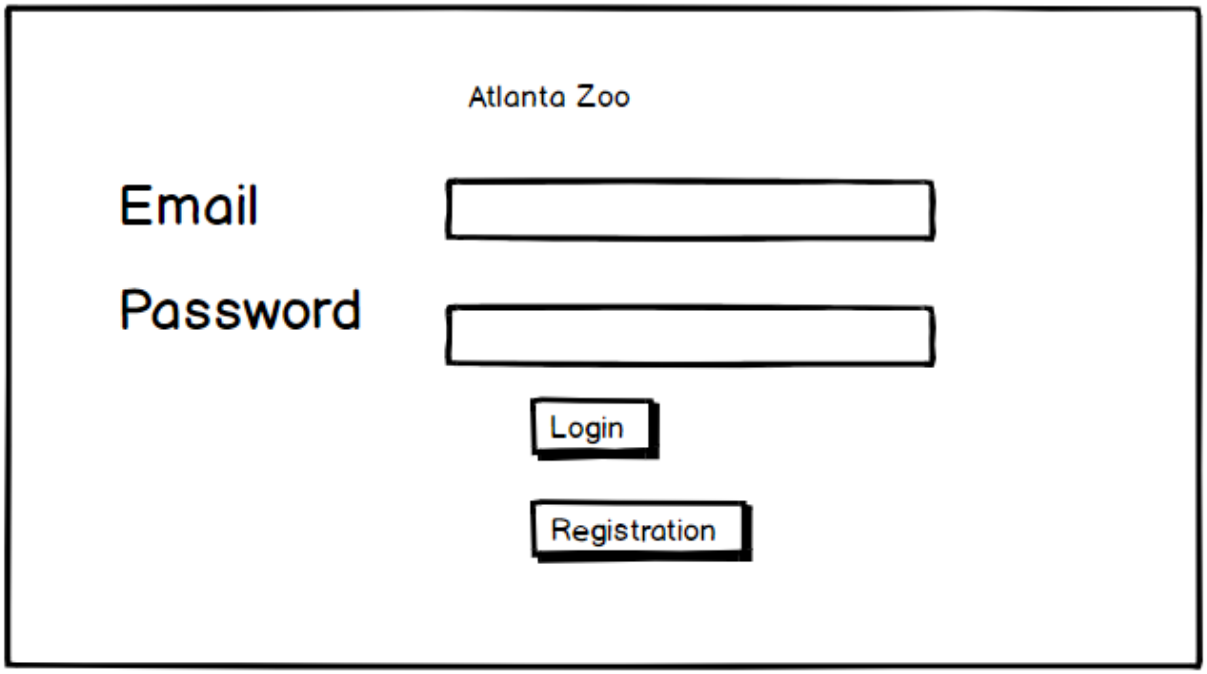
Phase 3

MySQL code

1.Welcome Page

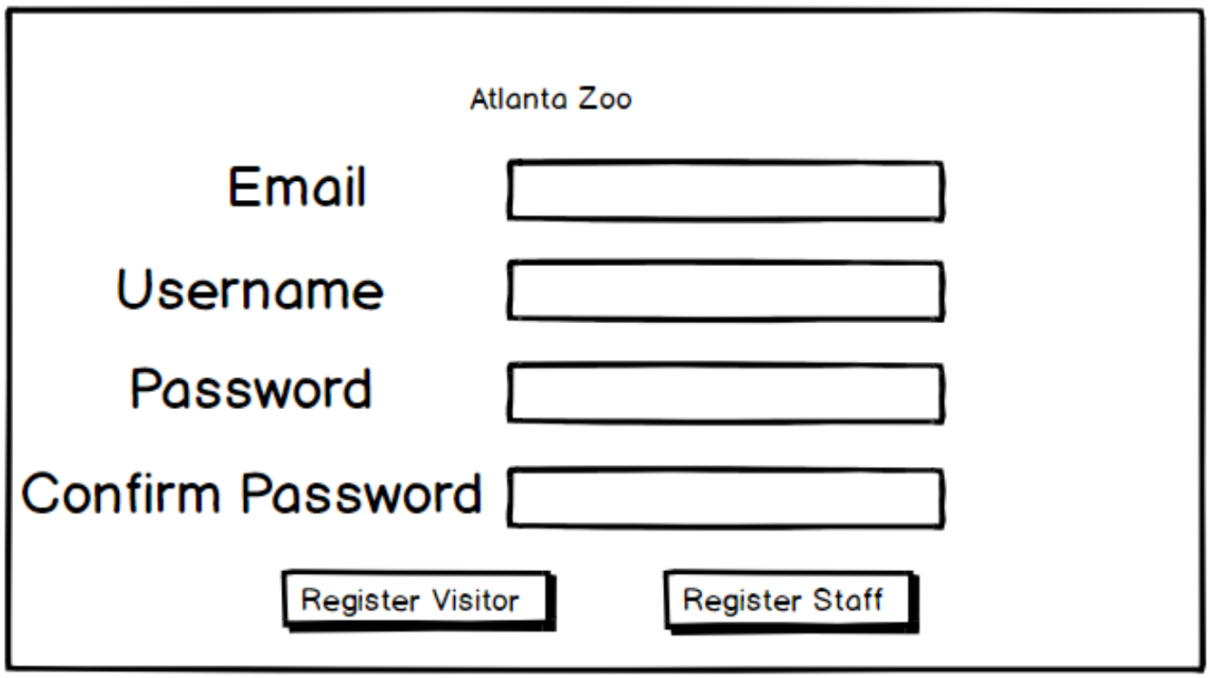
1.1.Log in:




```
//To check if a user who has a email ‘haoliu@gmail.com’ exists in database
SELECT Email
FROM User
Where Email = 'haoliu@gmail.com';
// if Email does not exists, application will ask user to enter another email

//To retrieve the use who has email ‘haoliu@gmail.com’ and password ‘12345678’ in database
SELECT *
From User
WHERE Email = 'haoliu@gmail.com' and Password = '12345678';
// if Email and password combination not match, application will ask user to try again
```

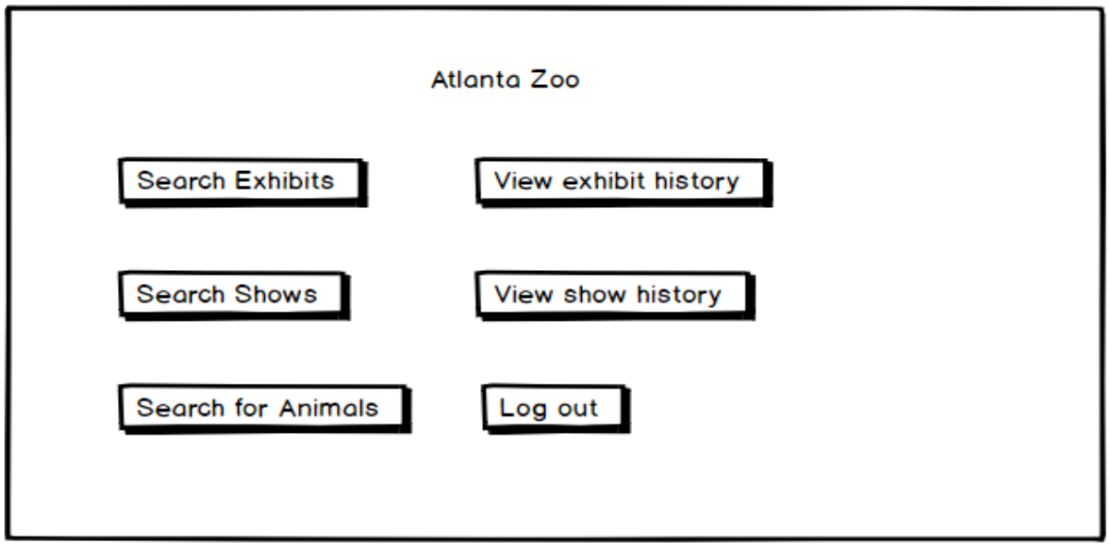
1.2. Registration:

Enter the Interface of Registration



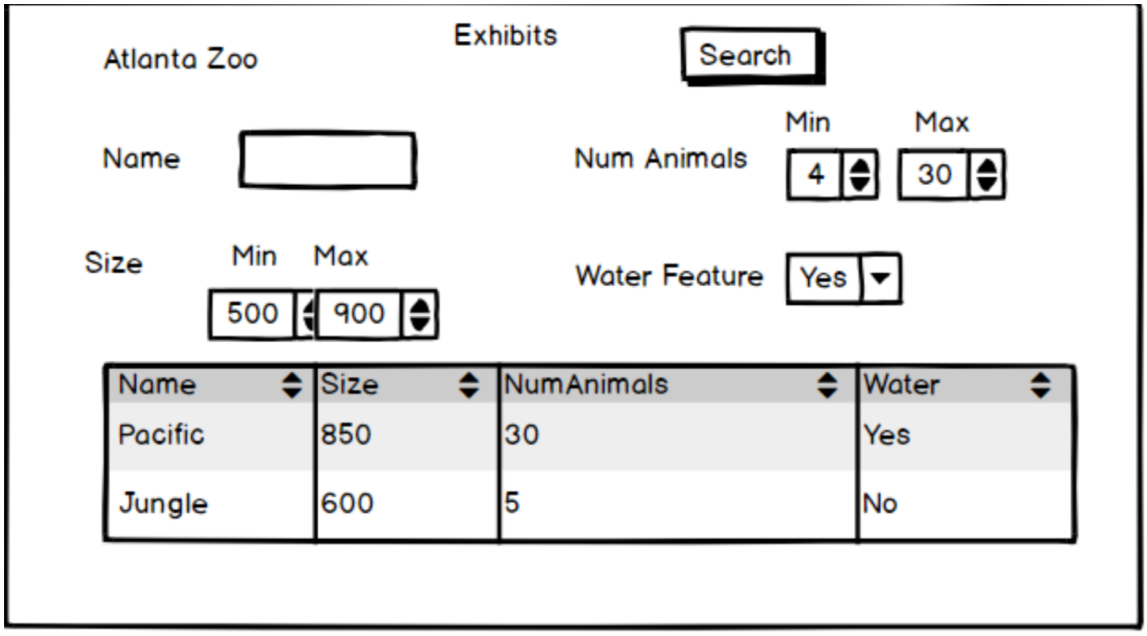

```
//Ensure username ‘chenxu0324’ is unique
SELECT Username
From User
WHERE Username = 'chenxu0324';

//Ensure email ‘chenxu@gmail.com’ is unique
SELECT Email
From User
WHERE Email = 'chenxu@gmail.com';

// If Username or Email already  exists, application will ask customer to enter a different one
//otherwise do the inset
//Scenario 1: registration of a visitor with the following specific: (Username: chenxu0324, Email: chenxu@gmail.com, Password: 12345678)
INSERT INTO User(Username, Email, Password, UserType) VALUES('chenxu0324', 'chenxu@gmail.com', '12345678', 'Visitor');
INSERT INTO Visitor(Username) VALUES('chenxu0324');

// Scenario 2: Registration of a staff with the following specific: (Username: haoliu, Email: haoliu@gmail.com, Password: 12345678)
INSERT INTO User(Username, Email, password, UserType) VALUES('haoliu', 'haoliu@gmail.com', '12345678', 'Staff');
INSERT INTO Staff(Username) VALUES('haoliu');
```

2.Visitor Functionality



(1)Search Exhibits
(2)Search Show
(3)Exhibit Detail
(4)Animal Detail
(5)Search Animals
(6)View Exhibit History
(7)View Show History

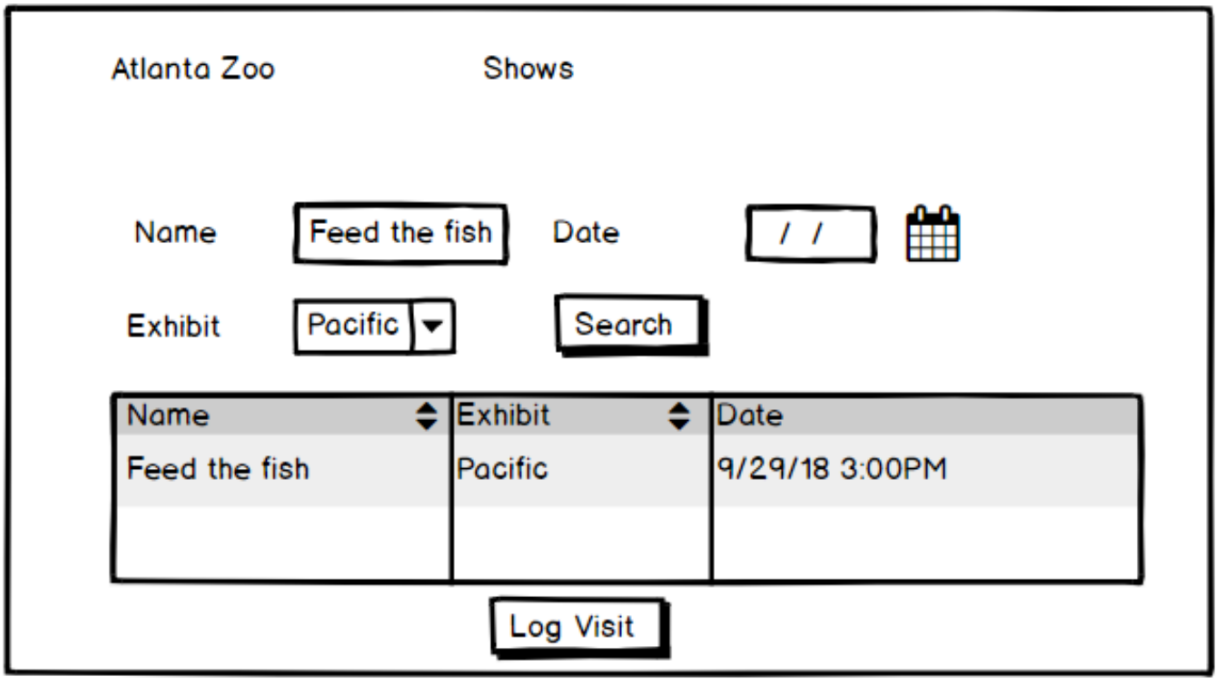
2.1. Search for Exhibits




```
//Scenario 1: Find an exhibit which is called 'Pacific'
SELECT E.Name, E.Size, count(*) as NumAnimals, E.Water_Feature as Water
FROM Exhibit as E, Animal As A
WHERE E.Name = 'Pacific' AND A.Exhibit= E.Name;

// Scenario 2: Find an exhibit which size is between 300 and 900, animal number is between 1 and 30
SELECT E.Name, E.Size, count(*) as NumAnimals, E.Water_Feature as Water
FROM Exhibit as E, Animal as A
WHERE E.size <=900 and E.Size >= 300 AND A.Exhibit = E.Name
GROUP BY A.Exhibit
HAVING count(*) >=1 and count(*) <=30;

//After user executed the scenario 2 search and clicked the sort arrow next to 'Size' the result will be ranked in ascending order by Exhibit Size
SELECT E.Name, E.Size, count(*) as NumAnimals, E.Water_Feature as Water
FROM Exhibit as E, Animal as A
WHERE E.size <=900 and E.Size >= 300 AND A.Exhibit = E.Name
GROUP BY A.Exhibit
HAVING count(*) >=1 and count(*) <=30
ORDER BY E.size;

//If user click the arrow next to 'Size' again, the result will be ranked in descending order by Exhibit Size
SELECT E.Name, E.Size, count(*) as NumAnimals, E.Water_Feature as Water
FROM Exhibit as E, Animal as A
```

```
WHERE E.size <=900 and E.Size >= 300 AND A.Exhibit = E.Name
GROUP BY A.Exhibit
HAVING count(*) >=1 and count(*) <=30
ORDER BY E.size DESC;
```

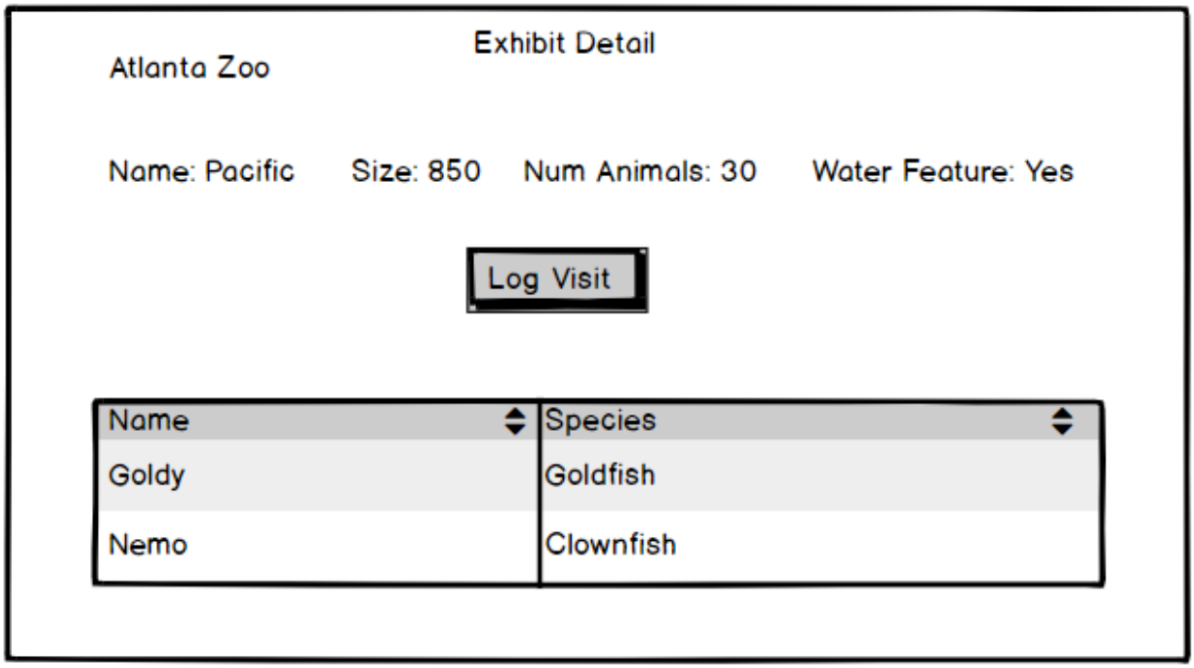
2.2. Search for Shows




```
//Scenario 1: Find all show in Exhibit 'Pacific' on 2008-11-11
SELECT Name, Exhibit, Datetime
FROM Shows
WHERE Datetime LIKE '2008-11-11%' AND Exhibit = 'Pacific';

//Scenario 2: Find all 'Feed the fish' show
SELECT Name, Exhibit, Datetime
FROM Shows
WHERE Name = 'Feed the fish';

//After user executed the scenario 1 search and clicked the sort arrow next to 'Name' the result will be ranked in alphabetical order by Show's name
SELECT Name, Exhibit, Datetime
```

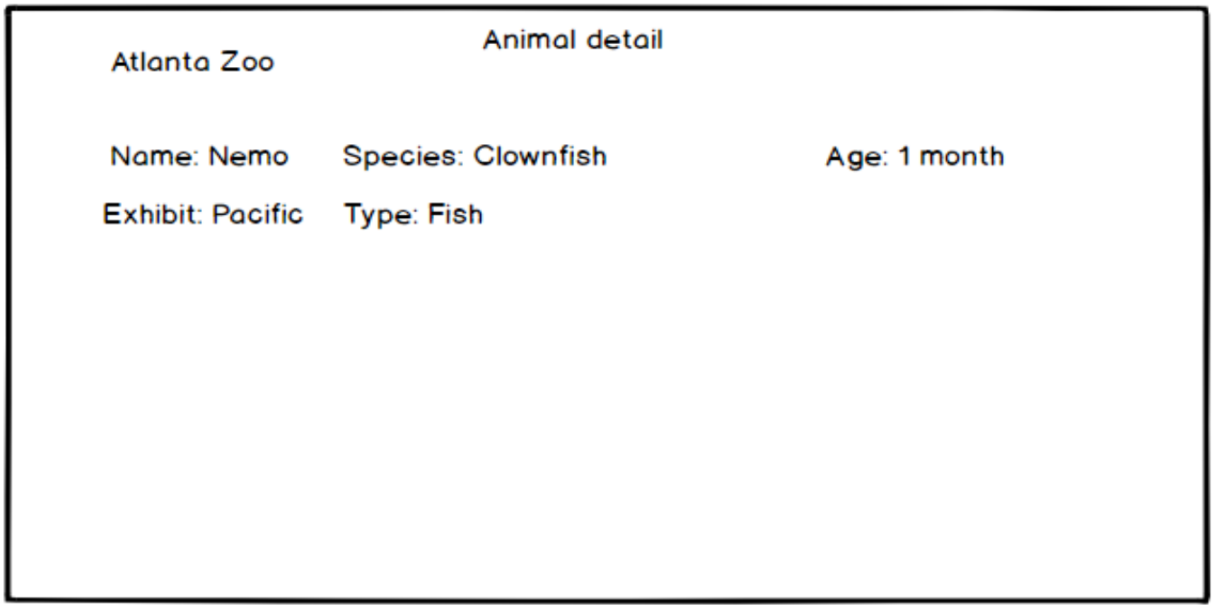
```
FROM Shows
WHERE Datetime LIKE '2008-11-11%' AND Exhibit = 'Pacific'
ORDER BY Name;

//If user click the arrow next to 'Name' again, the result will be ranked in reverse alphabetical order by Show's name
SELECT Name, Exhibit, Datetime
FROM Shows
WHERE Datetime LIKE '2008-11-11%' AND Exhibit = 'Pacific'
ORDER BY Name DESC;

//Log Visit to Visit_show will also trigger log visit to Visit_Exhibit
//Log Visitor Wenxin Tong's visit to the show 'Feed the fish' will excute the following sql
INSERT INTO Visit_Show(Visitor, ShowName, Datetime) VALUES('Wenxin Tong', 'Feed the fish', '2008-11-12 12:00:00');
INSERT INTO Visit_Exhibit(Exhibit, Visitor, Datetime) VALUES('Pacific', 'Wenxin Tong', '2008-11-11 12:00:00');
```

2.3. Exhibits Details



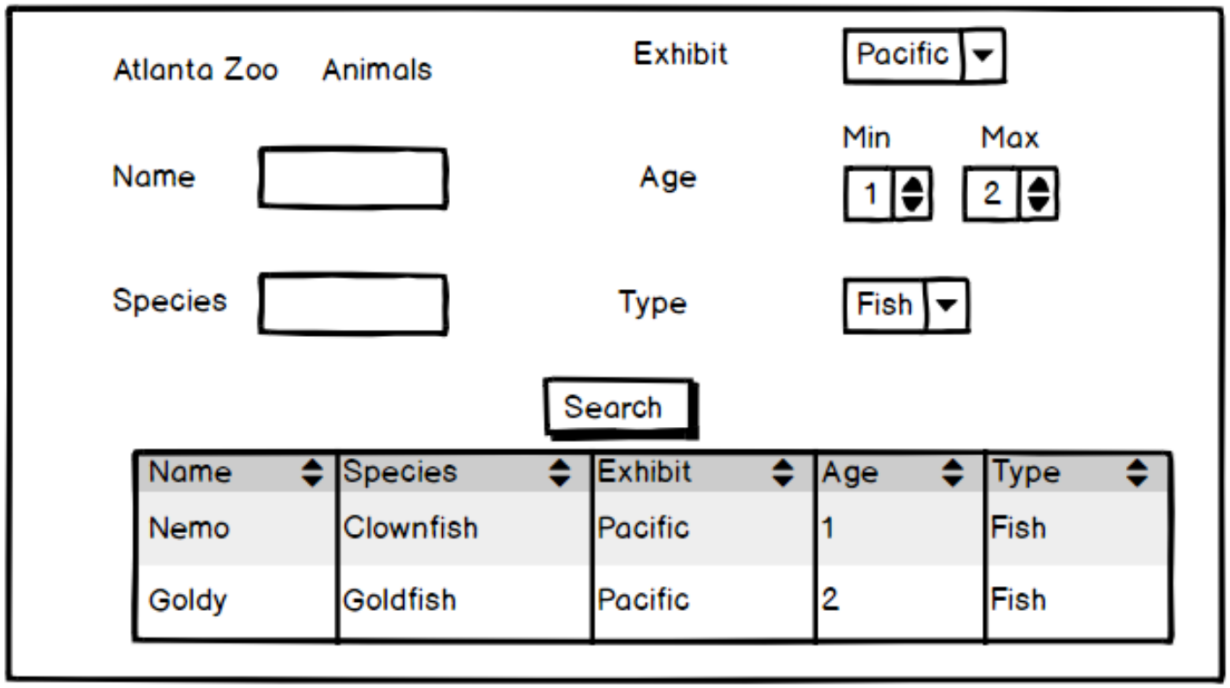
```
//Scenario 1: Exhibit detail page for exhibit 'Pacific'
SELECT E.Name, E.Size, COUNT(*) as Num_Animals, E.Water_Feature
FROM Exhibit as E, Animal as A
WHERE A.Exhibit = E.Name and E.Name = 'Pacific';

//Scenario 2: List of the name and species of all animal in the exhibit 'Pacific'
SELECT A.Name, A.Species
FROM Animal as A, Exhibit as E
WHERE E.Name = 'Pacific'
and A.Exhibit = E.Name;

//After user executed the scenario 1 search and clicked the sort arrow next to 'Name' the result will be ranked in alphabetical order by animal's name
SELECT E.Name, E.Size, COUNT(*) as Num_Animals, E.Water_Feature
FROM Exhibit as E, Animal as A
WHERE A.Exhibit = E.Name and E.Name = 'Pacific'
ORDER BY Name;

//If user click the arrow next to 'Size' again, the result will be ranked in reverse alphabetical order by animal's name
SELECT E.Name, E.Size, COUNT(*) as Num_Animals, E.Water_Feature
FROM Exhibit as E, Animal as A
WHERE A.Exhibit = E.Name and E.Name = 'Pacific'
ORDER BY Name DESC;


//Log Visit for visitor 'Wenxin Tong' to exhibit 'Pacific' and current time is 2018-11-13 12:00:00
INSERT INTO Visit_Exhibit(Exhibit, Visitor, Datetime) VALUES ('Pacific', 'Wenxin Tong',
'2018-11-13 12:00:00');
```

2.4. Animal Details




```
//Show the animal detail for the animal named 'Nemo' and species 'Clownfish'
SELECT *
FROM Animal
WHERE Name = 'Nemo' AND Species = 'Clownfish';
```

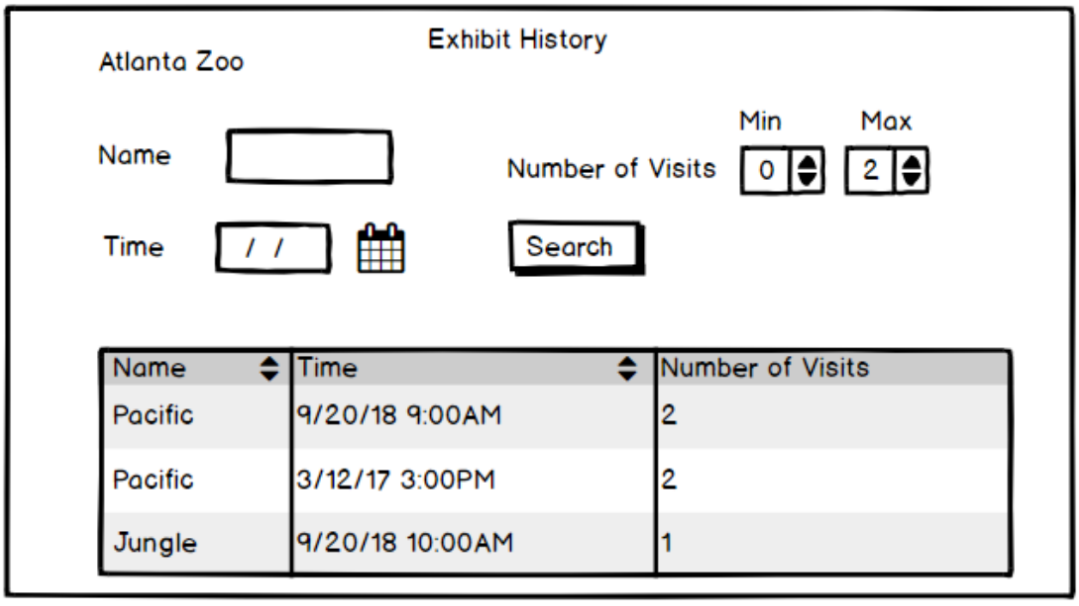
2.4. Search for Animals



.

```
Scenario 1: Search all animal from exhibit 'Pacific'
SELECT *
FROM Animal
WHERE Exhibit = 'Pacific';

Scenario 2: Search a bird called 'Nancy' who is 3 years old
SELECT *
FROM Animal
WHERE Name = 'Nancy' and Age = 3;

//After user executed the scenario 1 search and clicked the sort arrow next to 'Age' the result will be ranked in ascending order by Animal Age
SELECT *
FROM Animal
WHERE Exhibit = 'Pacific'
ORDER BY Age;

//If user click the arrow next to 'Age' again, the result will be ranked in descending order by Animal Age
SELECT *
FROM Animal
WHERE Exhibit = 'Pacific'
ORDER BY Age DESC;
```

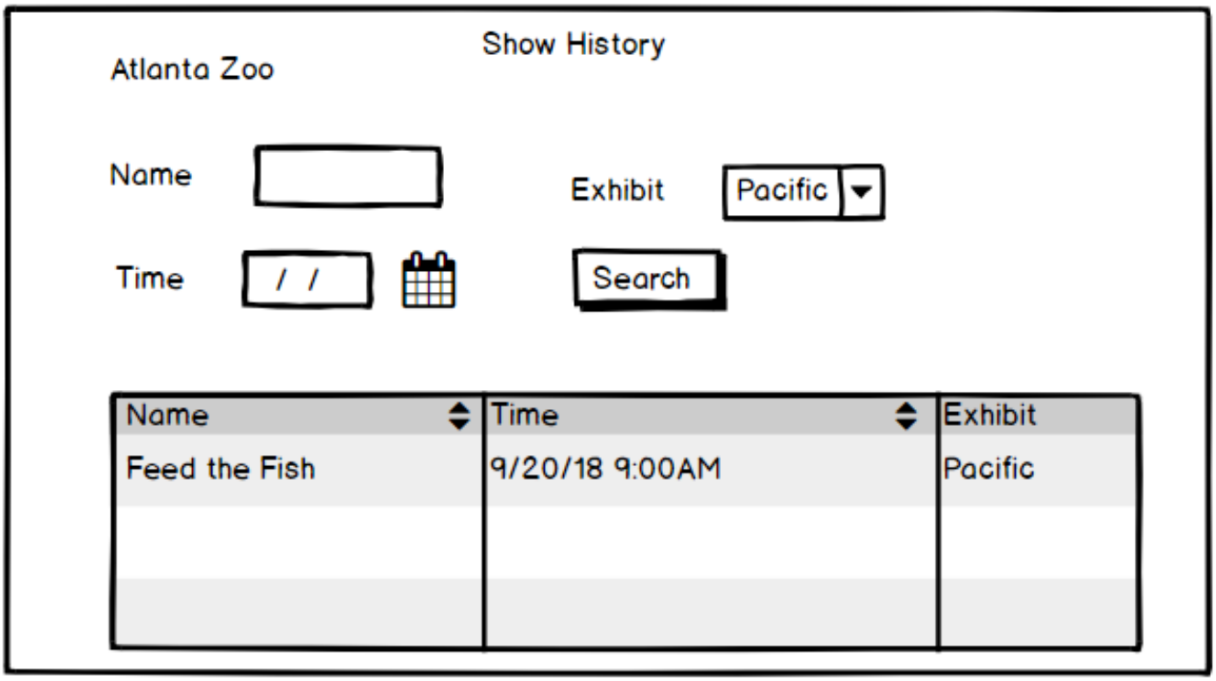
2.6. Exhibit History for Visitor



```
//Scenario 1: Find exhibit history for User whose username is 'Wenxin Tong'
SELECT Exhibit as Name, Datetime as Time, count as Number_of_Visits
FROM
((SELECT VE.Exhibit, VE.Datetime
FROM Visit_Exhibit as VE
WHERE VE.Visitor = 'Wenxin Tong') as n1
NATURAL JOIN
(SELECT VE2.Exhibit, COUNT(*) as count
FROM Visit_Exhibit as VE2
WHERE VE2.Visitor = 'Wenxin Tong'
GROUP BY VE2.Exhibit, VE2.Visitor)as n2) ;

//Scenario 2: Find Exhibit History for 'Wenxin Tong"s visit to the exhibit 'Pacific'
SELECT Exhibit as Name, Datetime as Time, count as Number_of_Visits
FROM
((SELECT VE.Exhibit, VE.Datetime
FROM Visit_Exhibit as VE
WHERE VE.Visitor = 'Wenxin Tong' and VE.Exhibit ='Pacific' ) as n1
NATURAL JOIN
(SELECT VE2.Exhibit, COUNT(*) as count
FROM Visit_Exhibit as VE2
WHERE VE2.Visitor = 'Wenxin Tong' and VE2.Exhibit ='Pacific'
GROUP BY VE2.Exhibit, VE2.Visitor)as n2) ;

//After user executed the scenario 1 search and clicked the sort arrow next to 'Name' the result
will be ranked in alphabetical order by Exhibit's name
SELECT Exhibit as Name, Datetime as Time, count as Number_of_Visits
FROM
((SELECT VE.Exhibit, VE.Datetime
FROM Visit_Exhibit as VE
WHERE VE.Visitor = 'Wenxin Tong') as n1
NATURAL JOIN
(SELECT VE2.Exhibit, COUNT(*) as count
FROM Visit_Exhibit as VE2
WHERE VE2.Visitor = 'Wenxin Tong'
GROUP BY VE2.Exhibit, VE2.Visitor)as n2)
ORDER BY Exhibit;

//If user click the arrow next to 'Name' again, the result will be ranked in reverse alphabetical
order by Exhibit's name
SELECT Exhibit as Name, Datetime as Time, count as Number_of_Visits
FROM
```

((SELECT VE.Exhibit, VE.Datetime
FROM Visit_Exhibit as VE
WHERE VE.Visitor = 'Wenxin Tong') as n1
NATURAL JOIN
(SELECT VE2.Exhibit, COUNT(*) as count
FROM Visit_Exhibit as VE2
WHERE VE2.Visitor = 'Wenxin Tong'
GROUP BY VE2.Exhibit, VE2.Visitor)as n2)
ORDER BY Exhibit DESC;

2.7. Show History for Visitor

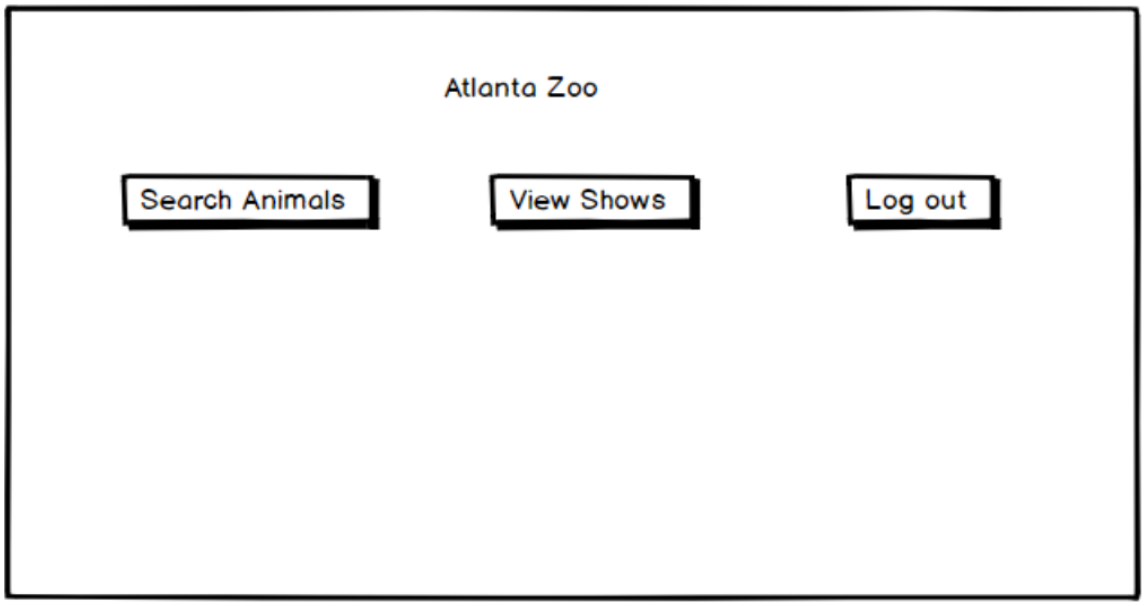


//Scenario 1: Visitor 'Wenxin Tong's visit to the show held in exhibit 'Pacific'
SELECT V.ShowName as Name, V.Datetime as Time, S.Exhibit
FROM Visit_Show as V, Shows as S
WHERE V.ShowName = S.Name AND V.Datetime = S.Datetime
AND V.Visitor = 'Wenxin Tong' AND S.Exhibit = 'Pacific';

//Scenario 2: Visitor 'Wenxin Tong's visit to all shows on 2018-11-12
SELECT V.ShowName as Name, V.Datetime as Time, S.Exhibit
FROM Visit_Show as V, Shows as S
WHERE V.ShowName = S.Name AND V.Datetime = S.Datetime
AND V.Visitor = 'Wenxin Tong' AND V.Datetime LIKE '2008-11-12%';

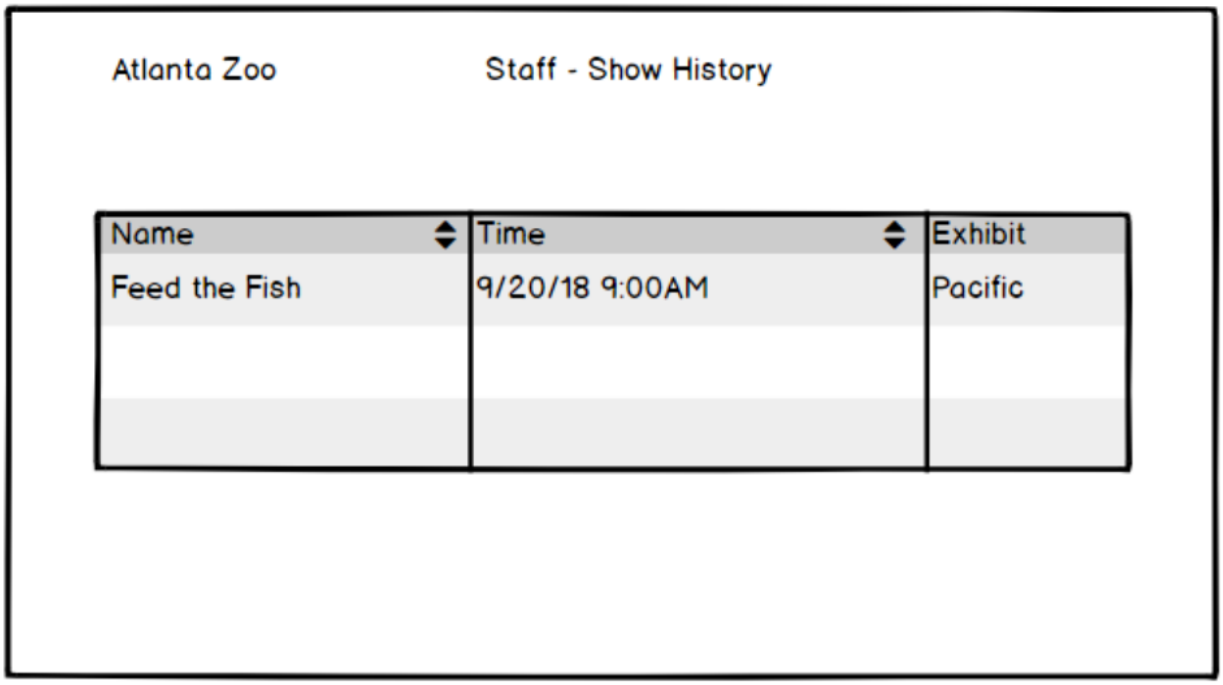
```
//After user executed the scenario 1 search and clicked the sort arrow next to 'Name' the result will be ranked in alphabetical order by Show's name
SELECT V.ShowName as Name, V.Datetime as Time, S.Exhibit
FROM Visit_Show as V, Shows as S
WHERE V.ShowName = S.Name AND V.Datetime = S.Datetime
AND V.Visitor = 'Wenxin Tong2' AND S.Exhibit = 'Pacific'
ORDER BY V.ShowName;

//If user click the arrow next to 'Name' again, the result will be ranked in reverse alphabetical order by Show's name
SELECT V.ShowName as Name, V.Datetime as Time, S.Exhibit
FROM Visit_Show as V, Shows as S
WHERE V.ShowName = S.Name AND V.Datetime = S.Datetime
AND V.Visitor = 'Wenxin Tong2' AND S.Exhibit = 'Pacific'
Order BY V.ShowName DESC;
```

3.Staff Functionality



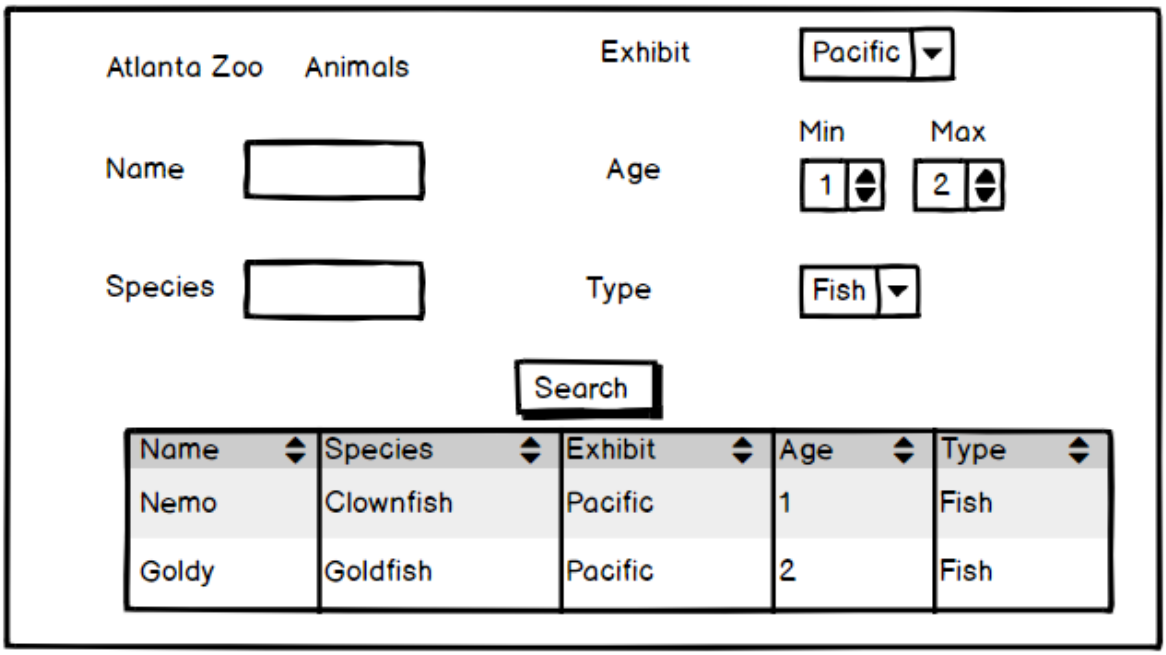
When a staff member logs in, they should have the following options:
(1) View Assigned Shows
(2) Search Animals
(3) Animal Care

3.1. View Assigned Shows



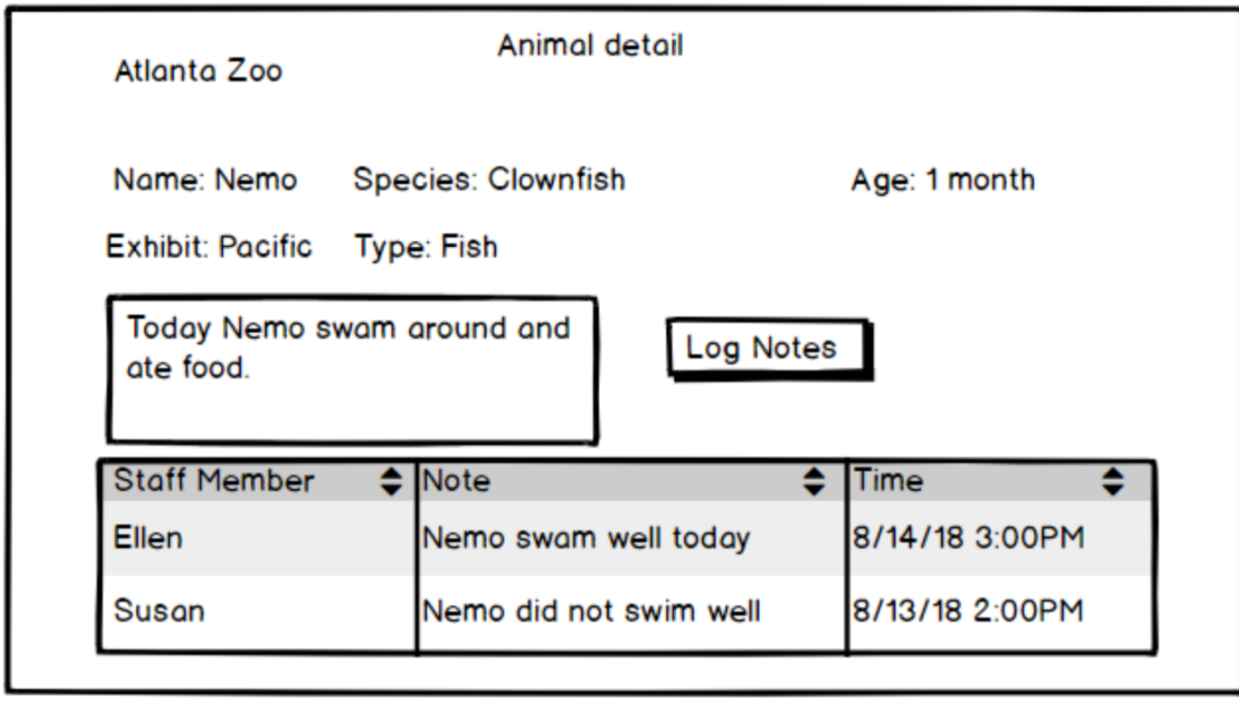

```
//Search shows hosted by staff 'hao liu'
SELECT Name, Datetime, Exhibit
FROM Shows
WHERE Host = 'hao liu';

//After the staff accessed the view shows window, he can click the sort arrow next to 'Time', the result will then be ranked in chronological order
SELECT Name, Datetime, Exhibit
FROM Shows
WHERE Host = 'hao liu'
ORDER BY  Datetime;

//If he click the sort arrow next to 'Time' again the result will them be ranked in reverse chronological order
SELECT Name, Datetime, Exhibit
FROM Shows
WHERE Host = 'hao liu'
ORDER BY  Datetime DESC;
```

3.2. Search for Animals (Staff)




```
//Scenario 1: search for all animal in exhibit 'Pacific'
SELECT *
FROM Animal
WHERE Exhibit = 'Pacific';


//Scenario 2: search a bird called Nancy who is 3 years old
SELECT *
FROM Animal
WHERE Name = 'Nancy' and Species = 'bird' and Age = 3;

//After user executed the scenario 1 search and clicked the sort arrow next to 'Age' the result will be ranked in ascending order by Animal Age
SELECT *
FROM Animal
WHERE Exhibit = 'Pacific'
ORDER BY Age;

//If user click the arrow next to 'Age' again, the result will be ranked in descending order by Animal Age
SELECT *
FROM Animal
WHERE Exhibit = 'Pacific'
ORDER BY Age DESC;
```

3.3. Animal Care

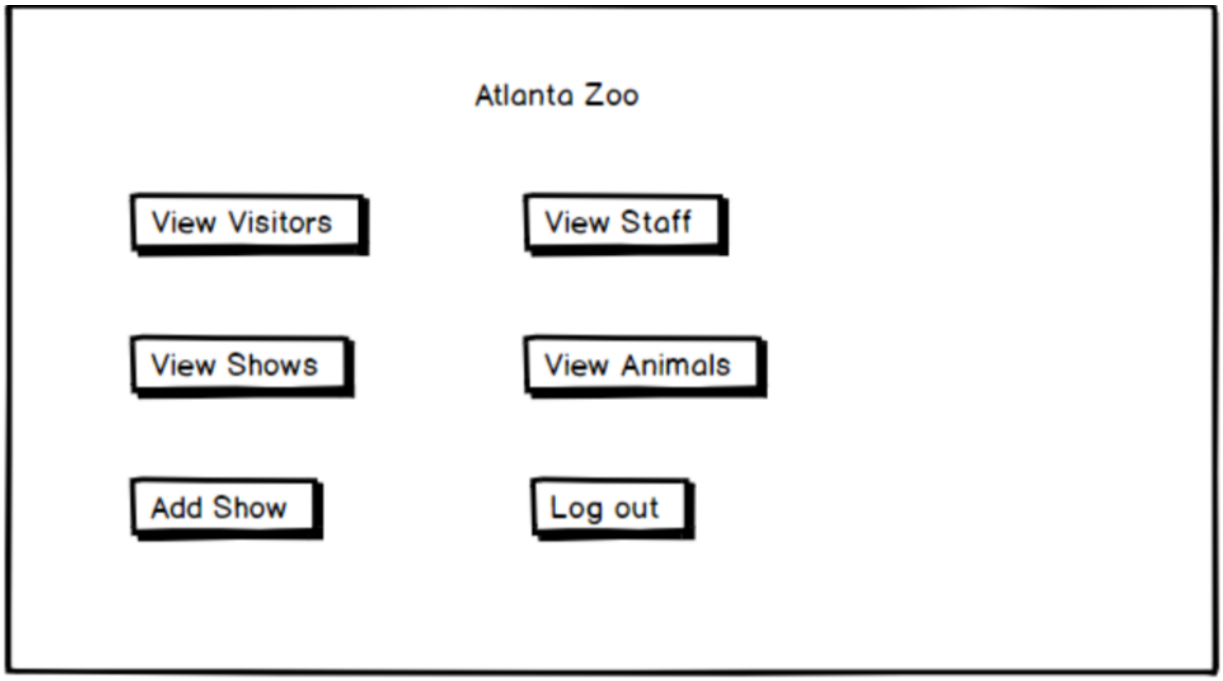



```
//Animal care record for ‘nemo’ the clownfish
SELECT Staff_Member, Text, Datetime
FROM Animal_Care
WHERE Animal = 'nemo' AND Species = 'clownfish';

//After staff accessed ‘nemo’ the clownfish’s animal care window, he  can click the sort arrow next to 'Time' then the result will be ranked in chronological order
SELECT Staff_Member, Text, Datetime
FROM Animal_Care
WHERE Animal = 'nemo' AND Species = 'clownfish'
ORDER BY Datetime;

//If staff clicks the arrow next to 'Time’ again, the result will be ranked in reverse chronological order

SELECT Staff_Member, Text, Datetime
FROM Animal_Care
WHERE Animal = 'nemo' AND Species = 'clownfish'
ORDER BY Datetime DESC;
```

```
//Log staff 'hao liu's animal care notes 'nemo is good' for 'nemo' the clownfish and the current time is 2008-12-11 12:00:00
INSERT INTO Animal_Care(Animal, Species, Staff_member, Datetime, Text)  VALUES('nemo', 'clownfish', 'hao liu2', '2008-12-11 12:00:00', 'nemo is good' );
```

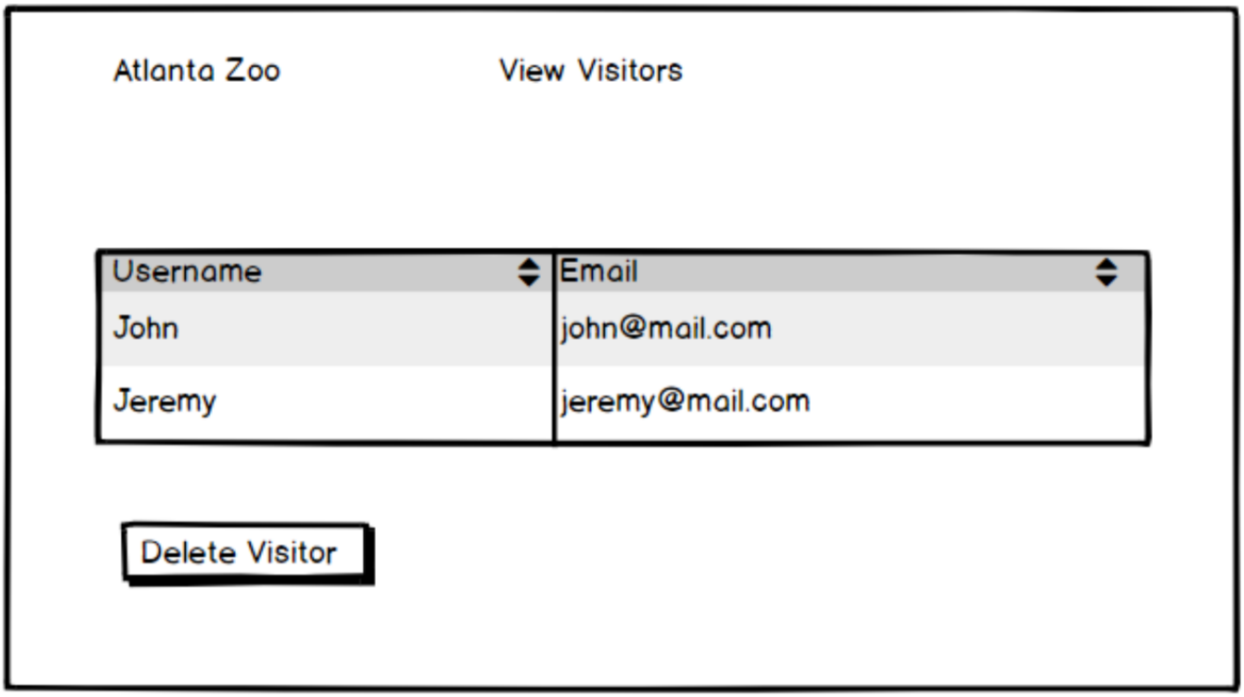
4.Administrator Functionality

When an administrator logs in, they should have the following options:
(1) View Visitors
(2) View Staff
(3) View Shows
(4) View Animals
(5) Add Animals
(6) Add Show



4.1. View Visitors

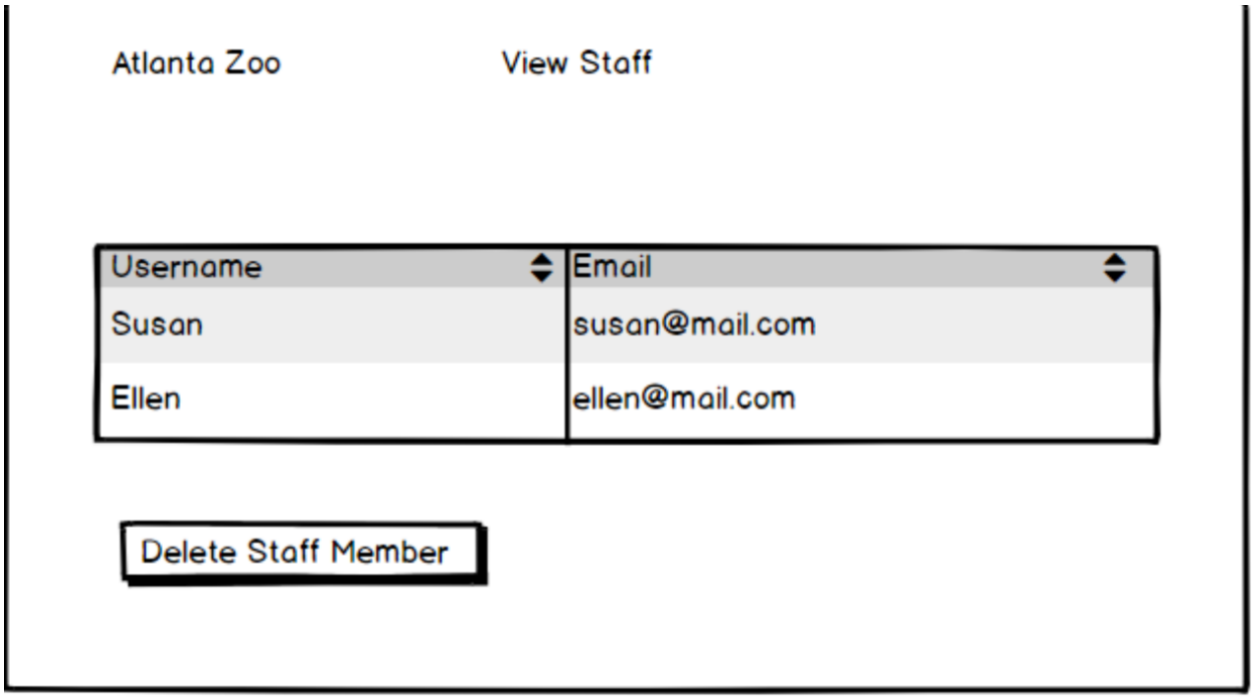
The admin should see a list of all the visitors with their username and email. The admin should be able to search the list of visitors. The admin can then remove visitor accounts, which would delete all information about the visitor.




```
//View all visitor
SELECT Username, Email
FROM User
WHERE UserType = 'Visitor';

//After admin accessed the view visitors window, he can click the sort arrow next to 'Username' then result will be ranked in alphabetical order by visitor's username
SELECT Username, Email
FROM User
WHERE UserType = 'Visitor'
ORDER BY Username;

//If admin click the arrow next to 'Username' again, the result will be ranked in reverse alphabetical order by visitor's username
SELECT Username, Email
FROM User
WHERE UserType = 'Visitor'
ORDER BY Username DESC;

//Delete visitor Wenxin Tong
DELETE
FROM User
WHERE Username = 'Wenxin Tong';
```

4.2.View Staff

The admin should see a list of all the staff members with their username and email, which the admin can search. The admin can remove staff accounts, which also removes all of the information about the staff member.

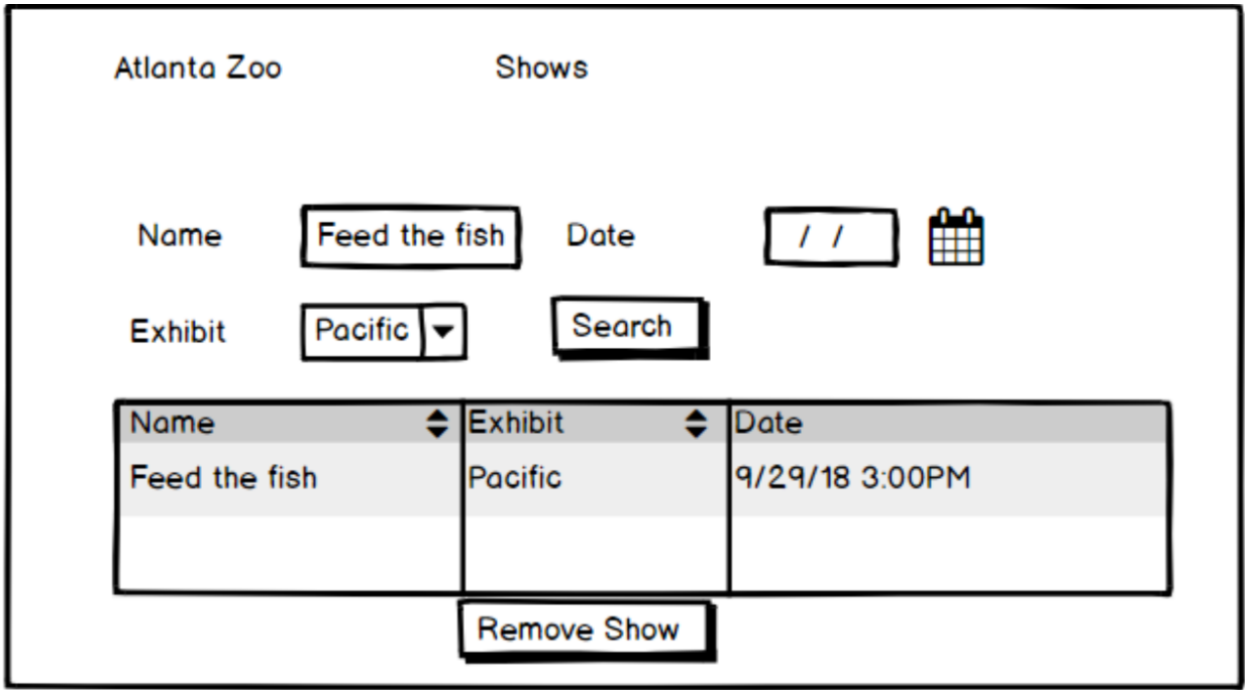



```
//View all staff
SELECT Username, Email
FROM User
WHERE UserType = 'Staff';

//After admin accessed the view Staffs window, he can click the sort arrow next to 'Username' then result will be ranked in alphabetical order by staff's username
SELECT Username, Email
FROM User
WHERE UserType = 'Staff'
ORDER BY Username;

//If admin click the arrow next to 'Username' again, the result will be ranked in reverse alphabetical order by visitor's username
SELECT Username, Email
FROM User
WHERE UserType = 'Staff';
ORDER BY Username DESC;
```

```
//Delete staff 'hao liu'
DELETE
FROM Staff
WHERE Username = 'hao liu';
```

4.3.View Shows

The admin can see the list of shows, and can search for shows. The admin can also remove shows. If a visitor has logged a visit to a show that was removed, that visit is also removed.

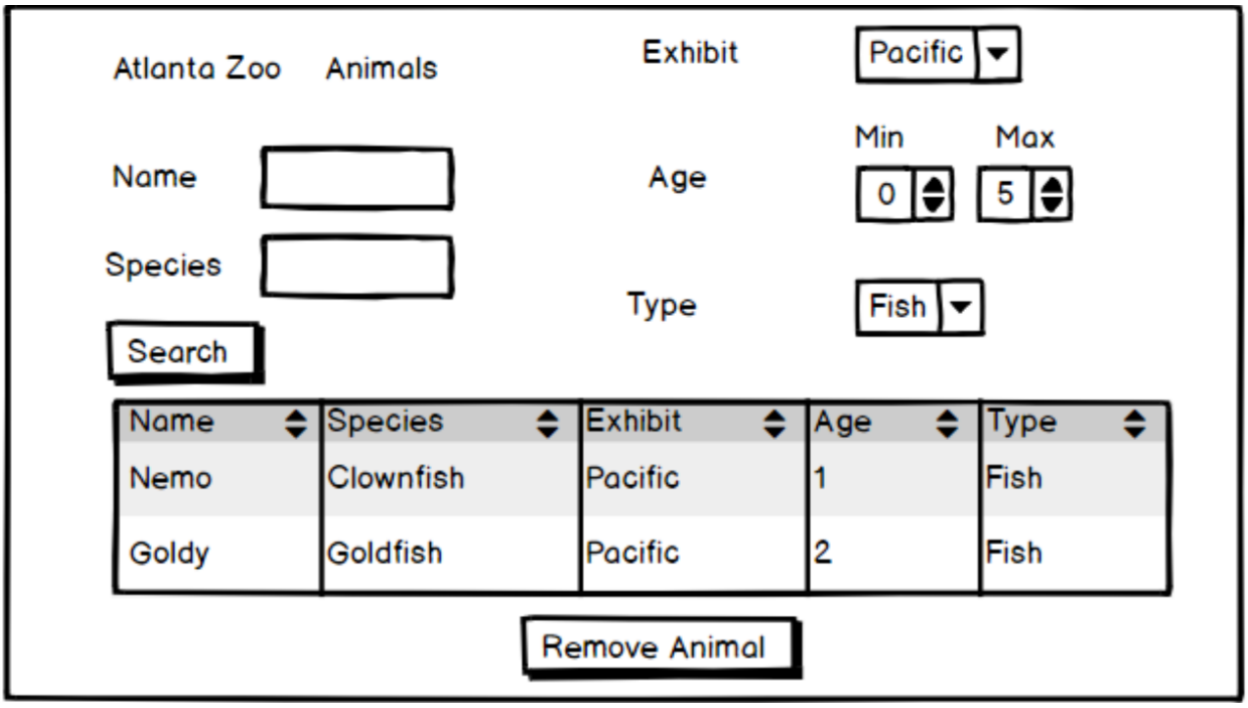



```
// Search for shows
// Scenario 1: Find all show in 'Pacific' Exhibit on 2018-11-16
SELECT Name, Datetime, Exhibit
FROM Shows
WHERE Exhibit = 'Pacific' AND Datetime LIKE '2018-11-16%';

// Scenario 2: Find all 'Feed the fish' show on 2008-11-11
SELECT Name, Datetime, Exhibit
FROM Shows
WHERE Datetime LIKE '2008-11-11%'  AND Name = 'Feed the fish';

//After admin executed the scenario 1 search and clicked the sort arrow next to 'Name' the result will be ranked in alphabetical order by Show's name
SELECT Name, Datetime, Exhibit
FROM Shows
WHERE Exhibit = 'Pacific' AND Datetime LIKE '2018-11-16%'
ORDER BY Name;
```

```
//If admin click the arrow next to 'Name' again, the result will be ranked in reverse alphabetical order by Show's name
SELECT Name, Datetime, Exhibit
FROM Shows
WHERE Exhibit = 'Pacific' AND Datetime LIKE '2018-11-16%'
ORDER BY Name DESC;

//Remove Show named 'GOGOGO' and on time '2018-11-16 12:00:00'
DELETE
FROM Shows
WHERE Name = 'GOGOGO' AND Datetime = '2018-11-16 12:00:00';

// Delete corresponding visit_show
DELETE
FROM Visit_Show
WHERE ShowName =  'GOGOGO' AND Datetime = '2018-11-16 12:00:00';
```

4.4.View Animals

The admin can view all the animals and search for animals. Animals can be removed by the admin.

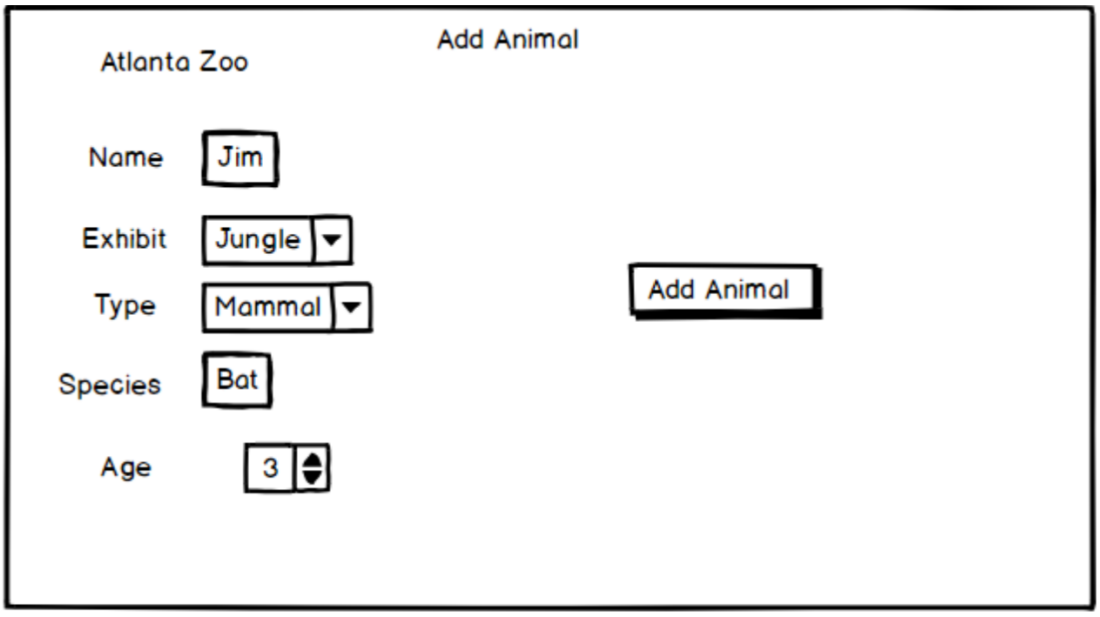



```
//Search for animals
//Scenario 1: Find all animals, without constrain
```

```
SELECT *
FROM Animal;

//Scenario 2: Find all fish live in exhibit ‘Pacific’ and age between 1 and 2
SELECT *
FROM Animal
WHERE Type = 'Fish' AND Age >= 1 AND Age <= 2 AND Exhibit = 'Pacific';

//After admin executed the scenario 1 search and clicked the sort arrow next to 'Age' the result will be ranked in ascending order by Animal Age
SELECT *
FROM Animal;
ORDER BY Age;

//If admin click the arrow next to 'Age' again, the result will be ranked in descending order by
SELECT *
FROM Animal;
ORDER BY Age DESC;

//Remove animal name 'john' and Species is ‘leopard’
DELETE
FROM Animal
WHERE Name = 'john' and Species = 'leopard';
```

4.5.Add Animals

The admin can add a new animal, and must supply all fields for the animal. These include name, species, type, age, and exhibit.



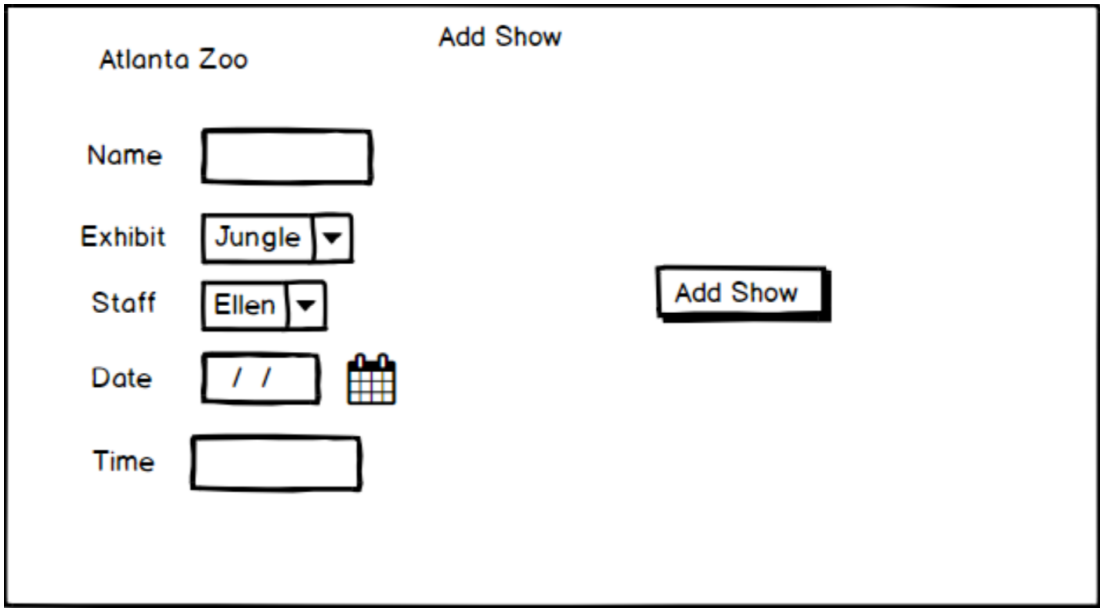
//Add the animal shown in the picture above to the database

INSERT INTO Animal(Name, Species, Type, Age, Exhibit) VALUES('Jim', 'Bat', 'Mammal', 3, 'Jungle');

4.6.Add Shows

The admin can add a new show to the schedule. The admin must provide a staff member to host the show, a name for the show, an exhibit for the show, and a time for the show. Here are a few more notes about shows.

- A staff member cannot host multiple shows at the same time.
- Each show needs one and only one host.
- It is possible for multiple shows to occur at the same time in an exhibit.



//Check if the staff ‘hao liu’ is already hosting a show at 2018-11-12 12:00:01
SELECT Host, Datetime
FROM Shows
WHERE Datetime = '2018-11-16 12:00:00' AND Host = 'hao liu';

//If the previous sql come back as null, then insert a ‘Watch Tiger’ show for ‘hao liu’ at that specific time
INSERT INTO Shows(Name, Datetime, Host, Exhibit) VALUES('Watch Tiger', '2018-11-12 12:00:01', 'hao liu', 'Africa');

Appendix

SQL language for creating the tables:

```
CREATE TABLE User
(Username  VARCHAR(20)  NOT NULL,
 Email          VARCHAR(40)  NOT NULL,
 Password    VARCHAR(40) NOT NULL,
 UserType    ENUM('Visitor', 'Staff', 'Admin'),
PRIMARY KEY(Username),
UNIQUE(Email) )
Engine = Innodb;

CREATE TABLE Admin
(Username   VARCHAR(20)  NOT NULL,
PRIMARY KEY (Username),
FOREIGN KEY (Username)  REFERENCES User (Username)
     ON DELETE CASCADE        ON UPDATE CASCADE )
Engine = Innodb;

CREATE TABLE Staff
(Username  VARCHAR(20)  NOT NULL,
PRIMARY KEY (Username),
FOREIGN KEY (Username)  REFERENCES User (Username)
ON DELETE CASCADE        ON UPDATE CASCADE)
Engine = Innodb;

CREATE TABLE Visitor
(Username  VARCHAR(20)  NOT NULL,
PRIMARY KEY (Username),
FOREIGN KEY (Username) REFERENCES User (Username)
ON DELETE CASCADE        ON UPDATE CASCADE)
Engine = Innodb;

CREATE TABLE Exhibit
(Name  VARCHAR(20)  NOT NULL,
Water_Feature  Boolean NOT NULL,
Size  INT  NOT NULL,
PRIMARY KEY(Name))
Engine = Innodb;

CREATE TABLE Animal
```

```
(Name VARCHAR(20) NOT NULL,
Species  VARCHAR(20) NOT NULL,
Type  VARCHAR(20) ,
Age  INT  NOT NULL,
Exhibit  VARCHAR(20) NOT NULL,
PRIMARY KEY(Name, Species),
FOREIGN KEY(Exhibit) REFERENCES Exhibit(Name)
 ON DELETE RESTRICT        ON UPDATE CASCADE )
Engine = Innodb;

CREATE TABLE Animal_Care
(Staff_member  VARCHAR(20)  NOT NULL,
Animal  VARCHAR(20) NOT NULL,
Species  VARCHAR(20) NOT NULL,
Datetime  DATETIME,
Text   VARCHAR(50),
PRIMARY KEY(Staff_member, Animal, Species, Datetime),
FOREIGN KEY(Staff_member) REFERENCES Staff(Username)
     ON DELETE CASCADE        ON UPDATE CASCADE ,
FOREIGN KEY(Animal,Species) REFERENCES Animal(Name,Species)
     ON DELETE CASCADE        ON UPDATE CASCADE )
Engine = Innodb;

CREATE TABLE Shows
(Name  VARCHAR(20) NOT NULL,
Datetime DATETIME  NOT NULL,
Exhibit VARCHAR(20)  NOT NULL,
Host VARCHAR(20)  NOT NULL,
PRIMARY KEY(Name, Datetime),
FOREIGN KEY(Exhibit) REFERENCES Exhibit(Name)
ON DELETE RESTRICT        ON UPDATE CASCADE,
FOREIGN KEY(Host) REFERENCES Staff(Username)
ON DELETE CASCADE        ON UPDATE CASCADE)
Engine = Innodb;

CREATE TABLE Visit_Show
(Visitor  VARCHAR(20) NOT NULL,
ShowName VARCHAR(20)  NOT NULL,
Datetime  DATETIME  NOT NULL,
PRIMARY KEY(Visitor, ShowName, Datetime),
FOREIGN KEY(Visitor) REFERENCES Visitor(Username)
     ON DELETE CASCADE        ON UPDATE CASCADE ,
FOREIGN KEY(ShowName,Datetime) REFERENCES Shows(Name,Datetime)
```

```
    ON DELETE CASCADE        ON UPDATE CASCADE )
Engine = Innodb;

CREATE TABLE Visit_Exhibit
(Visitor  VARCHAR(20) NOT NULL,
Exhibit VARCHAR(20)  NOT NULL,
Datetime DATETIME,
PRIMARY KEY(Exhibit, Visitor, Datetime),
FOREIGN KEY(Visitor) REFERENCES Visitor(Username)
ON DELETE CASCADE    ON UPDATE CASCADE,
FOREIGN KEY(Exhibit) REFERENCES Exhibit(Name)
ON DELETE RESTRICT    ON UPDATE CASCADE )
Engine = Innodb;
```