**Architectural Design**

**1 High level components and interfaces**

**1.1 Components**

1. **Cow Provider Component:**

This is one of the key components of the Cow Registration and Login System. This is where the User or Admin selection option is implemented. This also includes the User profile, Admin Profile, Butcher information, Cow information with specification with video and pictures, also there is information about location details.

1. **Admin and User Management Component:**

This is the major sub system that is responsible for the security of the Cow Registration system. It authenticate users and also handles the user profile and order activities such as creating new user accounts, removing accounts from the system etc. Furthermore this component Implements the “control access privilege matrix”.

1. **Cow Component:**

This is the key component that implements the functions related to the cow specification operations of administrator such as adding a new Cow with cow specification, editing cow details and order item or change location or change phone number and removing cows or details change about cow specification or specific cow order etc. Cow component is also responsible for displaying the available cow or animal list.

1. **Chart Component:**

Chart component is a component where the final page arrive. Here user can modify or change his/her phone number or location and add to the payment page and define the payment method. In this page help admin can know where he should delivered that cows or animal. In this case there is some important notification method where admin can send some important information or delivery message through notification.

1. **Public Component:**

This is a relatively small subsystem compared to the other components of the cow Registration System.

1. **Update Component:**

Here we can update user profile, admin profile and location or phone number and details change about cows and animals.

1. **Cow Provider Component**:

**userSelectCowsOrAnimal:** This interface will provide the available cow or animal list for the website. This allows the user to choose the cows or animal, hence a busy interface. This interface is the bridge between the user component and the animal component.

**userProfile:** This interface is used for the user to view and edit personnel information. Authentication and user management component

**authenticateUser:**This is the interface that allows the users to login to the system. This will guide the user to the relevant home page.

1. **Cow or Animal component:**

**newCowOrAnimals:**  This interface is where administrator adds new cows or animal. The cows and admin components are connected.

**editCowsOrAnimal:** In this interface the administrator edit existing cows or animals specification details. The Cows or animals and user components are connected.

1. **Publish component:**

**Getnotice:** This is the interface where the notifications are published. This is the interface where the time table is published. This is connected with the cows or animals component.

1. **Public component:**

**viewFinalChart:** This is the interface which shows the final shopping or order chart adding total money, therefore connected with the cowsOrAnimals component.

1. **Update Component:**

**updateHomeDelivery:** here user can change location and phone number where to delivery.

**updateBUTCCows, updateCUSDOCows, updateOWNDOCCows** this component have update all details about cows or animals.

Dependency and Interaction between Components:

|  |
| --- |
| Cow or Animal |
| -id(string);  -price(int);  -weight(int); |
| -Entry(string);  -OnBid(string);  -Sold(string); |

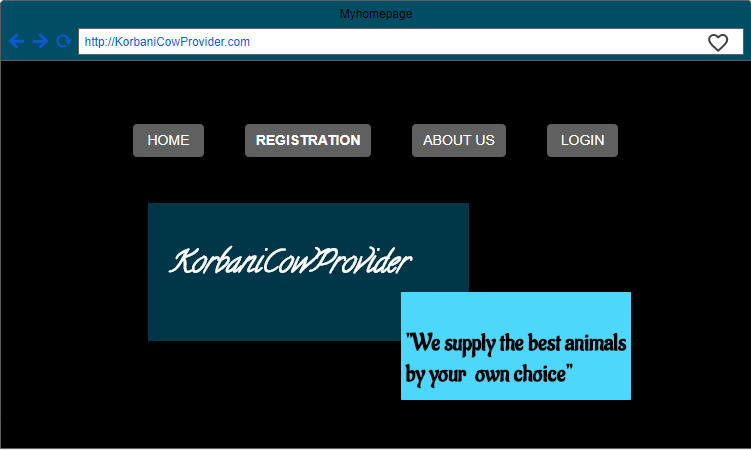
|  |
| --- |
| Admin |
| -id(string); |
| -insert\_animal(string);  -declaring\_sold(string);  -add\_customer(string);  -delivery(string);  -remove\_customer(string);  -remove\_animal(string);  -order\_history(string);  -calculate\_price(int);  -set\_bidfor\_price(string); |

|  |
| --- |
| Customer |
| -user\_name(string);  -password(string); |
| -buy\_animal(string);  -search\_for\_animal(string);  -registation(string);  -order(string);  +payment; |

|  |
| --- |
| Payment |
| -amount (int);  -time(string); |
| -booking(string);  -payondelivery(string);  + Customer;  +Animal;  +Admin; |

|  |
| --- |
| Bid\_for |
| -start\_price(int);  -on\_date(string); |
| -startcall(string);  + Customer ;  +Animal; |

Architecture Structure Sample:

****

**2 Architectural Styles / Patterns**

* The korbaniCowProvider System will be developed under two main architectural styles/ patterns. Development of the project will be done in MVC architectural style and also 3 tier Client/Server Architecture. User can browse the internet and access the korbaniCowProvider System provided within the local area network near Dhaka.

**2.1 MVC Architecture Style:**

* MVC Style separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other.
* The Model component -Manages the system data and associated operations on that data.  The View component- Defines and manages how the data is presented to the user.  The Controller component- Manages user interaction and passes these interactions to the View and the Model.
* We will use this MVC Style for the korbaniCowProvider System because, there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown. In some software systems the code between the process logic and interface are mixed. This will reduce the modularity of application and make the system more difficult to maintain. To avoid this problem we have decided to use MVC architectural style to separate the application logic with the interface. The main advantage of this is style allows the data to change independently of its representation and vice versa.

**2.2 Three-Tier Client Server Architecture:**

* In a client server architecture, the functionality of the system is organized into services, with each service delivered from separate server. Clients are users of these services and access servers to make use of them. We will use this 3- Tier Client Server Architecture because, when data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also be used when the load on a system is a variable.
* **Data Tire:**

The data tire maintains the applications data such as Users’ data , Departments’ data , Cows’ data , Payment’ data , specification tables’ data and the SQL queries . It stores these data in a relational database management system (RDBMS). All the connections with the RDBMS are managed in this tier.

* **Middle Tire:**

The middle tier ( web / application server ) implements the business logic , controller logic and presentation logic to control the interaction between the applications’ clients and data. Business rules enforced by the business logic dictate how clients and cannot access application data and how applications process data.

* **Client Tire:**

The client tire is the applications user interface connecting data entry forms and client side applications. It displays data to the user. User interact directly with the application through user interface. The client tier interacts with the web/ application server to make requests and to retrieve data from the database. It then displays to the user the data retrieved from the server.

* **Example of the 3-tier architecture in the Student Registration System**

* The KorbaniAnimalProvider system is composed of two components one is user and another is admin. Admin can upload every animal picture with details specification and user can sign up with name, username, email address, phone number and the location. User get own profile where he/she can update delete and modify his/her profile. User can chose any kind of animal with any range there is a range for money where use define what range he/she buy an animal. One animal have multiple buyer those case there has an auction who win this auction he/she can buy this animal.
* After buying an animal user have a page where user look for total money after adding home delivery charge and he/she can modify his/her location where to home delivery and how he/she pay for this money (there is multiple option for paying money line case on delivery or visa card or bkash). There is an option for Kasai, we provide different Kasai for different different location phone number, user can directly contact with Kosai without any charge.

**2.3 Different process and their communication**

* In the korbaniCowProvider System, there are number of different processes, such as database server process, web server process, connections between above servers likewise. When sending mails there should run a mail server. HTTP protocol is using to communicate with web servers, SMTP protocol is using to communicate with mail servers. They should communicate each other well to perform the functions of whole application.

* **Arrangement of devices and serves:**

Our University korbaniCowProvider System needs some specific set of servers and devices. Such as: Server to host web applications and web service applications. Database server to store and manage data. Personal computer, note book, smart phone etc… to access the website. Modem/ router/ switch/ hub/ Wi-Fi network/ cable network etc… and also need an Internet Service Provider to have the internet connectivity.

* **Communication among components:**

Above devices are communicating with each other. Personal computer communicates with web server and the database server through HTTP protocol. It communicates with mail server through SMTP protocol. Cable network or Wi-Fi network is also a communication method using in connecting different network components.

3. Design decisions applied whole application:

**3.1 Object oriented software development methods:**

**Reasons:**

* Improved software
* Faster development 
* Lower cost development 
* Improved software development productivity
* Higher quality software

**3.2 Three-Tier Client Server Architecture:**

**Reasons:**

* As more users access the system a three-tier solution is more scalable than the other solutions because you can add as many middle tiers as needed to ensure good performance.
* Security is also the best in the three-tier architecture because the middle layer protects the database tier.

**3.3 MVC Architectural Pattern:**

**Reasons:**

* It should interact with other machines or users effectively.
* For more efficient interaction system should have flexible interfaces. 
* MVC can be taken as for a popular and easy to handle web application development style that has the feature of separating the presentation, Business & intermediate logics. 
* Ease to coding and provide well defined interfaces within each logic