Project team 10
Meet-a-Pet

Team Members:
Ashish Bandgar
Joyta Choudhury
Manali Ghosh
Anirudh Lakshminarayanan

PROJECT PROPOSAL

Content:

Meet-a-pet is a basic pet adoption system application, where in people can browse and select pets for adoption. Users will be able to view variety of breeds and information for a particular pet/ animal they select. Users will be able to view pet from the catalogue and add it to wish-list according to the preferences. Payment can be done and pets can be collected from the closest possible adoption centre. Pet adoption System will provide a web based interface for users to browse and shop various type of animals/pets.

Scope and Objective:

Pet adoption System will provide a web based interface for users to browse and shop various type of animals/pets.

- User must be able to browse through the catalogue.
- User must have Login ID and Password to make the payment.
- User can check the availability of pet.
- User can access information like age of the dog, breed, suitable for which type of climate, food requirement, vaccination given, basic characteristic etc.
- User can add pet to wish-list and compare price.
- User must be able to view discount if there is any.
- User can buy accessory for pet.
- User can cancel the adoption process once requested before physically collecting the pet.
- User should receive a receipt about the payment made through e-mail.
- User can view the vaccination record of their respective pet.
- User can post suggestions regarding pet that is missing in our system.

PROJECT ENVIRONMENT

The project will be developed on Java and MySQL platforms. The software will have its database on local machine. JSP will be used along with html and CSS. MySQL workbench is to be used for database. And for front end - IntelliJ Idea will be used.

HIGH LEVEL REQUIREMENTS

User Roles

1. User
2. Member
3. Admin

User: A user can browse the website and can look for pets and pet's accessories. A user can also sign up to become a member of the online store.

Member: A member can browse through catalogue of the store and look for pets and its accessories and place orders.

Admin: An admin can add or remove pets from the pet catalogue. An admin can also add or remove accessories from the store.

User Stories

1. As a User, I want to browse the store's catalogue of pets.
2. As a User, I want to sign up for membership so that I can adopt pet.
3. As a Member, I want to adopt any available pet from the store.
4. As a Member, I want to buy any accessory for pet from the store.
5. As an Admin, I want to add new pet into the store catalogue.
6. As an Admin, I want to remove pet from the store catalogue.

Initial stories in decreasing order of priority:

1. As an Admin, I want to add new pet to the store catalog.
2. As a User, I want to browse the store's catalog of pet.
3. As a User, I want to sign up for membership so that I can adopt pet.
4. As a Member, I want to adopt any available pet from the store.
5. As a Member, I want to buy any accessory for pet from the store.
6. As an Admin, I want to remove pet from the store catalog.

HIGH LEVEL CONCEPTUAL DESIGN

Entities

1. User
2. Admin
3. Member
4. Pet
5. Accessory

Relationships:

1. Admin adds Pet.
2. Member adopts Pet.
3. Admin removes Pet.
4. Admin adds Accessory.
5. Admin removes Accessory.
6. Member buy Accessory.

SPRINT 1

User roles:

1. Admin

Admin: An admin can log into the system to add or delete pets from the online pet store to make them available for users to view them and adopt them.

User stories:

1. As admin, I want to log into the system.
2. As admin, I want to add new pets to the store catalog so that user can view them.
3. As admin, I want to delete pets from the store's catalog.

Story refinement, with notes:

1. As admin, I want to add new pets to the store catalog so that user can view them.

Note: - User must be logged in as admin to use this feature using username,password.

- Catalog must have pet's breed, age, sex, color and price.
- Only one pet is added at a time.
- There's no separate entity called Catalog. Catalog is simply a collection of pets.

Updated stories:

a. As an admin, I want to login to the system so that I can access features specific to my role.

b. As an admin, I want to logout from the system.

c. As an admin, I want to add a pet to the store's catalog.

2. As admin, I want to delete pets from the store's catalog.

Note:- User must be logged in as admin to use this feature using username,password.

- Only one pet is deleted at a time.
- There's no separate entity called Catalog. Catalog is simply a collection of pets.

Updated stories:

a. As an admin, I want to delete a pet from the store's catalog.

Refined Stories to be considered:

1. As an admin, I want to login to the system so that I can access features specific to my role.
2. As an admin, I want to add a pet to the store's catalog.
3. As an admin, I want to delete a pet from the store's catalog

Part 1: Conceptual design

Entity: **Admin**
Attributes:
username
password

Entity: **Pet**
Attributes:
Pet_id
Pet_type
Colour
Age
Sex
Price
Availability
Breed_name

Relationship: **Admin** adds **Pet**
Cardinality: One to many
Participation:
Admin has total participation
Dog has total participation

Relationship: **Admin** removes **Pet**

Cardinality: One to many
Participation:

Admin has total participation
Pet has total participation

Part 2: Logical design

Table: **Admin Login**
Columns:

username [primary key]
Password

Table: **Pet**
Column:

pet_id [primary key]
pet_type
color
age
sex

Table: **Pet_Availability**
Column:

pet_id [foreign key; references pet_id from Pet table]
price
availability

Table: **Pet_Breed**
Column:

pet_id [foreign key; references pet_id from Pet table]
breed_name

Part 3: Implementation:

Sql dump have been included with this submission.

Part 4: User Interface:

A user interface has been created through which we can access the database tables and demonstrate functionality relevant to sprint 1 of this project.

SPRINT 2

User stories:

1. As a User, I want to view pets in the store's catalog.
2. As a User, I want to view accessories in the store's catalog.
3. As a User, I want to register myself so that I can obtain membership.
4. As a User, I want to log in to the system so that I can adopt a pet.
5. As an Admin, I want to add accessory for dog to the store's catalog.
6. As an Admin, I want to remove accessory for dog from store's catalog.
7. As an Admin, I want to add accessory for cat to the store's catalog.
8. As an Admin, I want to remove accessory for cat from store's catalog.

Story refinement, with notes:

1. As a user, I want to register myself so that I can become member.

Note :- User must register using his/her information so that he/she can obtain membership.

.2. As a user, I want to log in to the system so that I can adopt pet.

Note:- User must be logged in to the system as member to adopt pet using username and password.

Refined Stories to be considered:

1. As a User, I want to register myself so that I can become Member.
2. As an Admin, I want to add accessory for pet to the store's catalog.
3. As an Admin, I want to remove accessory for pet from store's catalog.
4. As a User, I want to view pet in the store's catalog.
5. As a User, I want to view accessory in the store's catalog.
6. As a Member, I want to log in so that I can adopt a pet.
7. As a Member, I want to view pets in the store's catalog.
8. As a Member, I want to view accessories in the store's catalog.

Part 1: Conceptual design

Entity: **Admin**
Attributes:

username
password

Entity: **Pet**
Attributes:
Pet_id
Pet_type
Colour
Age
Sex
Price
Availability
Breed_name

Entity: **Customer**
Attributes: email
password
name [composite]
firstname
middlename
lastname
address [composite]
Add1
Add2
Zipcode
City
State
contact

Entity**: Accessory**
Attributes: accessory_id
accessory_name
price
pet_type
availability

Relationship:

Admin adds **Pet**
Cardinality: One to many
Participation:

Admin has total participation
Dog has total participation

Admin removes **Pet**
Cardinality: One to many
Participation:
Admin has total participation
Pet has total participation

Admin adds **Accessory**
Cardinality: one to many
Participation: **Admin** has total participation
Accessories has total participation

Admin removes **Accessory**
Cardinality: one to many
Participation: **Admin** has total participation
Accessory has total participation

User registers to become **Member**
Cardinality: one to one
Participation: **User** has total participation
Member has total participation

Part 2: Logical design

Table: **AdminLogin**
columns: username [primary key]
password
Highest normalization level: 4NF

Table: **MemberLogin**
columns: username [foreign key; references email of member table]
Password
Highest normalization level: 4NF

Table: **Member**
columns: email [primary key]
firstname
middlename
lastname

contact
add_line_1
add_line_2
zip
city
state

Highest normalization level: 2NF

Table: **Pet**
Column:
pet_id [primary key]
pet_type
color
age
sex
Highest normalization level: 4NF

Table: **Pet_Availability**
Column:
pet_id [foreign key; references pet_id from Pet table]
price
availability
Highest normalization level: 4NF

Table: **Pet_Breed**
Column:
pet_id [foreign key; references pet_id from Pet table]
breed_name
Highest normalization level: 4NF

Table**: Accessory**
columns: accessory_id [primary key]
accessoryname
color
pet_type
Highest normalization level: 4NF

Table: **Accessory_Availability**
columns: accessory_id [foreign key; references accessory_id from Accessory table]
price

availability

Highest normalization level: 4NF

Part 3: Normalization

The normal forms of each table have been mentioned in the logical design part.

Part 4: Implementation:

Sql dump have been included with this submission.

Part 5: User Interface:

A user interface has been created through which we can access the database tables and demonstrate functionality relevant to sprint 2 of this project.

Sprint 3:

User stories:

1. As a member, I want to add pets to the cart so that I can adopt them.
2. As a member, I want to add accessories to the cart so that I can buy them.
3. As a member, I want to remove accessories from the cart.
4. As a member, I want to check the availability of pets in the store.
5. As a member, I want to check the availability of accessories in the store.

Story refinement, with notes:

1. As a member I want to add pets to the cart so that I can buy them
   Note: Only one pet can be added to cart at a time.
2. As a member I want to add accessories to the cart so that I can buy them.
   Note: Only one accessory can be added to cart at a time.
3. As a member, I want to remove accessories from the cart.
   Note: Only one accessory can be removed from the cart at a time.

Refined Stories to be considered:

1. As a Member, I want to add a pet to the cart so that I can adopt it.
2. As a Member, I want to remove a pet from the cart.
3. As a Member, I want to add an accessory to the cart so that I can buy it.
4. As a Member, I want to check the availability of pets in the store.

5. As a Member, I want to check the availability of accessories in the store.
6. As a Member, I want to remove an accessory from cart.

Part 1: Conceptual design

Entity: **Admin**
Attributes:
- username
- password

Entity: **Pet**
Attributes:
- Pet_id
- Pet_type
- Colour
- Age
- Sex
- Price
- Availability
- breed_name

Entity: **Customer**
Attributes:
- Email
- Password
- Name [composite]
  - firstname
  - middlename
  - Lastname
- Address [composite]
  - Add1
  - Add2
  - Zipcode
  - City
  - State
- Contact

Entity**: Accessory**
Attributes:
- accessory_id
- Accessory_name
- Price
- Pet_type
- Availability

Entity: **Cart**

Attributes: cart_id
email
pet_id
accessory_id

Action: Member view Pet in the store's catalog.
Action: Member view Accessory in the store's catalog.
Action: Member checks availability of Pet.
Action: Member checks availability of Accessory.

Relationship:

Admin adds **Pet**
Cardinality: One to many
Participation:
Admin has total participation
Dog has total participation

Admin removes **Pet**
Cardinality: One to many
Participation:
Admin has total participation
Pet has total participation

Admin adds **Accessory**
Cardinality: one to many
Participation: **Admin** has total participation
Accessory has total participation

Admin removes **Accessory**
Cardinality: one to many
Participation: **Admin** has total participation
Accessory has total participation

User registers to become **Member**
Cardinality: one to one
Participation: **User** has total participation
Member has total participation

Member adds Pet to **Cart**
Cardinality: One to One
Participation: **Member** has total participation
Cart has total participation

Member removes Pet from **Cart**
Cardinality: One to One
Participation: **Member** has total participation
Cart has total participation

Member adds Accessory to **Cart**
Cardinality: One to One
Participation: **Member** has total participation
Cart has total participation

Member removes Accessory from **Cart**
Cardinality: One to One
Participation: **Member** has total participation
Cart has total participation

Part 2: Logical design

Table: **AdminLogin**
columns: username [primary key]
password
Highest normalization level: 4NF
INDEXES:
Clustered on username
Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier.

Table: **MemberLogin**
columns: username [foreign key; references email of member table]
Password
Highest normalization level: 4NF
INDEXES:
Clustered on username
Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier.

Table: **Member**

columns: email [primary key]
firstname
middlename
lastname
contact
add1
add2
zip
city
state

Highest normalization level: 2NF

INDEXES:

Clustered on email

Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier.

Non-Clustered on lastname

Justification:

i. As this field is primarily used for searching the most.
ii. This index will provide a quicker way to search as fields will be less in it.
iii. The index table may have less records than the actual table as this field can have duplicate values. Therefore searching in index table will be faster.

Table: **Pet**

Column:

pet_id [primary key]
pet_type_id [foreign key; references pet_type_id from Pet-type table]
breed_id [foreign key; references breed_id from Pet_Breed table]
color
age
sex
price
availability

Highest normalization level: 4NF

INDEXES:

Clustered on pet_id

Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier.

Table: **Pet_type**

Column : pet_type_id
pet_type_name

Highest normalization level: 4NF

INDEXES:
Clustered on pet_type_name
Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier.

Table: **Pet_Breed**
Column:
breed_id
breed_name
Highest normalization level: 4NF
INDEXES:
Clustered on breed_name
Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier.

Table**: Accessory**
columns: accessory_id [primary key]
accessory_name
color
pet_type_id [foreign key; references pet_type_id from Pet-type table]
price
availability
Highest normalization level: 4NF
INDEXES:
Clustered on accessory_name
Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier

Table: **Cart**
Columns: cart_id [primary key]
email [foreign key; references email from Member table]
Highest normalization level: 4NF

INDEXES:
Clustered on cart_id
Justification: Speeds up the search as the index table will have less fields and retrieval of the actual location of the record will be easier.

Table: **Cart_Pet**
Columns: cart_id [foreign key; references cart_id from Cart table]
pet_id [foreign key; references pet_id from Pet table]

Highest normalization level: 4NF

Table: **Cart_Accessory**
Columns: cart_id [foreign key; references cart_id from Cart table]
accessory_id [foreign key; references accessory_id from Accessory table]
Highest normalization level: 4NF

Part 3: Stored programs

Stored procedure: updateAccessoryAvailabiltyOnAdd
Parameters: IN acc_id INT (IN Parameter)
Goal: This stored procedure takes the input parameter as acc_id from the user and updates table Accessory_Availability which sets availability = 0.

Stored procedure: updatePetAvailabiltyOnAdd
Parameters: IN in_pet_id INT (IN Parameter)
Goal: This stored procedure takes the input parameter as in_pet_id from the user and updates table Pet_Availability which sets availability = 0.

Stored procedure: updateAccessoryAvailabiltyOnRemove
Parameters: IN acc_id INT (IN Parameter)
Goal: This stored procedure takes the input parameter as acc_id from the user and updates table Accessory_Availability which sets availability = 1.

Stored procedure: updatePetAvailabiltyOnRemove
Parameters: IN in_pet_id INT (IN Parameter)
Goal: This stored procedure takes the input parameter as in_pet_id from the user and updates table Pet_Availability which sets availability = 1.

Part 4: Indexes

The identification of indexes for each table has been done in logical design part.

Part 5: Implementation

Sql dump have been included with this submission.

Part 6: User Interface:

A user interface has been created through which we can access the database tables and demonstrate functionality relevant to sprint 3 of this project.