SOFTWARE ENGINEERING

CSC 4350 Spring 2018

Group Name: Renegon Bits

Jason Herrera, Brian Cabigon, Braxton McLean, Chris Scott, Nishant Sinha

1. Project Introduction

1.1. Project Description

Our group has decided to create a community-oriented Animal Record System. This software will require developers to work on a back end and front end. The front-end developers will have to design and implement a user interface that allows users to input information and make posts about animals they see around in their communities. The back-end developers will have to build databases that will support multiple users, and store the posts made by users and store information about animals provided by the users. The goal is to allow community members to input data on stray animals they see and keep that information in a database. Consequently, if that animal is seen again, it can be easily identified. In addition, it will allow recognized organizations, such as animal shelters, to input data on their animals and allow them to advertise animals in their shelters for a higher chance of adoption.

1.2. Team Members & Roles

Renegon Bits

Group Member	Role(s)
Jason Herrera	Group Coordinator, Documentation, Front-end Development
Brian Cabigon	Back-end Development, Databases
Braxton McLean	Back-end Development, Testing
Nishant Sinha	Front-end Development, Testing
Chris Scott	Front-end Development, Testing

2. Requirements Engineering

2.1. Outline Plan

The first thing we will do in requirements engineering is form a problem statement for our product which will be the basis for the requirements of our system. Then, we can create a context diagram for the system to see how it fits within the community. Simultaneously, we will write requirements for the system to further define what we want the system to do and how we want it to perform. Also, as part of the requirements we will add specifications to fully describe the system and help later on with implementation. Finally, we will create use cases from those requirement specifications to define the functionality of our system.

2.2. Schedule

Task	Effort	Duration	Dependencies	People
Problem Statement (T1)	5 person-days	1 day	None	Jason, Braxton, Brian, Chris, Nishant
Context Diagram (T2)	1 person-day	1 day	T1	Jason
Requirements (T3)	20 person-days	4 days	T1	Jason, Braxton, Brian, Chris, Nishant
Use Cases (T4)	20 person-days	4 days	T1, T2	Jason, Braxton, Brian, Chris, Nishant

2.3. Problem Statement

Our product is a database system that will allow users to provide data about animals in the community that may be stray. In addition, it will allow shelters and other recognized organizations to provide data about animals in the shelters. In addition, users will be able to receive data from the database about certain animals. The product is intended for community members concerned about stray animals in their areas. The animal shelters in the area are also able to make advertisements for animals in the shelters.

This product is intended to decrease the amount of unidentifiable stray animals in the community and help promote the adoption of animals from shelters. Our team has not identified any alternatives to this product which is a large reason this product should be developed. There is also a possibility of inviting advertisers to make the project profitable for our team while it helps the community at the same time.

Almost everything about our approach is novel because there aren’t alternatives now, but what makes it special is the ability for people in the community and animal shelters to communicate directly through the data each sends. This system not terribly complicated as it only needs servers for the database and some sort of user interface to work, so it is in the realm of feasibility. What makes this project interesting is the possibility of Facebook integration and the post system that will allow users to share information with others about animals that they have seen.

2.4. Requirements

2.4.1. The system shall store user information including name, email, geolocation, username, and password.

2.4.2. The system shall store animal information including images, name, species, and features.

2.4.3. The system shall allow users to create a new account.

2.4.3.1. If all fields are completed, and all entries are valid, account shall be created.

2.4.3.2. Error message shall be displayed if any entries are invalid.

2.4.3.3. Error message shall be displayed if any fields are empty.

2.4.4. The system shall validate login information of users.

2.4.4.1. If correct username and password are entered, allow access to system.

2.4.4.2. If username or password are incorrect, display error message.

2.4.4.3. Allow up to 5 incorrect accesses within 5 minutes from one IP address before disallowing access from that IP address for 15 minutes.

2.4.5. The system shall display information queried from the database.

2.4.5.1. If query is valid, and results are found, display results.

2.4.5.2. If query is invalid, display error message.

2.4.5.3. If no results found for query, display error message.

2.4.6. The system shall allow users to upload an image to the database with a response form attached with information about the animal.

2.4.6.1. If image file is valid, and response form is filled, entry shall be created.

2.4.6.2. Error message shall be displayed if any image file is invalid.

2.4.6.3. Error message shall be displayed if response form is invalid.

2.4.7. The system shall allow users to post onto the Feed.

2.4.7.1. If message is valid, post to Feed.

2.4.7.2. If message is invalid, display error message.

2.4.8. The system shall allow users to share direct links to posts.

2.4.8.1. If linked post exists, link is shared.

2.4.8.2. If linked post does not exist, display error message.

2.4.9. The system shall allow recognized organizations to collect a datasheet of posts in their defined geolocation work area.

2.4.9.1. If the user is a recognized organization, and the defined geolocation work area has posts, distribute datasheet.

2.4.9.2. If user is not a recognized organization account, display error message.

2.4.9.3. If geolocation work area has no posts, display error message.

2.4.10. The system shall allow recognized organization to post advertisements for animals in the shelter.

2.4.10.1. If user is a recognized organization, post advertisement.

2.4.10.2. If user is not a recognized organization, display error message.

2.4.11. The system shall allow administrators to delete posts made by other users.

2.4.11.1. If user is an administrator and post exists, delete post.

2.4.11.2. If user is not an administrator, display error message.

2.4.11.3. If post does not exist display error message.

2.4.12. The system shall allow administrators to delete entries in the database.

2.4.12.1. If user is an administrator and entry exists, delete entry.

2.4.12.2. If user is not an administrator, display error message.

2.4.12.3. If entry does not exist, display error message.

2.4.13. The system shall allow administrators to delete user accounts.

2.4.13.1. If user is an administrator and account exists, delete account.

2.4.13.2. If user is not an administrator, display error message.

2.4.13.3. If account does not exist, display error message.

2.5. Use Cases

2.5.1.

Summary: Login to system using account information.

Actors: Users and account database

Basic Course of Events:

1. User accesses the application.
2. The application prompts the user to log in to access the features of the application.
3. The user inputs their username and password into their respective fields.
4. The application connects to the database and determines if the account exists.
5. If the account exists, the application takes the account username and verifies that the input password matches the password in the database.
6. If the password and account match correctly to the database, the application logs the user in.

Alternative Paths: None. The user cannot access any other paths without logging in first.

Exception Paths: If the user's account does not exist, the application will prompt the user to enter a username and password to be added to the database. Once the user inputs a username and password, it is entered into the database and they can proceed with steps 2 – 6.

Trigger: For the user to interact with the application, they must first log in.

Assumptions: The user must already have an account or be willing to create an account with a username and password into the database.

Precondition: The user must have the application and an account.

Postcondition: The user is logged into their account and can access more features.

2.5.2.

Summary: Check geolocation of user

Actors: Users and application.

Basic Course of Events:

1. User logs into their account.
2. The application determines the user's location automatically.

3. The user's location is updated in the database.

Alternative Paths: If the system cannot determine a person's location automatically, the user can input a location manually.
Exception Paths:
Trigger: To determine what relevant posts to show based on geolocation, the system needs to know the user's geolocation. This system would accomplish this automatically.
Assumptions: The user has an account and is logged in
Precondition: The user must have the application and an account.
Postcondition: The system has the geolocation of the user to display a feed of posts that match their geolocation.

2.5.3.
Summary: The system shall be able to query the database.
Basic Course of Events:

1. The system sends a query to the database.
2. 0The database checks whether the query has correct permissions to perform the database operation.
3. If the query is well-formed and the query was sent with correct permissions, the database returns the queried information.
4. If the query is not well-formed or the query was not sent with correct permissions, the database returns an error.

Alternative Paths: None.
Exception Paths: If the database has too many queries to handle, the query may be rejected.
Precondition:

- The query was sent with correct access permissions for the database operation.
- The database contains information for the given query.

Postcondition: The query receives a response with information from the database.

2.5.4.
Summary: The system shall display information queried from the database.
Basic Course of Events:

1. The database is queried and correct information is returned.
2. The response from the database is displayed by the system.

Alternative Paths: None
Exception Paths: If incorrect information is returned by the database (wrong encoding), the information may not appear correctly.
Precondition:

- A well-formed query with correct permissions is sent to the database.

- The database returns the correct information.

Postcondition: The queried information is displayed on the system

2.5.5.
Summary: The system shall allow users to post onto the Feed.
Basic Course of Events:

1. A user with correct permissions posts a well-formed post to the Feed.
2. The post is sent to the database and the database returns a response to the query that it was correctly received
3. The post shows up on the Feed.

Alternative Paths: None.
Exception Paths:

- If the database does not correctly receive the post in step 2, it will not show up in the Feed.
- If the user's post is not well-formed, the system displays an error.

Precondition:

- The user is logged in and has permissions to post to the Feed.

Postcondition: The Feed now shows the user's post.

2.5.6.
Summary: Display posts made by other users in the Feed
Basic Course of Events:

1. Completion of use case *Check geolocation of user*.
2. User selects the option to view the Feed.
3. The system finds posts made by users in the same geolocation.
4. If there are posts made by users in the geolocation, those posts are displayed

Alternative Paths: At step 3, the user can opt to enter a geolocation manually instead and execution proceeds to step 4.
Exception Paths: If there are no posts in the geolocation, the system will display a message that says there were no posts found in place of step 4.
Precondition: The user must have a geolocation.
Postcondition: Posts have been displayed for the user.

2.5.7.
Summary: Share direct link viewable by others
Basic Course of Events:

1. Complete use case *Login to system using account information*.
2. Complete use case *Display posts made by other users in the Feed*.
3. User selects option to share a post made by another user.
4. The system generates a link for that post.
5. The system provides the link to the user.
6. User is prompted to copy the link.

7. The system returns the user to the Feed.

Alternative Paths: At steps 5 and 6, the user can choose to cancel this operation and go to step 7.
Exception Paths: If post does not exist, an error message is displayed and execution proceeds to step 7.
Precondition: Use case login to system using account information was successful and use case display posts made by other users in the Feed was successful.
Postcondition: Link is provided to user.

2.5.8.
Summary: Collect datasheet of posts in defined geolocation work area
Basic Course of Events:

1. Complete use case *login to system using account information* with credentials of a recognized organization
2. Complete use case *check geolocation of user*
3. User selects option to collect datasheet of posts.
4. The system finds the posts made by users in the geolocation.
5. The system creates a file with the posts.
6. System provides a preview of the file.
7. User selects the download option.
8. User is returned from the datasheet collection.

Alternative Paths: At step 3, user can manually enter a geolocation and go to step 4. At step 6, the user can cancel this operation and go to step 8.
Exception Paths: If there are no posts in the geolocation, an error message is displayed and execution proceeds to step 8.
Precondition: Use case login to system using account information was successful and use case check geolocation of user is successful.
Postcondition: Datasheet is provided to the user.

2.5.9.
Summary: Post advertisements for animals in shelter in Feed.
Basic Course of Events:

1. Complete use case *Login to system using account information*.
2. Complete use case *Display posts made by other users in the Feed*.
3. User selects option to make a post (authorized as an organization in step 1).
4. The system provides a response prompt like use case 3
    a. Information, image upload, additional details, personal blurb, etc.
5. Repeat set 4 as prompted until the user selects finish.
6. Query the user about how many hours/days they wish the ad to run.
7. Confirms selection, and uploads information to database.
8. The system returns the user to the Feed.

Alternative Paths: At steps 3 through 7, the user can choose to cancel this operation and go to step 2.

Exception Paths: If inaccurate/unacceptable date ranges are entered at step 6, they are prompted to enter the information again/differently.
Precondition: Use case login to system using account information was successful and use case display posts made by other users in the Feed was successful.
Postcondition: Ads will run against Feed spots upon approval.

2.5.10.
Summary: Toggle geolocation filters
Basic Course of Events:

1. Complete use case *Login to system using account information*.
2. Complete use case *Display posts made by other users in the Feed*.
3. User navigates to the Settings of the app.
4. The User toggles on/off geolocation filtering
5. The user may navigate away from the Settings.

Alternative Paths: None
Exception Paths: If the device's location services are disabled/unavailable, prompt the user to enable them via their phone's settings.
Precondition: Use case login to system using account information was successful and use case display posts made by other users in the Feed was successful.
Postcondition: After step 4, if the option is toggled on, the Feed contents will be more weighted to display nearby posts. If disabled, posts based on the initial state entry will be displayed.

2.5.11.
Summary: Delete posts made by other users.
Basic Course of Events:

1. Complete use case *Login to system using account information*.
2. Complete use case *Display posts made by other users in the Feed*.
3. User selects a post made by another user in the Feed.
4. The user selects the post options button
5. The user selects the option to delete post.
6. The system returns the user to the Feed.

Alternative Paths: At steps 3 - 5, the user can choose to cancel this operation and go to step 6.
Exception Paths: If user is not authorized as an administrator, Step 5 will return an error warning the user they are not authorized.
Precondition: Use case login to system using account information was successful and use case display posts made by other users in the Feed was successful.
Postcondition: The post is deleted and not viewable by future users.

2.5.12.
Summary: Remove Entries in Database
Basic Course of Events:

1. The system sends a query to the database
2. The database checks whether the client has correct permissions to perform the query.
3. If the query is sent with permissions and is a correct query, then the database entry that aligns with the requirements of the query will be removed and the client will receive an OK.
4. If the query is went without permissions or is an erroneous query, then the client will receive an error and nothing in the database will be removed.

Alternative Paths: None
Exception Paths: If the database experiences excessive traffic, the query may be rejected.
Precondition:

- The query was sent with permissions to perform a database operation
- The database contains information that aligns with the requirements of the query.

Postcondition: The client receives an OK and the requested entry in the database will be removed.

2.5.13.
Summary: Remove Accounts in System
Basic Course of Events:

1. A user with the proper permissions will log in to the system and go to their settings where they can remove their account.
2. They will select a button in order to remove their account
3. The user will be prompted that their account will be non-recoverable if they proceeds
4. If they proceed, query the database and remove their user account entry
5. If they do not proceed, return back to the settings screen.

Alternative Paths: None
Exception Paths: If the database experiences excessive traffic, the query may be rejected.
Precondition:

- The user is logged in with proper permissions to access their settings
- The query was sent with permissions to perform a database operation
- The database contains information that aligns with the requirements of the query.

Postcondition: The client's account and information will be removed from the database and the user will be led back to the home page.

2.5.14.
Summary: Delete advertisements made by organization
Basic Course of Events:

1. A user with the proper permissions will log in to the system and go to an interface that will allow for the removal of posts that they deem as non-genuine advertisement posts.
2. They will select a post that they deem as a non-genuine advertisement post
3. They will select an option to delete the said post.

4. They will prompted a confirmation for deleting the post
5. If they proceed, the post's database entry will be removed
6. If they do not proceed, the post's database entry will not be removed and the user will be returned to the options page.

Alternative Paths: None

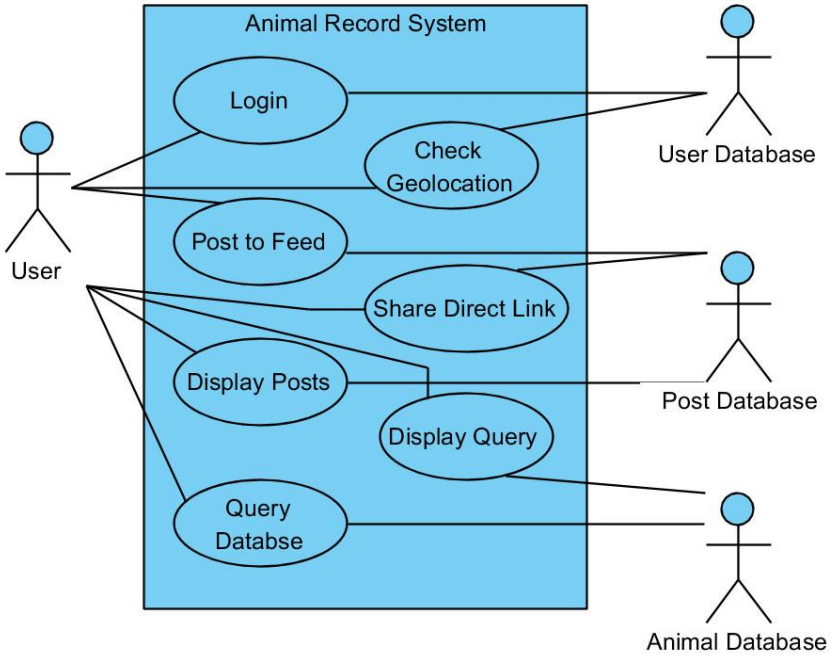
Exception Paths: If the database experiences excessive traffic, the query may be rejected.

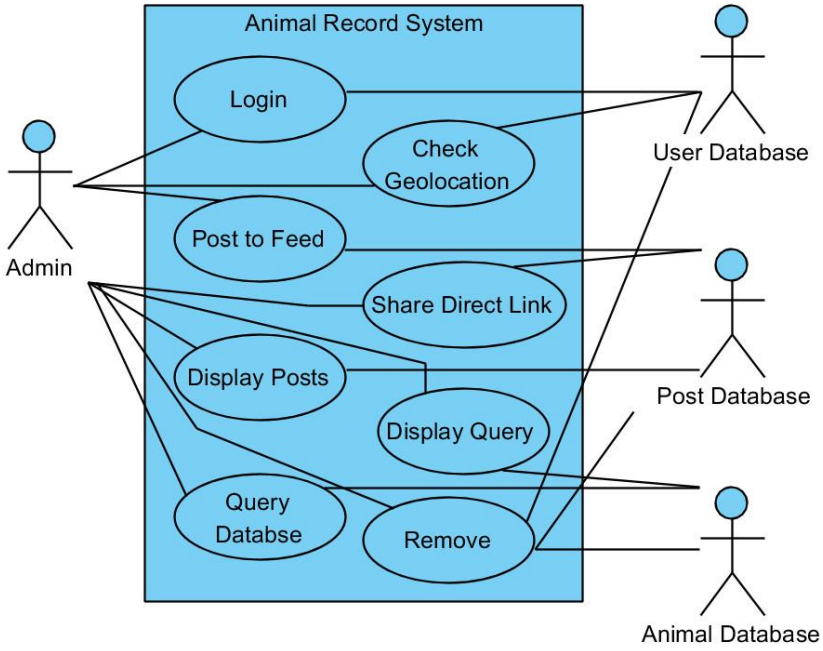
Precondition:

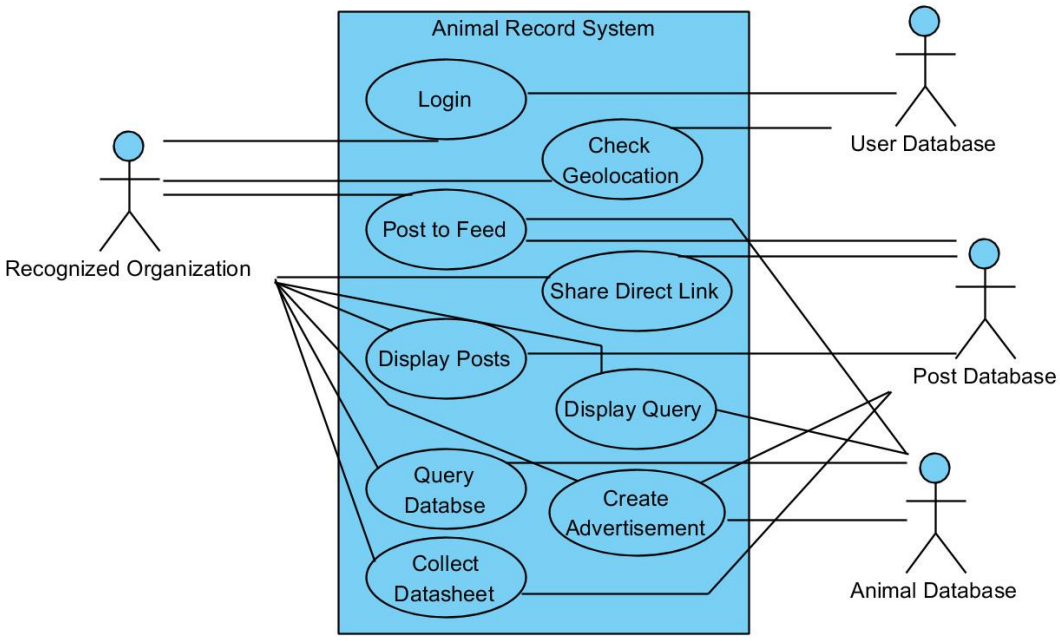
- The user is logged in with proper permissions to access their settings
- The query was sent with permissions to perform a database operation
- The database contains information that aligns with the requirements of the query

Postcondition: The post will be removed from the database and the user will be led back to the list of posts.

2.6. Use Case Diagrams




Animal Record System
Login
Check Geolocation
Post to Feed
Share Direct Link
Display Posts
Display Query
Query Databse
Remove
Admin
User Database
Post Database
Animal Database


Animal Record System
Login
Check Geolocation
Post to Feed
Share Direct Link
Display Posts
Display Query
Query Databse
Create Advertisement
Collect Datasheet
Recognized Organization
User Database
Post Database
Animal Database

2.7. Test Cases

2.7.1.

Requirement ID: 4
Requirement Description: The system should be able to query the database.
Rationale: To ensure that the database is atomic and will always either return a correct response or an error.
Inputs:

- A well-formed query.
- An ill-formed query.

Outputs:

- The correct information from the query.
- An error.

Persistent Changes:

- If the query to the database adds a state or updates a state, the database must then contain that information.

Related Requirements and Use Cases: 5, 6
Test Cases:

- Send both well-formed and ill-formed queries to the database and then check the database to ensure any persistent changes remain.

Creator: Nishant Sinha

2.7.2.

Requirement ID: 5
Requirement Description: The system shall display information queried from the database.
Rationale: To ensure the queried information from the database is visible.
Inputs:

- A search term. E.g. dogs, cats, spotted.

Outputs:

- A list of posts relevant to the search terms.

Persistent Changes: None.
Related Requirements and Use Cases: 4
Test Cases:

- Different search terms.
- Cats, dogs, spotted, golden retriever.

Creator: Nishant Sinha

2.7.3.

Requirement ID: 6
Requirement Description: The system shall allow users to post on the Feed.
Rationale: So that users can engage with posts and view other user's content
Inputs: None. The user should be able to view the feed as soon as they are logged in.
Outputs: All posts in a scrollable container.
Persistent Changes: None.
Related Requirements and Use Cases: 5
Test Cases:

- Viewing the system not logged in.
- Viewing the system as a user properly logged in.

Creator: Nishant Sinha

2.7.4.
Requirement ID: 7
Written by: Braxton McLean
Requirement Description: Post to Feed as user
Rationale: This is the source of the majority of the application's user content, to allow users to post sightings/information about lost animals.
Inputs:

1. The user navigates to the home/main page of the application
2. The user presses a "make post" button
3. A form requesting additional information will appear, including photo, description of animal, location, and time (required), as well as a text description
4. A confirmation notice of post submission

Outputs: The application will send the post data to the backend where it will be processed and added to the database.
Persistent Changes: The post data will be added to the database for querying and appearing in Feeds.
Related Requirements and Use Cases: Requirements 4, 5, 7 - Uses 1, 3, 6.

2.7.5.
Requirement ID: 8
Requirement Description: Share direct link to posts viewable by others
Rationale: This requirement was to allow for users to share posts with other people in order to spread information of lost animals that they may be familiar with.
Inputs:

The user navigates to a post that exists in the database

They will be able to copy a link that leads directly to the post so that the post may be viewable for others

Outputs: The output will be a link that they can give to others to allow them to view the post.
Persistent Changes: None
Related Requirements and Use Cases: Requirement 4, 8
Test Cases: 5

2.7.6.
Requirement ID: 9
Requirement Description: Allow recognized organizations to collect a datasheet of posts in their defined geolocation work area
Rationale: This requirement was to allow organizations to get a log of the animals that have been seen in their immediate area and they can work with people to see who the animal belongs to as well as who they can return it to if they find it among their animals.
Inputs:

The organization user navigates to the Feed
The organization user will click a button
The organization user will then download a datasheet of posts written for the area close to the organization
Outputs: The output will be a datasheet of posts in their defined geolocation work area
Persistent Changes: Persistent Changes
Related Requirements and Use Cases: Requirement 4, 5, 9
Test Cases: 5

2.7.7.
Requirement ID: 10
Written by: Braxton McLean
Requirement Description: Shelters can post advertisements for animals in shelters to be displayed to other users.
Rationale: This is the source of the majority of the application's user content, to allow users to post sightings/information about lost animals.
Inputs:

1. The user navigates to the home/main page of the application
2. The user, authorized as a shelter presses the "make post/advertisement" button
3. A form requesting additional information will appear, including photo, description of animal, shelter name and locations well as a text description.
4. Repeat steps 3 until the user selects "finish"
5. Prompt for a duration with a specified maximum and minimum.
6. A confirmation notice of post submission

Outputs: The application will send the post data to the backend where it will be processed and added to the database as advertisements to be displayed to users.
Persistent Changes: The advertisement data will be added to the database for querying and appearing in Feeds.
Related Requirements and Use Cases: Requirements 4, 5, 10 - Uses 1, 2, 3, 7, 10.

2.7.8.
Requirement ID: 11
Written by: Braxton McLean
Requirement Description: Administrators may delete posts made by other users.
Rationale: Administrators will be tasked with moderation of submissions, or review of submissions, as well as active user status of viewing the Feed. They may need to delete posts they find violate the purpose of the application.
Inputs:

1. The user navigates to the home/main page of the application and views the Feed.
2. The user, authorized as an administrator navigates to a particular post
3. The user selects the "Extra Options" menu of the post and selects "delete"
4. A confirmation notice of post deletion.

Outputs: The application will send a request to the backend where it will be processed and post removed from the database.
Persistent Changes: Completion of these steps will remove the entry from the Feed and database.

Related Requirements and Use Cases: Requirements 4, 5, 11 - Uses 1, 2, 3, 7, 11.

2.7.9.
Requirement ID: 12
Requirement Description: Allow administrators to delete entries in the database
Rationale: This requirement would allow administrators to remove posts from the database that are either malicious and/or irrelevant to the purposes of the website
Inputs:

The administrator user is logged in

The administrator user will go to a post

The admin user will then click the post and select the delete option and be prompted to verify his action

If they proceed, the post will be deleted and the admin user will be brought back to the feed.

If they do not proceed, then the user will be led back to the post screen

Ouputs: The post will be deleted
Persistent Changes: The post and its information will be removed from the database and the database will have one less post.
Related Requirements and Use Cases: Requirement 4, 5, 12
Test Cases: 5

2.7.10.
Requirement ID: 13
Requirement Description: Allow administrators to delete user accounts
Rationale: The requirement would allow administrators to remove users from the product if they infringe on any legal/copyright laws, are disorderly, or are misusing/abusing the service.
Inputs:

The administrator user is logged in

The administrator user will go to a user's information

The admin user will then click the user and select the "Close Account" option and be prompted to verity his action

If they proceed, the user account will be removed and the admin user will be brought back to the feed.

If they do not proceed, then the user will be led back to the user's profile page

Outputs: The user account will be deactivated
Persistent Changes: The user account and its information will be removed from the database and the database will have one less post
Related Requirements and Use Cases: Requirement 4, 5, 12
Test Cases: 5

3. System Modeling
   3.1. Outline Plan

For the Requirements and System Modeling of this product, we first must reformat our requirements to make them more specific and detailed for the use cases. Then, after the requirements have been reformatted, we will create our use cases based on the requirements with details to make the object modeling and test case identification easier. After all the use cases have been completed, we will create the test cases and object models simultaneously as neither one depends on the other. And finally, after the object modeling is finished, we will create the behavior models.

The most serious challenge we see is allowing users to identify animals and query the database for specific animals. A risk for this project that we have already encountered is having the idea fail. We had an idea before this and realized that we could not expand on ideas that already existed. We had to start from scratch and create a new idea which put us behind schedule and made us move more quickly to catch up. Another risk is that the scope of our project may be too large. We may have to scale back our project and start with only local shelters, small communities, and small amounts of data. Then, over time, the software will grow, and we could add support for larger areas, interact with more shelters and identify animals in more regions.

3.2. Schedule

3.2.1. Reporting and Monitoring

Updates will be provided halfway through the duration of the task. Then, they more updates will be given based on status and completion of the task.

3.2.2. Tasks

Task	Effort	Duration	Dependencies	People
Requirements (T1)	1 person-day	1 day	None	Jason
Use Cases (T2)	10 person-days	2 days	T1	Jason, Braxton, Brian, Chris, Nishant
Test Cases (T3)	4 person-days	1 day	T1, T2	Braxton, Brian, Chris, Nishant
Object Modeling (T4)	1 person-day	1 day	T2	Jason
Behavioral Modeling	1 person-day	1 day	T2	Jason

3.3. Objects

3.3.1. UserAccount
3.3.2. OrgAccount
3.3.3. AdminAccount
3.3.4. Animal
3.3.5. Post
3.3.6. Advertisement

3.4. Relationships

- 3.4.1. OrgAccounts is a UserAccount (Generalization)
- 3.4.2. AdminAccounts is a UserAccount (Generalization)
- 3.4.3. Advertisment is a Post (Generalization)
- 3.4.4. UserAccounts create Posts (Aggregation)
- 3.4.5. UserAccounts describe Animals (Aggregation)
- 3.4.6. AdminAccounts verify Posts
- 3.4.7. AdminAccounts verify Animals
- 3.4.8. AdminAccounts verify UserAccounts
- 3.4.9. AdminAccounts verify OrgAccounts
- 3.4.10. OrgAccounts create Advertisments (Aggregation)
- 3.4.11. Posts contain Animals (Aggregation)

3.5. Multiplicity

- 3.5.1. OrgAccount: 1, UserAccount: 1
- 3.5.2. AdminAccount: 1, UserAccount: 1
- 3.5.3. Advertisement: 1, Post: 1
- 3.5.4. UserAccount: 1, Post: 1..*
- 3.5.5. UserAccount: 1, Animal: 1..*
- 3.5.6. AdminAccount: 1..*, Post: 1..*
- 3.5.7. AdminAccount: 1..*, Animal: 1..*
- 3.5.8. AdminAccount: 1..*, UserAccount: 1..*
- 3.5.9. AdminAccount: 1..*, OrgAccount: 1..*
- 3.5.10. OrgAccount: 1, Advertisement: 1..*
- 3.5.11. Post: 1, Animal: 1..*

3.6. Attributes

- 3.6.1. UserAccount
  - 3.6.1.1. name (String)
  - 3.6.1.2. email (String)
  - 3.6.1.3. username (String)
  - 3.6.1.4. password (Hashed String)
  - 3.6.1.5. geolocation (Location)
  - 3.6.1.6. associatedPosts (Post Set)
  - 3.6.1.7. associatedAnimals (Animal Set)
  - 3.6.1.8. timeCreated (long)
- 3.6.2. OrgAccount
  - 3.6.2.1. name (String)
  - 3.6.2.2. email (String)
  - 3.6.2.3. username (String)
  - 3.6.2.4. password (Hashed String)
  - 3.6.2.5. geolocation (Location)
  - 3.6.2.6. associatedPosts (Post Set)
  - 3.6.2.7. associatedAnimals (Post Set)
  - 3.6.2.8. timeCreated (long)
  - 3.6.2.9. associatedAds (Ad Set)

- 3.6.3. AdminAccount
  - 3.6.3.1. name (String)
  - 3.6.3.2. email (String)
  - 3.6.3.3. username (String)
  - 3.6.3.4. password (Hashed String)
  - 3.6.3.5. geolocation
  - 3.6.3.6. associatedPosts (Post Set)
  - 3.6.3.7. associatedAnimals (Animal Set)
  - 3.6.3.8. timeCreated (long)
- 3.6.4. Animal
  - 3.6.4.1. name (String)
  - 3.6.4.2. physicalFeatures (Features Set)
  - 3.6.4.3. geolocation (
  - 3.6.4.4. associatedUsers (UserAccount Set)
  - 3.6.4.5. timeAdded (long)
- 3.6.5. Post
  - 3.6.5.1. creator (UserAccount)
  - 3.6.5.2. geolocation (Location)
  - 3.6.5.3. associatedAnimals (Animal Set)
  - 3.6.5.4. timeCreated (long)
- 3.6.6. Advertisement
  - 3.6.6.1. creator (UserAccount)
  - 3.6.6.2. geolocation (Location)
  - 3.6.6.3. associatedAnimals (Animal Set)
  - 3.6.6.4. timeCreated (long)

3.7. Operations

- 3.7.1. UserAccount
  - 3.7.1.1. new( ) (void)
  - 3.7.1.2. checkGeolocation( ) (Location)
  - 3.7.1.3. addAnimal( ) (void)
  - 3.7.1.4. createPost( ) (void)
  - 3.7.1.5. sharePost( ) (Post)
  - 3.7.1.6. findAnimal( ) (Animal Set)
  - 3.7.1.7. displayPosts( ) (Post Set)
- 3.7.2. OrgAccount
  - 3.7.2.1. new( )
  - 3.7.2.2. checkGeolocation( ) (Location)
  - 3.7.2.3. addAnimal( ) (void)
  - 3.7.2.4. createPost( ) (void)
  - 3.7.2.5. sharePost( ) (Post)
  - 3.7.2.6. findAnimal( ) (Animal Set)
  - 3.7.2.7. displayPosts( ) (Post Set)
  - 3.7.2.8. createAdvertisement( ) (void)

3.7.2.9. collectDatasheet( ) (Post Set)

3.7.3. AdminAccount

3.7.3.1. new( ) (void)

3.7.3.2. checkGeolocation( ) (Location)

3.7.3.3. addAnimal( ) (void)

3.7.3.4. createPost( ) (void)

3.7.3.5. sharePost( ) (Post)

3.7.3.6. findAnimal( ) (Animal Set)

3.7.3.7. displayPosts( ) (Post Set)

3.7.3.8. removeUser( ) (UserAccount)

3.7.3.9. removeAnimal( ) (Animal)

3.7.3.10. removePost( ) (Post)

3.7.3.11. removeAdvertisement( ) (Advertisement)

3.7.4. Animal

3.7.4.1. new( ) (void)

3.7.4.2. addFeatures( ) (void)

3.7.4.3. addAssociation( ) (void)

3.7.5. Post

3.7.5.1. new( ) (void)

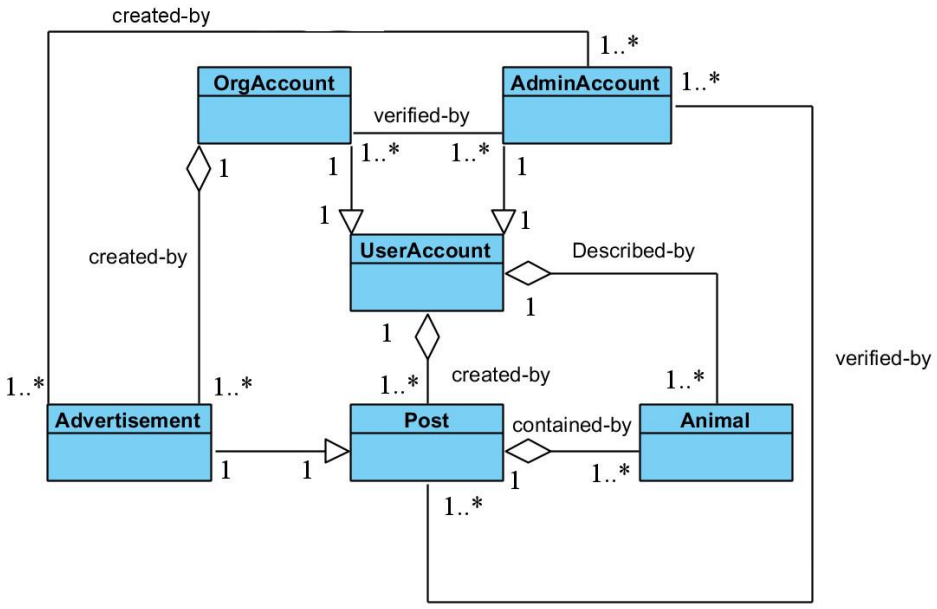
3.7.5.2. createLink( ) (void)

3.7.6. Advertisement

3.7.6.1. new( ) (void)

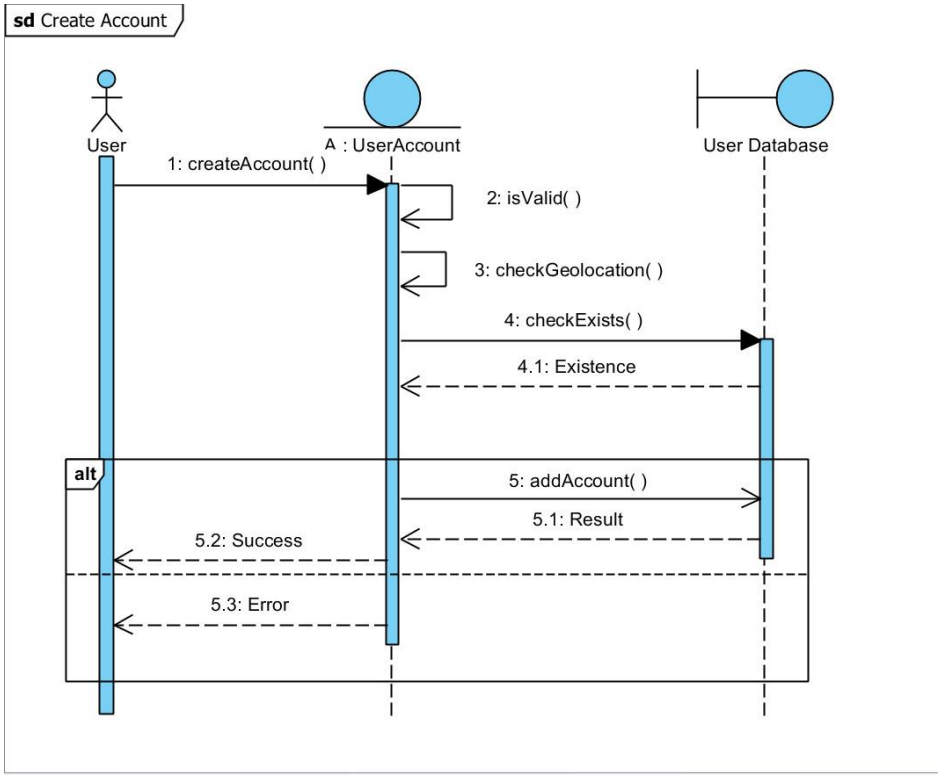
3.7.6.2. createLink( ) (void)

3.8. Class Diagram

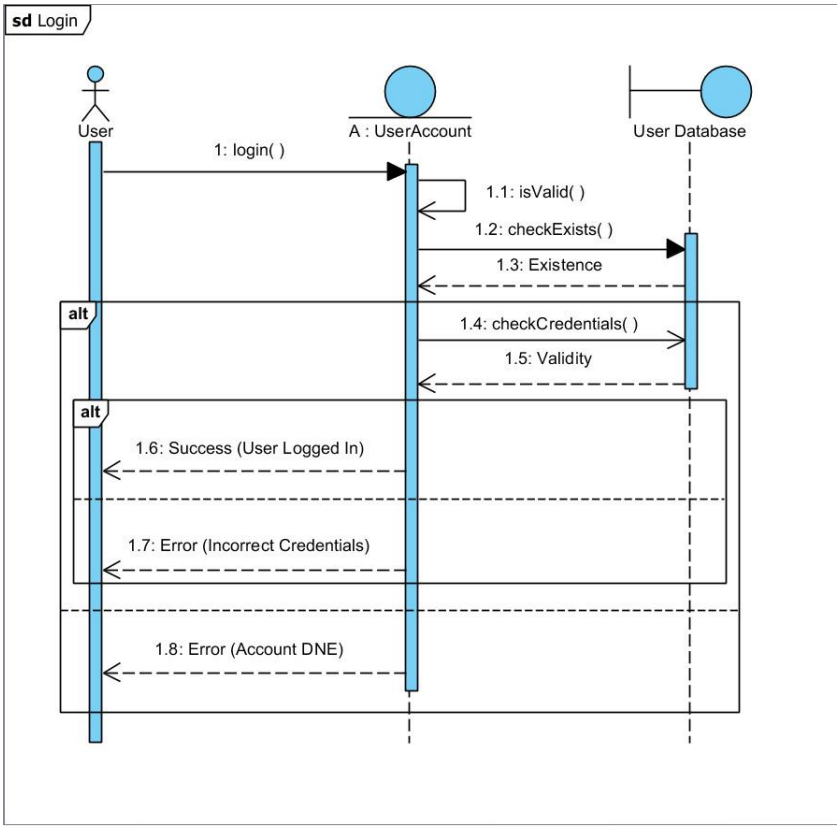


3.9. Sequence Diagrams

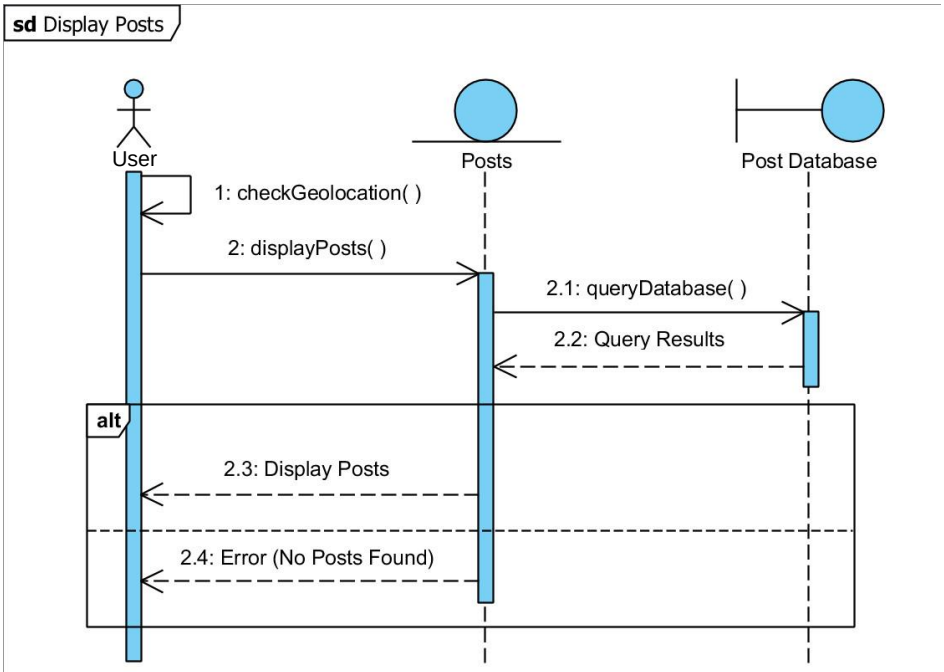
3.9.1. Create Account



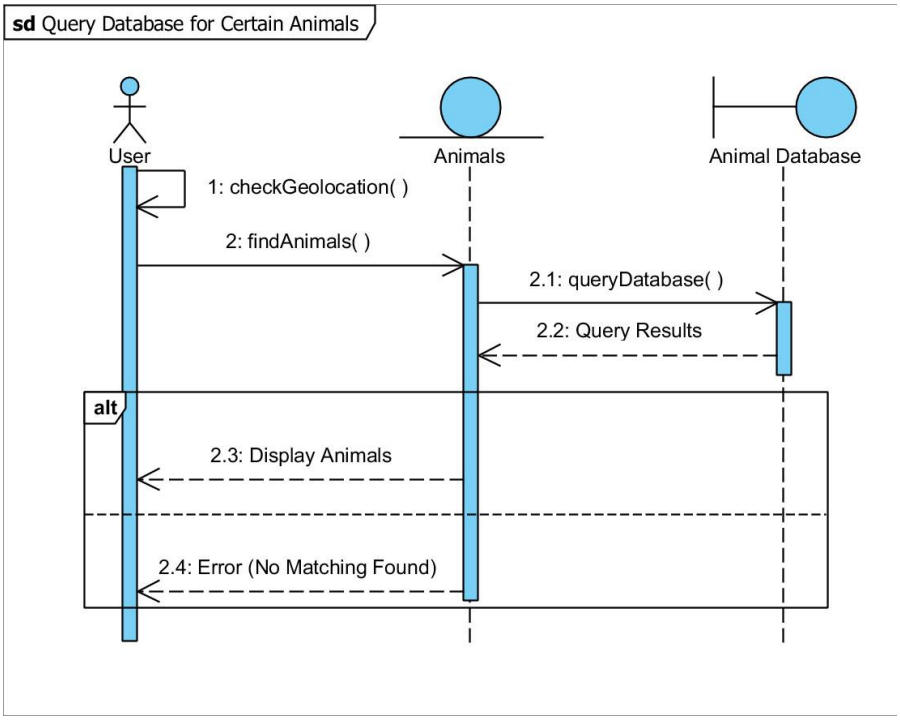
3.9.2. Login



3.9.3. Display Posts



3.9.4. Query Database



4. Architectural Design

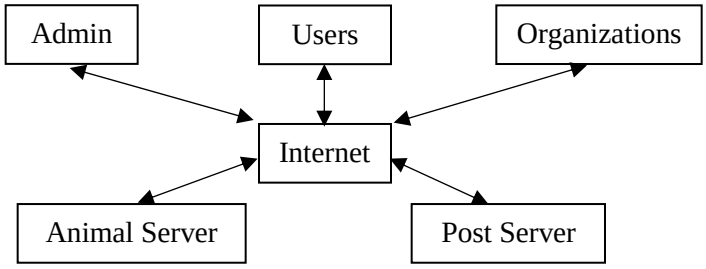
4.1. Outline Plan

For our architectural design, we must first decide how we are going to implement the system. We are currently deciding between using a web-based system and Heroku databases and a Java system and MySQL databases. Depending on which implementation style we choose, we will then choose the architecture design of our system. Then, the last thing to do for architecture design is to make a diagram for the architecture we use.

4.2. Schedule

Task	Effort	Duration	Dependencies	People
Decide Implementation (T1)	1 person-day	<1 day	None	Jason, Braxton, Brian, Chris, Nishant
Decide Architecture Design (T2)	1 person-day	<1 day	T1	Jason, Braxton, Chris, Brian, Nishant
Create Diagram (T3)	1 person-day	1 day	T1, T2	Jason

4.3. Architecture Design (Client-Server Diagram)



5. Implementation

5.1. Outline Plan

The first thing we will do in implementation is decide which use cases we will focus on implementing first. After that, we will have two people work on each use case until both are completed.

5.2. Schedule

Task	Effort	Duration	Dependencies	People
Choose Use Cases (T1)	1 person-day	<1 day	None	Jason, Chris, Braxton, Brian, Nishant
Implement Use Case 1	6 person-days	3 days	T1	Brian, Nishant
Implement Use Case 2	6 person-days	3 days	T1	Braxton, Chris

5.3. How to Use

Watch Implementation Video.

Appendix

Project Description Video:

https://www.youtube.com/channel/UCm0MVS3hisUg1tg7LhFCrtQ?view_as=subscriber

GitHub:

```
commit 2154f9bc7268ade055da35bb5390b6f8030f6ca1 (HEAD -> master, origin/master,
origin/HEAD)
Merge: ce6eb93 30c767c
Author: sonicfangs <briancabigon@gmail.com>
Date:   Sat Mar 10 20:06:04 2018 -0500

    Merge branch 'master' of https://github.com/JasonHerrera/Software-Engineering-
Project

commit ce6eb933f89c826b07e30222259deffa18b94a53
Author: sonicfangs <briancabigon@gmail.com>
Date:   Sat Mar 10 20:05:48 2018 -0500

    Working on authentication

commit 30c767c9536610608bc5eb7e6bccc188d42bfb07
Author: Jason Herrera <jasonrh4513@bellsouth.net>
Date:   Fri Mar 9 20:44:44 2018 -0500

    Separate Directory for Documentation

commit fbfbb65fadfb5c3c722c0418ede6fc614edfcb43
Author: sonicfangs <briancabigon@gmail.com>
```

Slack:

https://github.com/JasonHerrera/Software-Engineering-Project/blob/master/Proj_Docs/Assignment4/Renegon%20Bits%20Slack%20export%20Mar%2010%202018.zip