost of the software increases as we add more layers which affects the budget of our customers. Despite all these concerns, the main motivation for us to choose this kind of architecture is the security of the data which has become a great concern for millions of the users of the internet.

# Risk Management

## Potential Risks and Mitigation Strategies

| **Sr.** | **Risk Description** | **Mitigation Strategy** |
| --- | --- | --- |
|  | Scope variations, when the project tends to over run the agreed upon time. | Reduction, the scope changes can be reduced by taking in account the changes that are possible along the way with the client this includes agile methodology which gives time for the client to reflect, after small iterations. |
|  | Inadequate Productivity of the development team causing them to fall behind the work schedule and not meeting milestones/targets that were set initially. | One possible way to deal with such situations is to first of all set targets that are realistic and achievable within the given time frame. Another way can be for the development team members to keep check on each other and meet up at fixed periods. Furthermore, group member screening can also be done, to make sure that all the members are almost of the same calibre and to assign tasks based on the skill set of each member. To achieve more organisation a manager role can be assigned to keep check and balance. |
|  | Imprecise estimation,  arises since a timeframe has to be given to the client and other estimations. | reduction, Developers can consider the uncertainty involved in the estimations and certain allocation for estimations such as time frames. Reengineering process so more interaction with the people needed from the client’s organization. |
| 4 | Lack of a clear product vision. This is a potential risk that may occur if the development team and or client are unclear on what they want for the final product. This may also occur if they are not on the same page regarding the final product outcome. | A mitigation technique to deal with this may be to frequently ask questions from the client regarding the final product they want, meet up with the client, show the client the product developed in the early stages and what the development team is planning to do in the coming time. To make the development of the product roll swiftly there needs to be a clear final product vision about what you want to achieve. |
| 5 | Involvement with the end user, making softwares for client’s users(external user) is difficult since they are opposed to change. | Avoidance, test with users using a beta version of the software or have feedback mechanism that considers the opinions of the end user, conduct meetings with the end user to understand the design decisions. |
| 6 | Over simplistic or Over complicated design of the product. This problem may arise after the product has been developed. The client may find the product design to be over simplistic(the product does not offer an adequate number of features) or over complicated(the feature are so much that they overwhelm the user) | . The mitigation strategy to deal with this is to start with a minimum viable product, launch it, get feedback from the user and integrate the received feedback into the product. This can be even done repeatedly to further enhance the product. |
| 7 | Poor quality code, it would entail poor coding practices, released without testing and rushed. | Reduction, Following coding good practices, conduct unit testing for the developed components of the code. There can be a dedicated manager to monitor the code base and along with frequent code reviews. |
| 8 | Inadequate engagement/involvement of the client with the development team which may lead to a gap between what the client actually wants and what the developer team thinks the client wants. | One possible and obvious way of dealing with the situation, which may ultimately lead to what the client might consider a disaster, is to keep the client more involved/ engaged with the development team. The client should be able to clearly put forth their expectations of the product and the development team must be able to effectively and efficiently convey to the client what is achievable in what time span and at what cost. |
| 9 | Lack of adequate resources, developers can leave a project at a point in time, creating problems for the project’s completion. | Avoidance, Proper documentation of the code base can be maintained, or considering having new team members with a training period. |
| 10 | Disagreement between the client stakeholder and the development team. | The mitigation strategy for this risk can be more client involvement and keeping them informed. Both the clientele and development team need to be vocal and expressive about their concerns and ideas. Either of them should not feel that they are left our and that their ideas and opinions are not being taken under consideration. |

# Tools and Technologies

The tools and technologies which will be used in our system are as follows:

* [ReactJS](https://reactjs.org/) version react@16.14.0 for front end development of the webapp.
* Front end tools like [Redux sagas](https://redux-saga.js.org/) redux@4.0.4 for state management and API call management.
* [React-router](https://reactrouter.com/) react-router@5.2.0 a tool to navigate between components.
* [NodeJS](https://nodejs.org/en/) version v14.17.6 for backend deveoplment.
* [MongoDB](https://www.mongodb.com/) a noSQL serverless database MongoDB4.4
* The app would be deployed on [AWS](https://aws.amazon.com/codedeploy/) (2021 - latest version).

# Hardware Requirements

The hardware requirements for the development and deployment machines are as follows:

* **Development Machines** 
  + - 8/16 GB RAM
    - 256/500 GB Hard drive
    - Monitors with minimal blue light exposure are also a crucial part of development machines, since a large amount of time is spent in front of the screen.
    - 4-Core Processor that supports multi threading
* **Deployment Machines**
  + **Server**
    - Minimum 8/16 GB RAM
    - Internet Connection
      * stable
      * good download/upload speed for smooth processing
    - Large Storage Hard disks to handle large amount of information
      * 1 TB
    - Processor
      * 3.2 GHz
      * 8 Core
      * 64 bit
* **User**
  + **The user may use a mobile phone device or a desktop/laptop machine to access the website**
  + 4 Gb Ram
  + A significantly smaller amount of hard disk space can support the software

**Note:** The above stated specification may be subject to change. The Hardware specifications are subject to the amount and intensity of tasks that may be performed on them. Furthermore, specs may deviate from the above stated values and they may also be able to perform their tasks well. These values were stated for smooth workflow.

# Who Did What?

| **Name of the Team Member** | **Tasks done** |
| --- | --- |
| Abdullah Saleem, Furqan Athar | * 2 - System Architecture   + 2.1 Architecture Diagram   + 2.2 Architecture Description   + 2.3 Justification of the Architecture |
| Saad Qadeer, Khawaja Junaid | * 1 - Introduction * 3 - Risk Management   + 3.1 Potential Risk and Mitigation Strategies * 4 - Tools and Technologies * 5 - Hardware Requirements |

# Review checklist

The details of the internal review of the document are as follows:

| **Section** **Title** | **Reviewer Name(s)** |
| --- | --- |
| 1. Introduction | Furqan Athar |
| 2.1 Architecture Diagram | Khawaja Junaid, Saad Qadeer |
| 2.2 Architecture Description | Saad Qadeer |
| 2.3 Justification of Architecture | Khawaja Junaid |
| 3.1 Risk Management | Abdullah Saleem |
| 4. Tools and Technologies | Abdullah Saleem |
| 5. Hardware Requirements | Furqan Athar |