# Chapter 3: Design

# Design:

Design is the plan or decorative pattern that show the functions, objects, methods of the software before it is developed. It provides the initial view of the proposed software such as how it will look, works.

Since I’m using object-oriented programming, this process helps me to divide the tasks into classes, how these classes will interact, what data each will contain and what action each will perform. It will make easier to understand the requirement easier and prevent redundancy. Risk will be very low if we use this process. It projects initial vision of the proposed software.

Types of software design level:

1. **Architectural Design:** It recognized the software as a system with multiple components associated with each other.
2. **High-level Design:** It focuses into how the system components can be implemented in forms of modules.
3. **Detailed Design:** It deals with the implementation of modules and the interactions between them.

# Structural Modelling:

Structural model projects the concept of the system and illustrate the relationship between system’s components. This model is independent of time.

# Final class diagram:

A class diagram is illustration of the architecture of a system which maps out the system’s classes, their attributes, methods and relationship between them. It provides the static view of the system which can be used as a blueprint for the final product.

**Justification of approach:**

* Forward and reverse engineering process.
* Analysis and design of the static view of a system.
* Provide basic notation for other modelling in UML.
* Construction of system using object-oriented programming (OOP)

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| + | This sign represents the class is public and is accessible within the namespace |  |
| - | This sign represents the class is private and is not accessible to other class |  |
| # | This sign represents the class is protected and only accessible within the solution inside a single namespace |  |
| Generalization | Generalization: -  This shows the inheritance from one class to another |  |
| Aggregation | Used to show relationship between class that have relation among their instances |  |
| Composition | Used to show the child class that is fully dependent to parent class |  |
|  | Used to show classes with attribute and function |  |

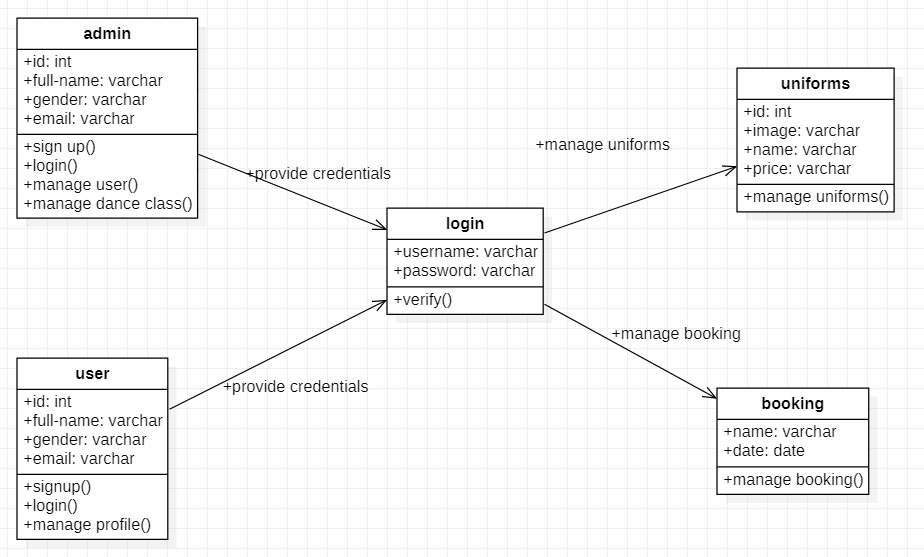


Figure 1:class diagram

# Data flow diagram:

Data flow is used to show the flow of data in the system. DFD describes the process that are involved in a system to transfer data from the input to the file storage and reports generation.

**Justification of the approach:**

* logical information flow of the system
* determination of physical system construction requirements
* simplicity of the notation
* establishment of manual and automated system requirements

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| External entity | Represent human, system or subsystem. It is where data come from. |  |
| Process | It is function or activity where data is manipulated. |  |
| Data store | Data are store here. Simply, database |  |
| Data flow | Show the direction of flow of information. |  |

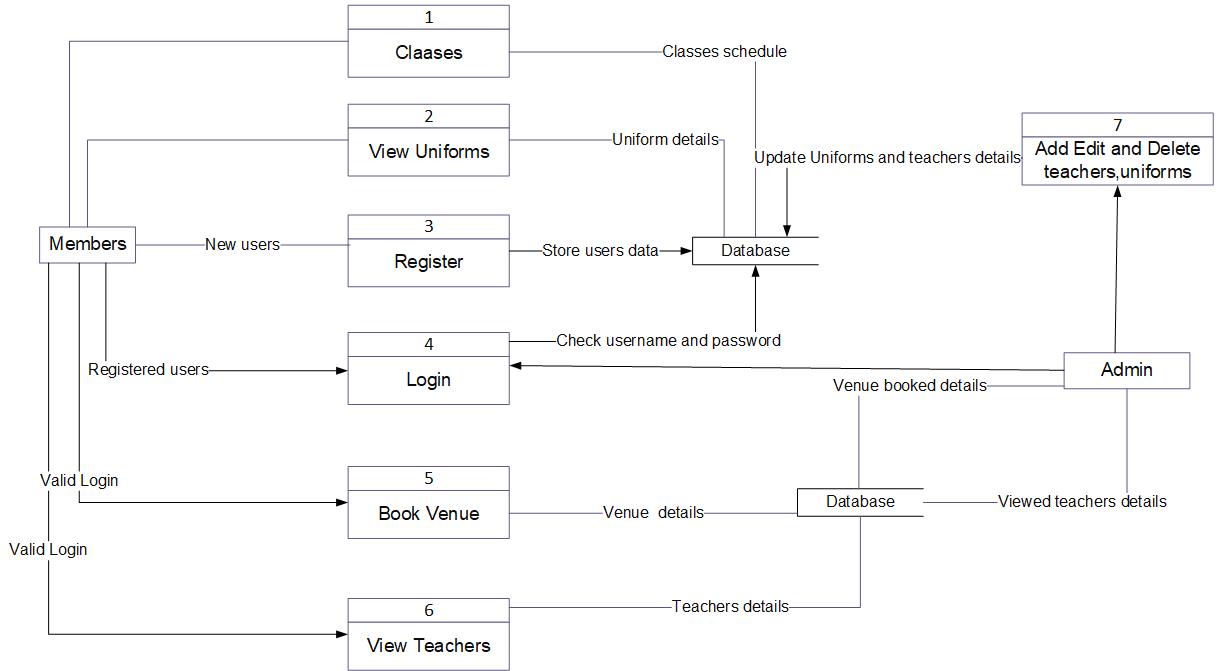


Figure 2:DFD

**DFD explanation:**

1. Admin provide login details
2. Login leads admin to admin panel
3. Admin can add/delete admin user and product
4. Member request to view the facilities and features
5. Product details are retrieved from database and display to customer
6. New users provide their details for registration
7. Details are store in database
8. Members provide details to login
9. System check the details with the database
10. Give access if details are valid otherwise deny
11. Details are checked with the details in the database
12. Give access and open home page if details are correct
13. Return back to login if details are wrong
14. Admin either manipulate customer or product details
15. Admin add/delete either customer or product
16. Products details are stored in database after manipulation

# Behavior Modelling:

It captures the varieties of interaction and instantaneous states within a model as it executes over time; tracking how the system will act in a real-world environment, and observing the effects of an operation or event, including its results.

# Activity diagram:

Activity diagram is graphical representation of flow of activity from one to another. This describe the dynamic aspect of the system. It is essentially an advanced version of flow chart that modeling the flow from one activity to another.

**Justification of approach:**

* Describe the sequence from one activity to another.
* Demonstrate the logic of an algorithm.
* Illustrate the various steps involved in a UML use case

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| Start | This symbol represents the start point of the system |  |
| Action/activity | This symbol represents the action or activity |  |
| Action flow | This symbol represents the flow of action from one to another |  |
| Decision | This shows where decision is made of the action |  |
| Merge |  |  |
| Join node | It joins the multiple activities into single activity |  |
| Fork node | This splits one activity into multiple activities |  |
| Swim lane |  |  |
| End point | This symbol represents the end point of the activity |  |

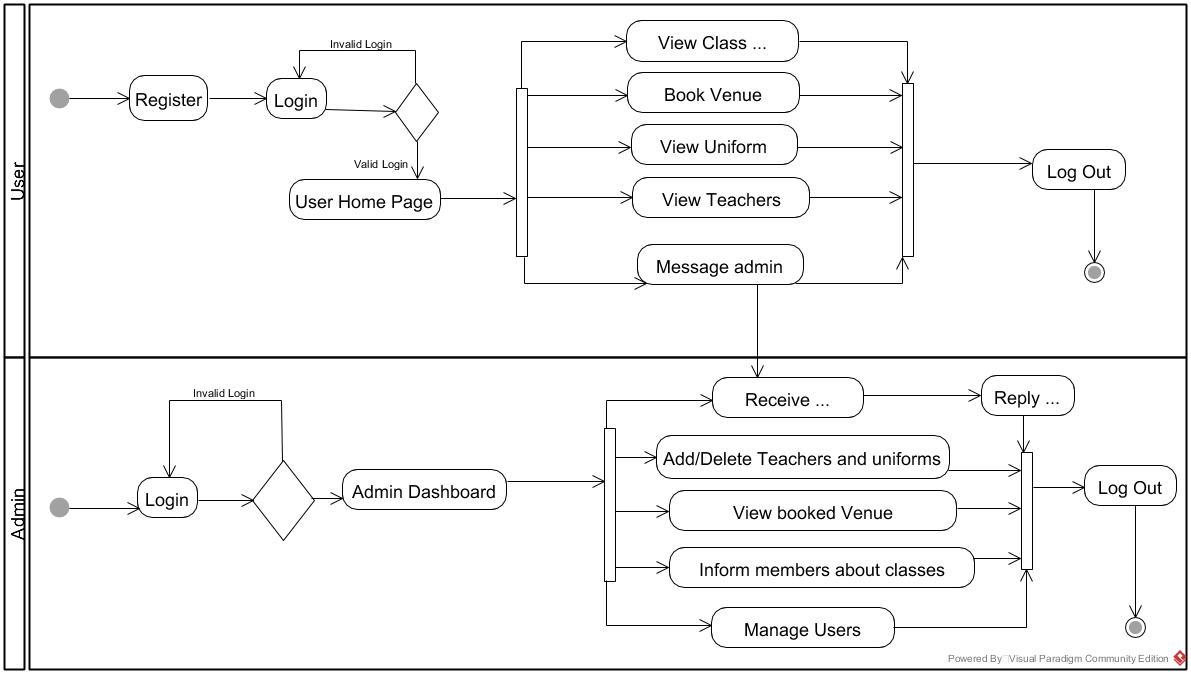


Figure 3:activity diagram for login

**Activity diagram explanation:**

1. Users provide login details to access the system
2. System check the details
3. If details are valid, dashboard is opened otherwise it moves back to login
4. User logout from the system
5. Admin make decision either to add, delete or update item and logout
6. User can make order and, manage classes and manage teachers

# Sequence diagram:

This diagram describes interaction of objects in sequential order. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what message are sent and when.

**Justification of approach:**

* Used to understand the detailed functionality of current or future systems
* Visualize how message and tasks move between objects in the system
* Analyze and visualize the logic behind function, operation or procedure

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| Lifeline | Used to represent the time duration. |  |
| Message | Used to share message between classes. |  |
| Reply message | Response to the message |  |
| Self-message | Process or method arise within the lifeline operation |  |
| Alternate | Used to show the choices |  |

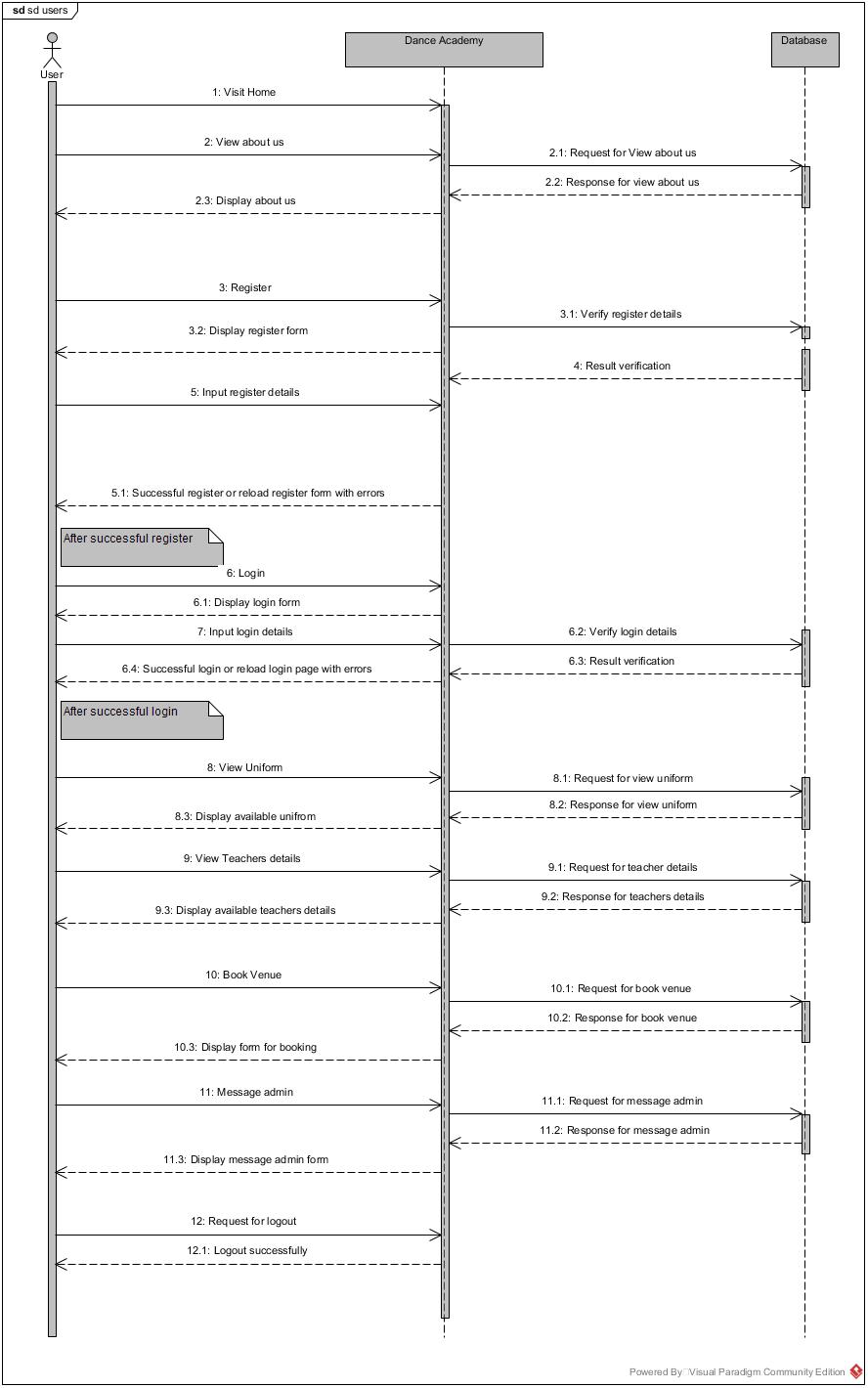


Figure 4:sequence diagram for user

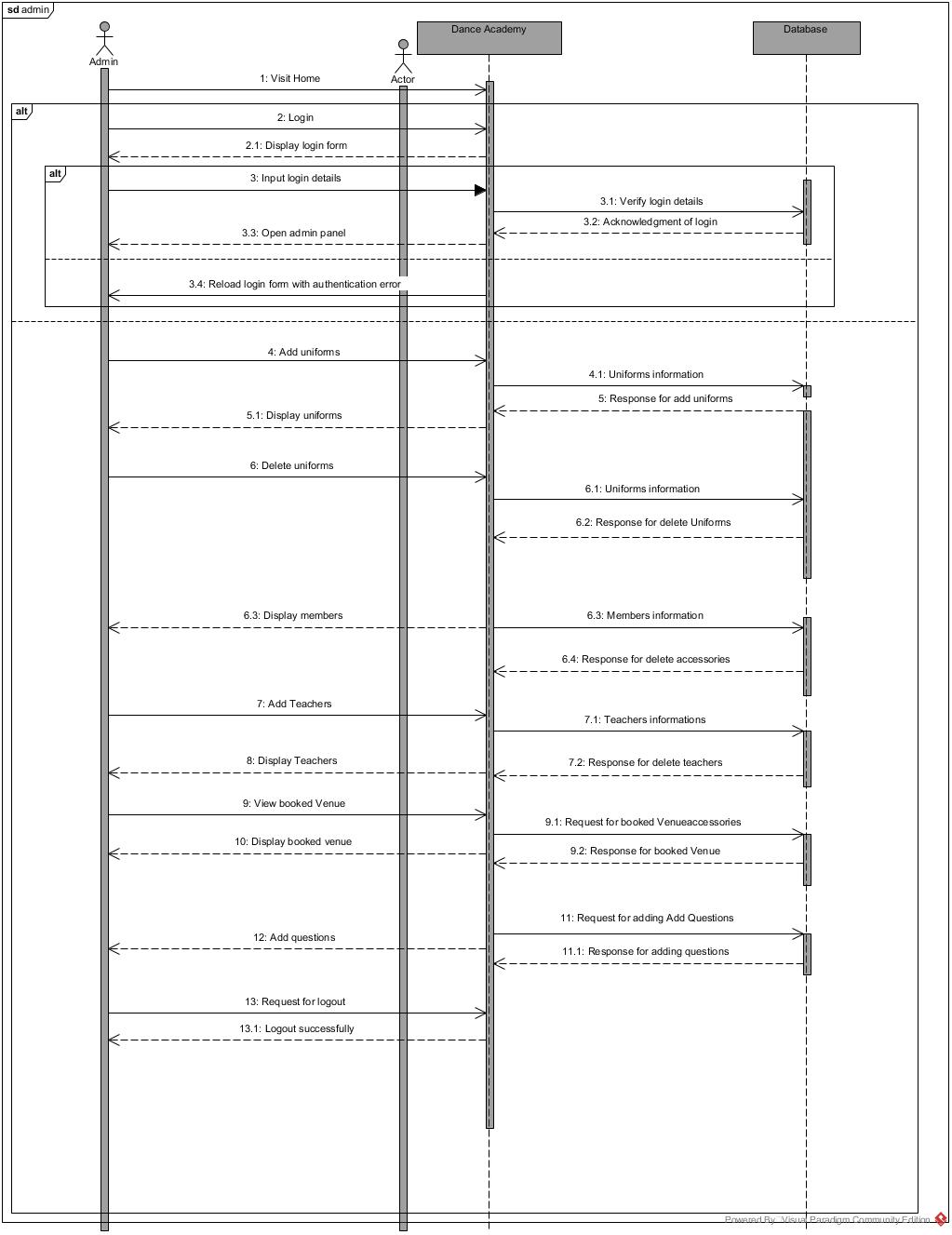


Figure 5:sequence diagram for admin

# Database Modelling:

Database is another important thing in system. Database modelling is the process of creating a database model for the data to be stored in it. Database modelling helps in the visual representation of data and enforces business rules, regulatory compliance.

# Data dictionary:

The table which contain the overall details of data that define the principle of database design and work. It contains database metadata. It describes things such as access to data, where the database is physically located.

**Column User**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Length** | **Nullable** | **Key** | **Constraint** |
| Id | Integer | 20 | - | Primary key | Pk\_id |
| Full\_name | Varchar | 100 | Null | - | - |
| Gender | Varchar | 100 | Null | - | - |
| Email | Varchar | 100 | Null | - | - |
| Phone | Varchar | 100 | Null | - | - |
| Dob | Varchar | 100 | Null | - | - |
| Username | Varchar | 100 | Null | - | - |
| Password | Varchar | 100 | Null | - | - |
| Re-password | Varchar | 100 | Null | - | - |

**Column Admin**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Length** | **Nullable** | **Key** | **Constraint** |
| Id | Integer | 20 | - | Primary key | Pk\_id |
| Image | Varchar | 200 | Null | - | - |
| Name | Varchar | 200 | Null | - | - |
| Gender | Varchar | 200 | Null | - | - |
| Email | Varchar | 200 | Null | - | - |
| Phone | Varchar | 200 | Null | - | - |
| Dance\_style | Varchar | 200 | Null | - | - |
| Description\_of\_dance\_style | Varchar | 200 | Null | - | - |
| Time | Varchar | 200 | Null | - | - |

**Column Uniform**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Length** | **Nullable** | **Key** | **Constraint** |
| Id | Integer | 20 | - | Primary key | Pk\_id |
| Image | Varchar | 100 | Null | - | - |
| Name | Varchar | 100 | Null | - | - |
| Dance\_style | Varchar | 100 | Null | - | - |
| Gender | Varchar | 100 | Null | - | - |
| Price | Varchar | 100 | Null | - | - |

**Column Booking**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Column Name** | **Type** | **Length** | **Nullable** | **Key** | **Constraint** |
| Name | Varchar | 100 | - | Primary key | Pk\_name |
| Venue | Varchar | 100 | Null | - | - |
| Date | Varchar | 100 | Null | - | - |
| Size | Varchar | 100 | Null | - | - |

# ER diagram:

This diagram project the relationship between main entities of the system by mapping out the attributes, methods of the entities. It is used in database designing. ER model is based on the notation of real- world entities and the relationship between them.

It is used due to follow reason:

* Helps to define terms related to ER modeling
* Helps to describe entities, attributes, relationships
* Database designer gains better understanding of the information to be contained in the database.

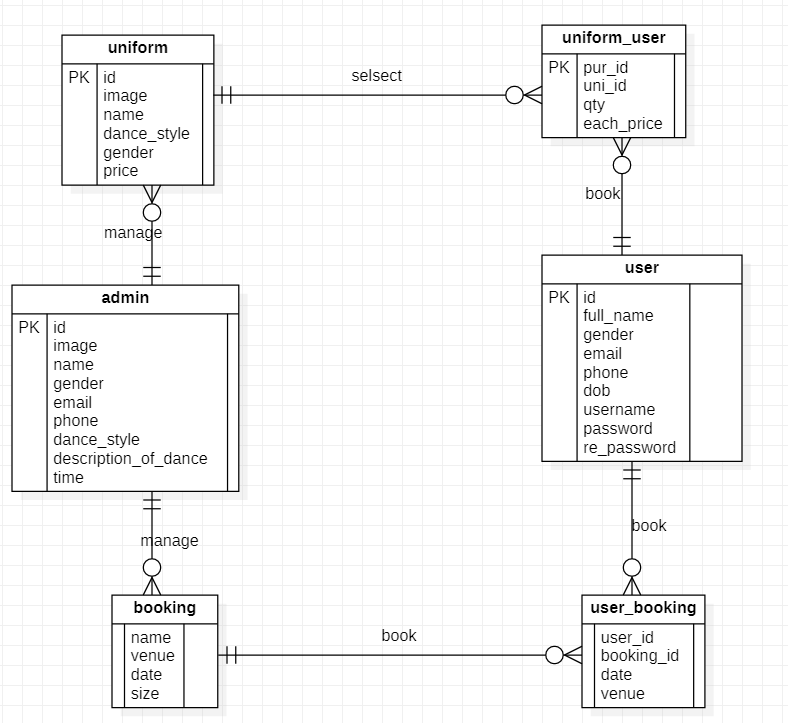


Figure 6:ER diagram

# User Interface Modelling:

This model illustrates the way how human react with the computer or system to perform any task in the system. It focused in looks or style.

Great UI have:

* Common icon and well leveled
* Simple and clear interface

# Prototyping:

This is the representation of how the system will look after its completion. This makes easy in coding.

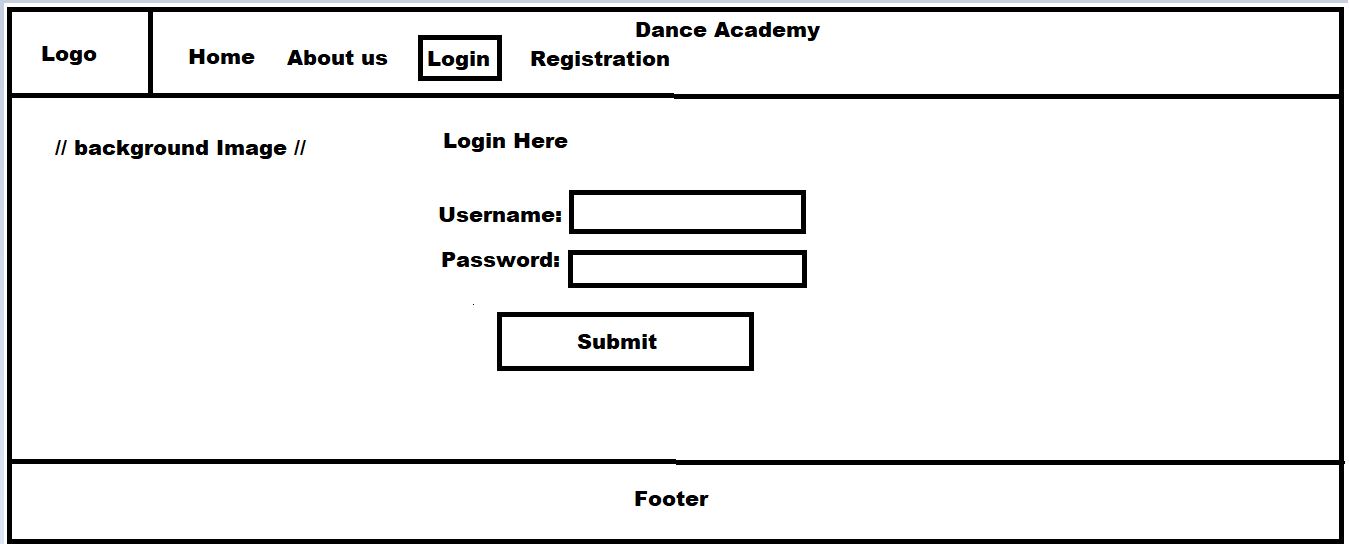
**Login Prototype**

Figure 7:login UI

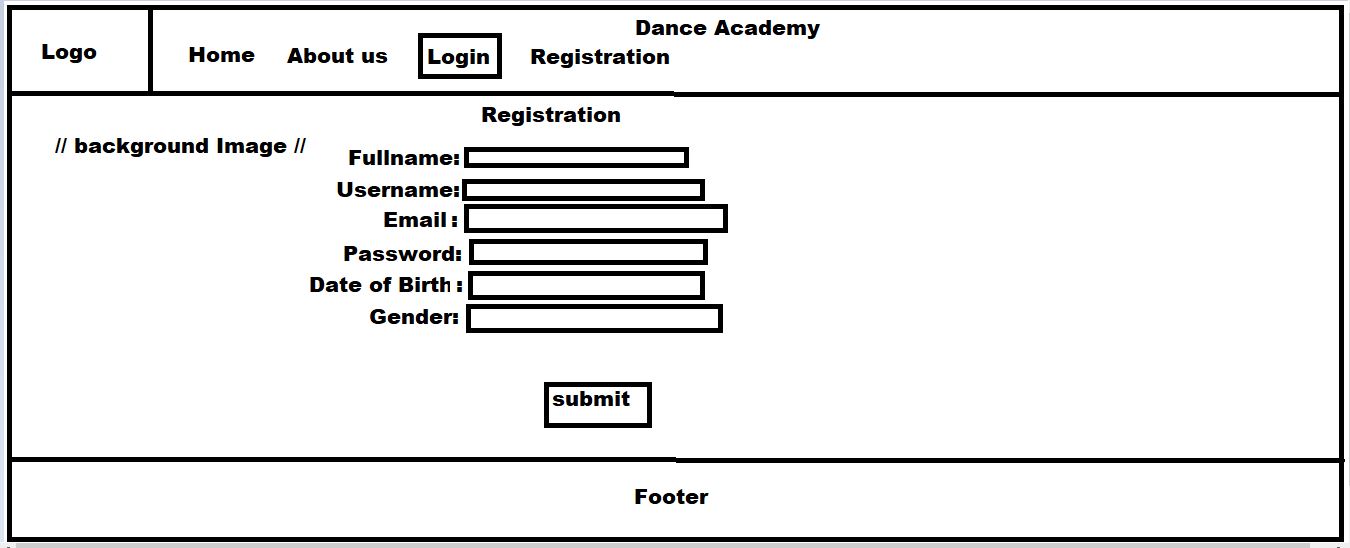
**Registration Prototype**

Figure 8: registration UI

**About us Prototype**

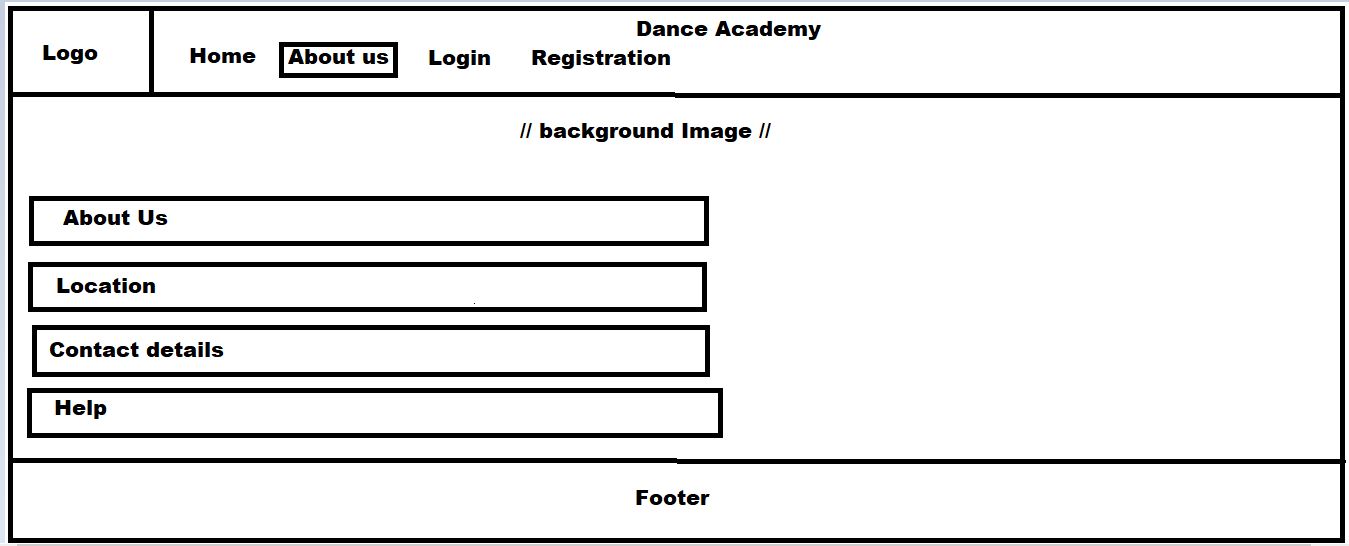


Figure9:About us UI