**Project FeederWatch**

Software Requirements Specification

|  |  |
| --- | --- |
| **Name** | **Student ID** |
| Hazan Anayurt | 2098747 |
| Sezai Artun Özyeğin | 2036515 |

# CHANGE HISTORY

|  |  |
| --- | --- |
| **Version** | **Date** |
| 1.0 | 20.03.2017 |
| 1.1 | 27.03.2017 |

**TABLE OF CONTENTS**

1. [Introduction 5](#_bookmark0)
   1. [Purpose 5](#_bookmark1)
   2. [Scope 5](#_bookmark2)
   3. [Product Overview 6](#_bookmark3)
      1. [Product perspective 6](#_bookmark4)
         1. [System interfaces 7](#_bookmark5)
         2. [User interfaces 8](#_bookmark6)
         3. [Software interfaces 15](#_bookmark7)
         4. [Communications interfaces 15](#_bookmark8)
         5. [Memory constraints 15](#_bookmark9)
         6. [Operations 15](#_bookmark10)
      2. [Product functions 17](#_bookmark11)
      3. [User characteristics 18](#_bookmark12)
      4. [Limitations 19](#_bookmark13)
   4. [Definitions 20](#_bookmark14)
2. [References 21](#_bookmark15)
3. [Specific Requirements 22](#_bookmark16)
   1. [External interfaces 22](#_bookmark17)
   2. [Functions 27](#_bookmark18)
   3. [Usability requirements 45](#_bookmark19)
   4. [Performance requirements 45](#_bookmark20)
   5. [Logical database requirements 46](#_bookmark21)
   6. [Design constraints 47](#_bookmark22)
   7. [Software system attributes 48](#_bookmark23)
   8. [Supporting information 49](#_bookmark24)
4. [Verification. 49](#_bookmark25)
5. [Appendices 49](#_bookmark26)
   1. [Assumptions and dependencies 49](#_bookmark27)
   2. [Acronyms and abbreviations 49](#_bookmark28)

**FIGURES**

Figure 1: Context model… 6

Figure 2: Main page of the FeederWatch website 8

Figure 3: Tally sheet submission interface 10

Figure 4: Photo browsing and uploading interface 11

Figure 5: Explore tab on FeederWatch website 12

Figure 6: Filtering trend graphs by region 12

Figure 7: Distribution trends on map shown according to years. 13

Figure 8: Trend graphs according to years 13

Figure 9: Posting a comment to a blog post 14

Figure 10: FeederWatch ID confirmation 14

Figure 11: External Interfaces class diagram. 22

Figure 12: Login sequence diagram 26

Figure 13: Use case diagram 27

Figure 14: Send research kit order sequence diagram 29

Figure 15: Confirm FeederWatch ID sequence diagram. 30

Figure 16: Logical Database class diagram 46

# TABLES

Table 1: Product functions 17

Table 2: Definitions. 20

Table 3: Send research kit order. 28

Table 4: Confirm FeederWatch ID. 30

Table5: View error logs. 31

Table 6: Donate on Cornell Lab 31

Table 7: Donate on Bird Studies Canada 32

Table 8: Post blog entry 32

Table 9: Access data 33

Table 10: View distributions of birds. 34

Table 11: Comment on blog post 34

Table 12: Browse photos. 35

Table 13: Explore common feeder birds. 36

Table 14: View statistical data 36

Table 15: Sign up for newsletter. 37

Table 16: Create or renew a paid account on Cornell Lab 38

Table 17: Create or renew a paid account on Bird Studies Canada 38

Table 18: Get membership 39

Table 19: Gift membership on Cornell Lab 40

Table 20: Gift membership on Bird Studies Canada 40

Table 21: Sign in 41

Table 22: Upload photo 42

Table 23: Manage account 42

Table 24: Enter observation data 43

Table 25: Review observation data 44

Table 26: Add statistical data 44

# Introduction

# Purpose

Started by Cornell Lab of Ornithology, Project FeederWatch will be a web-based platform allowing regular citizens of America and Canada to monitor movements of more than 1000 bird species "just from their backyard", providing very valuable data for scientists who can use these data to document and understand distribution and abundance of birds in North America to obtain important information about several different things including whether a species is going through population decline or not.

# Scope

* + - The system will contain a database where information about staff, data about bird movements provided by users, photos submitted by users and blog posts and comments will be kept.
    - The system will contain a Graphing Tool that data analyzers (scientists) can use to graph the data they have analyzed. The resulting graphs will be able to be put on the website (Explore tab) from within this Graphing part of the system.
    - The system will have a website which non-staff-member users will access the system through, and use all of the functionalities of the system that they are authorized to.
    - There will be a user interface for staff members which will allow them to manage all website functions, access the database of the system, and use the Graphing Tool through.
    - The system will have an interface for communicating with the system of the warehouse where research kits for new participants are kept to notify them to ship out research kits.
    - Keeping the data of user accounts, creating new accounts and managing these accounts is not in the scope of this system. The system will redirect to Cornell Lab website when a user wants to use any of these functionalities.
    - Contacting the bank for payment for memberships are also not in the scope of this system, as creating or renewing accounts are done through Cornell’s system.

# Product Overview

# Product perspective

FeederWatch system is not exactly a part of a larger system but has other systems that it frequently interacts with, namely the system of Cornell Lab of Ornithology and the system of Bird Studies Canada. It also interacts with Google Maps, to be able to show data on a map, and with the system of the warehouse where the Research kits that are sent to new participants are kept, to be able to fill out shipment orders. The account management subsystem of Cornell Lab system is responsible for carrying out all of the user-management-related functionalities of FeederWatch system, therefore the two systems could be said to be interleaved.

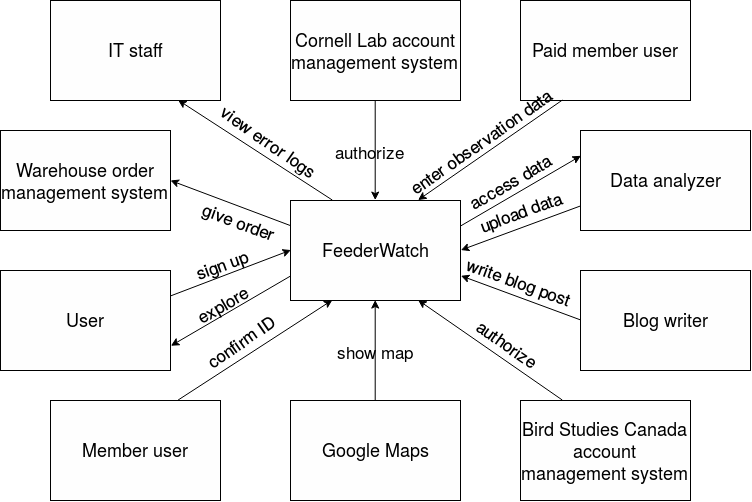


Figure 1: Context model

On the inside, the FeederWatch system can’t be split into subsystems very clearly. However there is a database where raw and processed data about observations, the contents of FeederWatch blog and photos taken by the users are kept, the user interface for non-staff-member users, the user interface for the staff, and the system interfaces which are responsible for communicating with other systems.

# System interfaces

**Warehouse Order Notification Interface:** FeederWatch system communicates with the Warehouse order management system through the Warehouse Order Notification Interface when notified about a new participant by the Cornell Lab account management system. Through the interface, the number of research kit orders and the addresses the orders are going to be sent to are sent to the Warehouse order management system.

**Account Authorization Interface:** A member user needs to enter the FeederWatch ID sent in the research kits to confirm that they are a paid member to be able to use paid member privileges. When this FeederWatch ID is entered to FeederWatch system, it communicates with Cornell Lab account management system or Canada Bird Studies account management system, according to which platform has been selected by the user, through the Account Authorization Interface to check that the FeederWatch ID is in their records and authorize the member user.

**Google Maps API :** FeederWatch system uses Google Maps API to show the distribution of birds on map to users.

**Cornell Login Authorization Interface:** As login is done through Cornell Lab account management system, when a user wants to log into the FeederWatch system, they are redirected to Cornell

Lab website. Then, if Cornell Lab account management system authorizes the user, it redirects the user to FeederWatch system again with a special code parameter. With that code parameter, FeederWatch, when needed, can ask to Cornell Lab whether the user is authorized.

# User interfaces

The user interface of FeederWatch system will be easy to use as older people may also want to observe birds as a “FeederWatcher”. The system has several interfaces for different purposes, which are explained below. More detailed information with visuals will be provided in External Interfaces Section (3.1). Below is a look at the topmost level of the website.



Figure 2: Main page of the FeederWatch website

**Tally Sheet Submission Interface:** Observers, which are paid member users, will be given a hardcopy of Tally Sheet with the Research kit, which is used for counting birds and recording conditions of the observation. Observers can print out more Tally Sheet from the soft copy that is provided on the website. During the observation, the Tally Sheets should be used to record the data.

After the observation, the users will enter the information to the database by filling the form on the website.

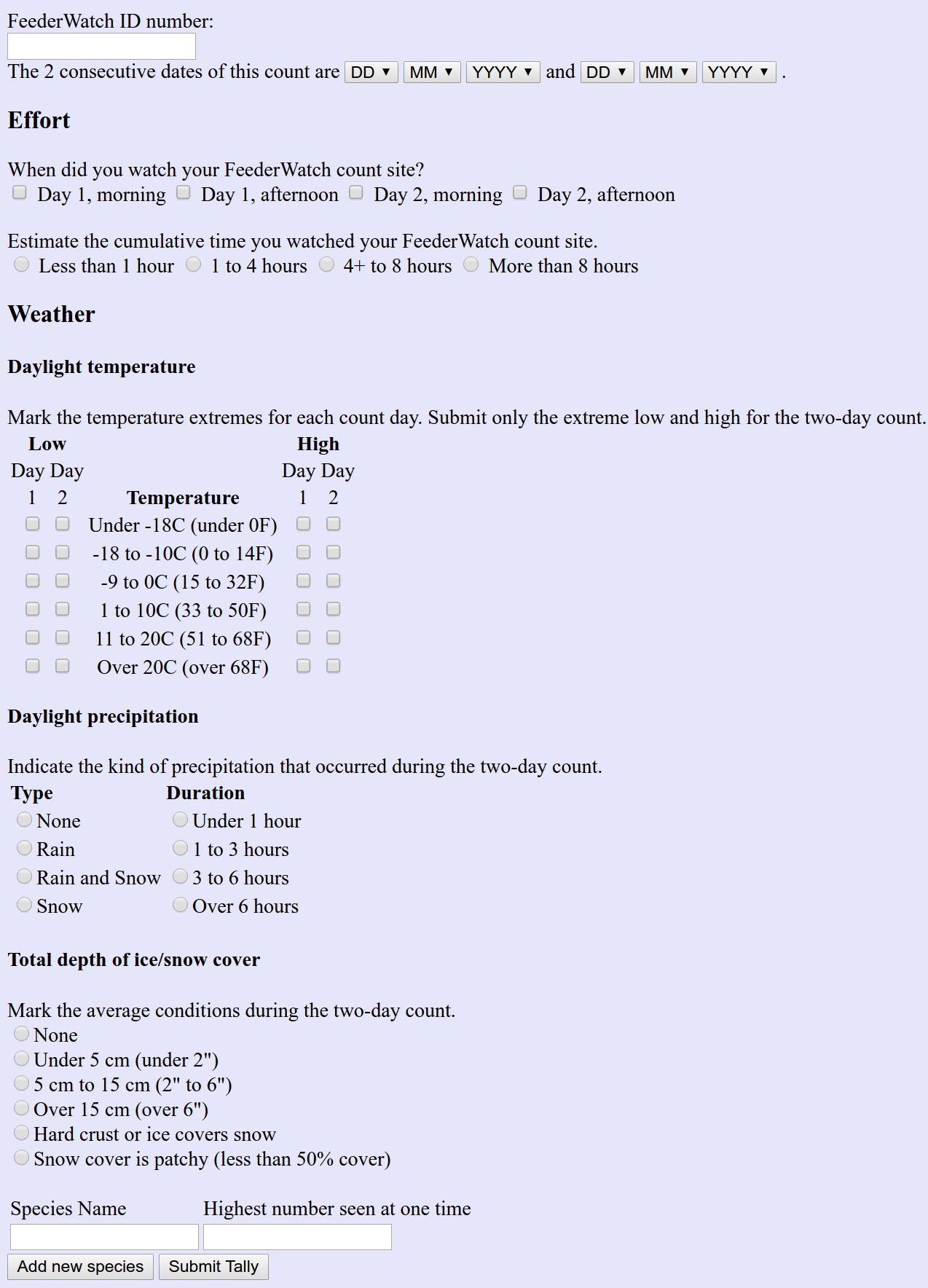


Figure 3: Tally Sheet Submission Interface

**Photo Browsing and Uploading Interface:** Users can upload their photos to FeederWatch website. They can select various

categories while uploading like predatory birds, people, other… These uploaded photos, then, can be browsed by other users. While browsing, users can filter photos according to various fields.

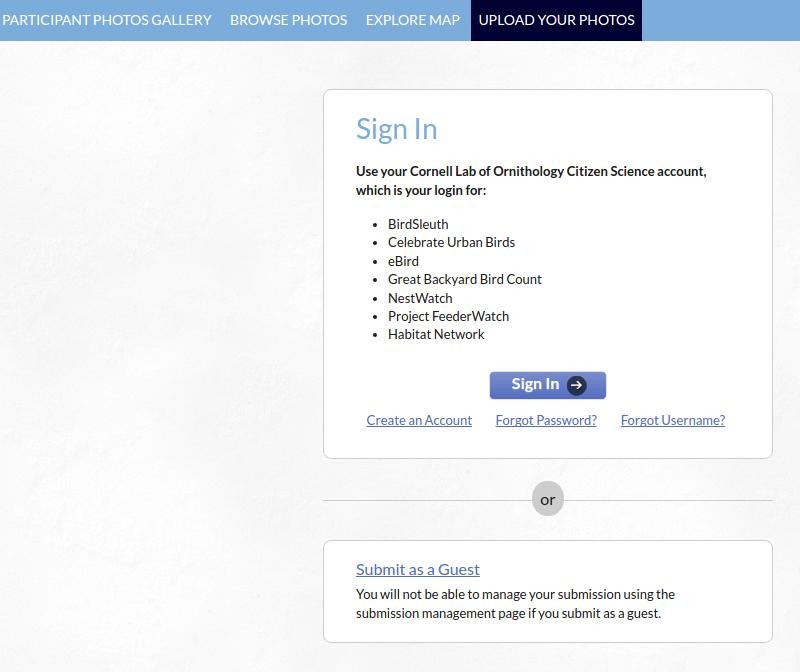


Figure 4: Photo Browsing and Uploading Interface

**Data Access and Analysis Interface:** This interface can only be reached by data analyzers. Through this interface data analyzers can get the raw statistical data they need in the form of a database file and can upload the data they interpreted for the Explore tab on FeederWatch website.

**Admin Panel Interface:** This is where IT staff can see error logs, reach communication information of other staff members, extract raw data to send users who request it, query the FeederWatch database and make changes on the system.

**Statistical Data Exploration Interface (Explore tab):** Through this interface, users can see visualization of the interpreted statistical data by the means of graphs or maps. In Explore tab, there are several visualization options for user to select. They can filter each type of data format according to different fields.



Figure 5: Explore tab on FeederWatch website

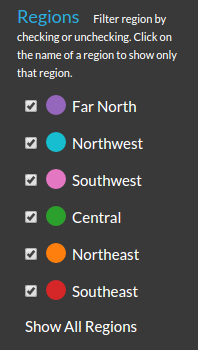


Figure 6: Filtering trend graphs by region

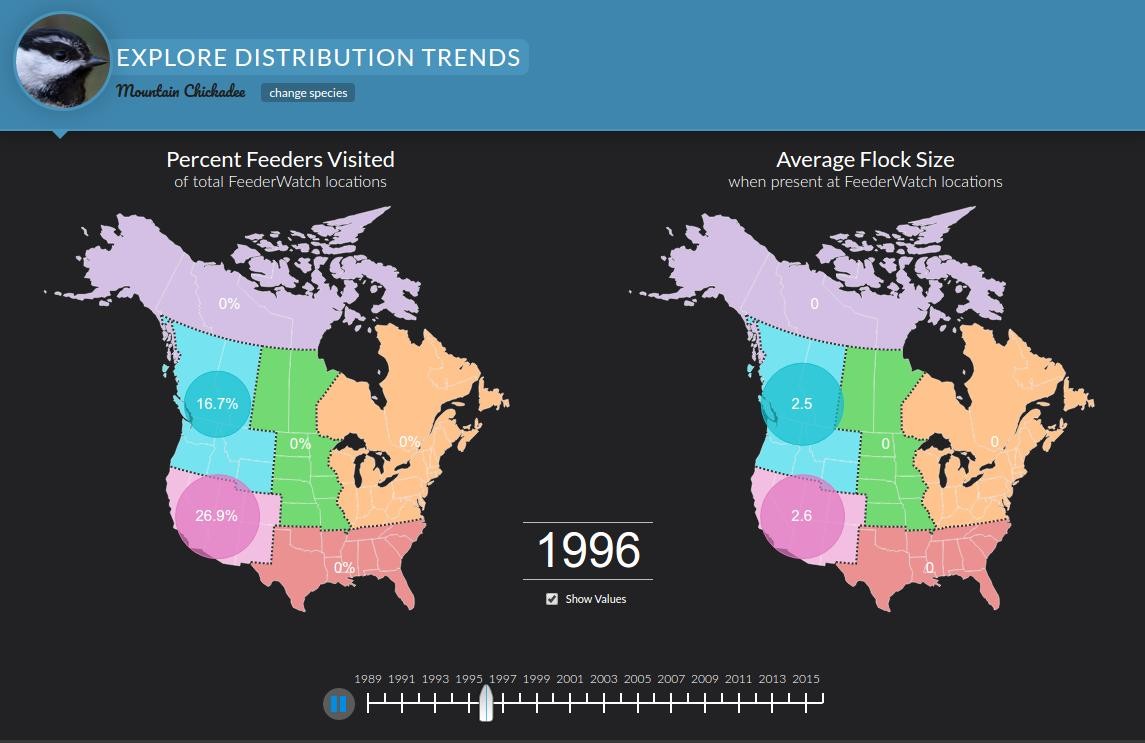


Figure 7: Distribution trends on map shown according to years

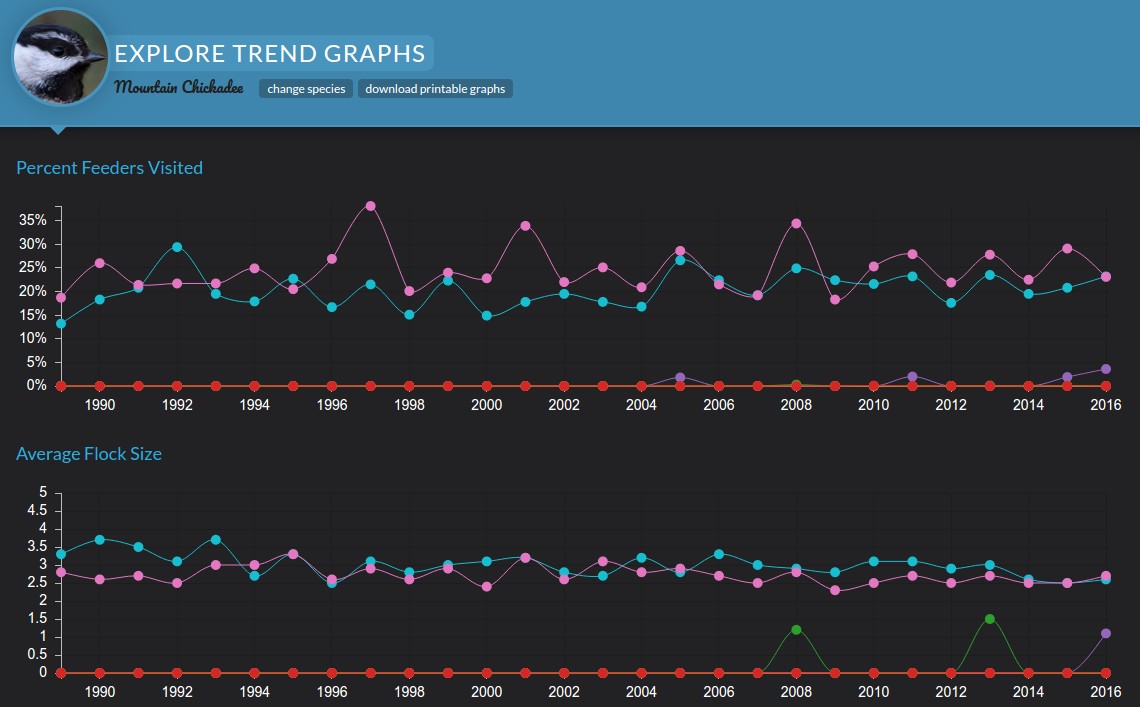


Figure 8: Trend graphs according to year

**Blog Interface:** This is a simple blog interface where Blog writers can post their articles and Users can view and comment them.

Users can filter posts by their publishing month and categories.



Figure 9: Posting a comment to a blog post

**FeederWatch ID Confirmation Interface:** If a user is logged in but hasn’t entered their FeederWatch ID to the system to authenticate their paid membership yet, in the Your Data tab they will be met by a screen prompting them to enter the FeederWatch ID they received in the research kit to get full paid member privileges.

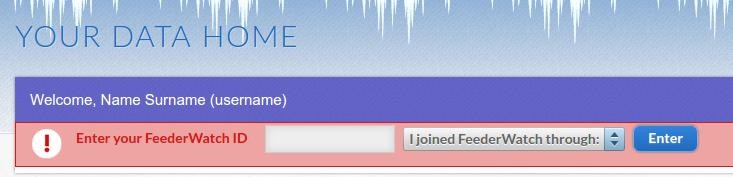


Figure 10: FeederWatch ID confirmation

# Software interfaces

**DBMS:** FeederWatch system uses a database to store data about birds, observations (raw data), photos and the data interpreted by data analyzers.

**Graph Drawing Software:** This software is used for drawing graphical content for the user from the interpreted data stored on the database.

**Operating System:** Operating system will be one of the linux based distributions.

# Communications interfaces

The FeederWatch site will use HTTP. HTTPS protocol is not needed in the website side since security related for account management are done by Cornell Lab account management system and Bird Studies Canada account management system. Notification of warehouse are done by the protocol of their interface.

# Memory constraints

Memory constraints are not a big issue of Project FeederWatch. A memory size capable of running linux and executing basic database and website operations is sufficient.

# Operations

The operations provided by FeederWatch system can be categorised as:

## Staff member operations:

* + - * + View error logs (only for IT staff)
        + Add statistical data (only for data analyzers)
        + Access data (only for data analyzers)

## Paid member user operations:

* + - * + Enter observation data
        + Review observation data

## Member user operations:

* + - * + Manage account
        + Confirm FeederWatch ID
        + Post blog entry (only for users that are blog writers)

## User operations:

* + - * + Comment on blog post
        + View distribution of birds
        + Browse photos
        + Upload photos
        + Explore common feeder birds
        + View statistical data
        + Donate
        + Create or renew paid membership
        + Gift membership
        + Sign up
        + Sign in
        + Sign up for newsletter

## System operations:

* + - * + Send research kit order

All will be elaborated on further in Functions Section (3.2).

# Product functions

Functions (use cases) and summaries of what they do are written below. More detailed and formal explanations of use cases with complete description tables can be found in Functions Section (3.2).

|  |  |
| --- | --- |
| **Function** | **Summary** |
| View error logs | Shows error logs to IT staff through staff member UI |
| Add statistical data | Lets data analyzer post a graph of processed data to website |
| Access data | Makes a database file with raw data available for download for data analyzer |
| Enter observation data | Lets paid user enter and submit observation data |
| Review observation data | Lets paid user review previously submitted observation data |
| Manage account | Redirects member user to Cornell Lab website to change or add user information |
| Confirm FeederWatch ID | Lets member user enter FeederWatch ID that came with research kit to become authorized as paid member user |
| Post blog entry | Lets blog writer (a special type or member user) to enter a new blog post through Our Blog section of Community tab on website |
| Comment on blog post | Lets user comment on a blog post through Our Blog section of Community tab on website |
| View distribution of birds | Shows user current and previous distributions of birds on map through Map Room section of Explore tab on website |
| Browse photos | Shows user photos previously uploaded by other users through Participant Photos section of Community tab on website |
| Upload photos | Lets user upload photos through Participant Photos section of Community tab on website |
| Explore common feeder birds | Lets user learn about common feeder birds through Common Feeder Birds Interactive section of Learn tab on website |
| View statistical data | Shows user statistical data that were processed by data analyzers on graphs and reports through Explore tab on website |
| Donate (on Cornell Lab | Redirects to Cornell Lab or Bird Studies Canada website to |

|  |  |
| --- | --- |
| or on Bird Studies Canada) | donate |
| Create or renew paid membership (on Cornell Lab or on Bird Studies Canada) | Redirects to Cornell Lab or Bird Studies Canada website to create new paid account or renew paid membership |
| Gift membership (on Cornell Lab or on Bird Studies Canada) | Redirects to Cornell Lab or Bird Studies Canada website to gift paid membership to another person |
| Get membership | Redirects to Cornell Lab website to create new account |
| Sign in | Redirects to Cornell Lab website to sign in |
| Sign up for newsletter | Signs users up for Cornell Lab and/or FeederWatch newsletters |
| Send research kit order | Notifies warehouse’s system to ship out research kits |

Table 1: Product functions

# User characteristics

The target users of FeederWatch system can be categorized into two as staff members and non-staff-members.

Staff members in general will need to have basic computer skills, as even administratives who don’t deal with the technical side of things will need to operate the simple staff UI. The IT staff will be experts in their fields. Data analyzers should be advanced at statistics and graphing programs.

Non-staff-member users don’t need to have any special skills other than basic computer skills that will allow them to browse the website and submit forms. They need to be an English or French speaker as English and French will be the only languages the website will offer. For paid membership, they should have a credit card as they have to pay for it through the internet and they should live in Canada or the United States as the project is only collecting data from North America.

# Limitations

**Regulatory policies:** Email addresses of the commenters in the blog sections should not be published. Cornell Lab and Bird Studies Canada are going to deal with the account and bank regulations.

**Hardware limitations:** Since time of processing in Project FeederWatch is not a crucial thing, hardware that is sufficient for a DBMS and graphing operations is enough.

**Interfaces to other applications:** Our system design must be compatible with systems such as Google Maps, Cornell Lab account management system, Bird Studies Canada account management system and Warehouse order management system.

**Parallel operation:** Parallelization is not a crucial thing, since operations are basic. The system can be on a single core machine.

**Audit and Control functions:** There are not any audit or control functions since they are not related to FeederWatch system and banking is managed by Cornell Lab and Bird Studies Canada.

**Higher-order language requirements:** Back-end of the system will be written in Java in order to support multiple operating systems, in case of a change in the platform that may happen in the future. While designing and developing the website, css, html and javascript will be used.

**Signal handshake protocols:** HTTPS (TCP, TLS) is used between FeederWatch and account management systems and Warehouse order management systems. Also between users and FeederWatch, HTTP will be used.

**Quality requirements:** Recoverability and reliability of the data of FeederWatch system is the topmost priority. Therefore, HDDs must duplicate their data all the time and backups must be frequent.

**Criticality of application:** Even if the system crashes, it would not cause any loss of life. That's why the system is not critical. Thus, reliability of system, other than preservation of data, doesn't have to be very high.

**Safety and security considerations:** Security of accounts are managed by Cornell Lab account management system and Bird Studies Canada account management system. Other than that, the database should be secure, immune to attacks since it would risk the safety of data.

**Physical/mental considerations:** Anyone that can use a computer, and has a place to observe birds can be a FeederWatcher; as long as they can pay the money for the paid membership.

# Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| User | Website user that isn’t logged in |
| Member user | Website user that is logged in but hasn’t confirmed their FeederWatch ID (either because they haven’t got a paid  membership or they haven’t received their research kit yet) |
| Paid member user | Website user that is logged in and has confirmed their FeederWatch ID |
| Database file | .db file |
| FeederWatcher | A paid member user of FeederWatch |
| JVM | Java Virtual Machine |
| AUTHDATA | Authorization data |
| PID | Package Identification |

Table 2: Definitions

# References

## This document is written with respect to the specifications of the document below:

29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering

-- Life cycle processes --Requirements engineering. (2013, January 24). Retrieved March 15, 2016, from https://standards.ieee.org/findstds/standard/29148-2011.html

## Other sources:

Dong Ngo March 29, 2013 1:02 PM PDT @riceandstirfry. (2012, December 03). Digital storage basics, Part 3: Backup vs. redundancy. Retrieved March 23, 2017, from https://[www.cnet.com/how-to/digital-storage-basics-part-3-backup-vs-redundancy/](http://www.cnet.com/how-to/digital-storage-basics-part-3-backup-vs-redundancy/)

Sommerville, I. (2016). Software engineering. Boston: Pearson Education Limited.

# Specific Requirements

# External interfaces

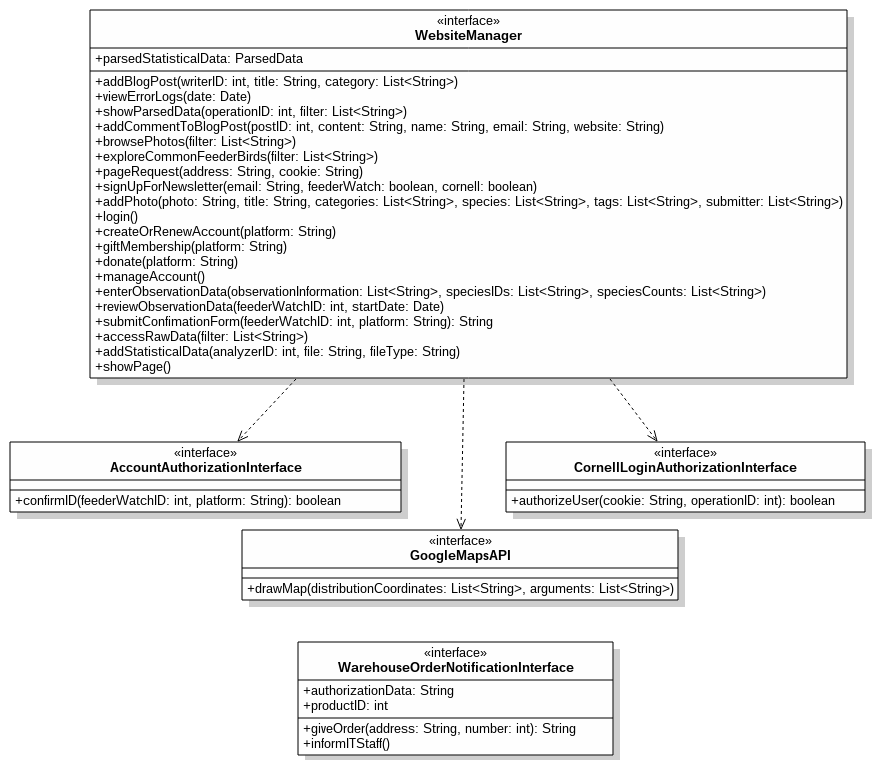


Figure 11: External Interfaces Class Diagram

# Tally Sheet Submission Interface

**req01:** If any data is missing in the submitted form, the interface shall show the user the data they have already entered while prompting the user to enter the missing.

**req02:** If the dates the user entered aren’t consequent, the interface shall warn the user and prompt the user to either correct if

they entered wrong values or to exit without submitting data if they didn’t make an observation in two consequent days.

**req03:** If the submitted data is correct and complete, the interface shall send the data collected to DBMS.

# Photo Browsing and Uploading Interface

**req04:** Whether a photo adheres the guidelines shall be checked upon submission. If the photo isn’t appropriate, the user shall be warned and the photo shall not be posted.

**req05:** The options to post as member user or post as guest shall be separate.

**req06:** While browsing the photos, the user shall be able to filter photos by activity, by category and by year.

**req07:** The information of the name of the photographer and where they are from shall be shown.

# Data Access and Analysis Interface

**req08:** The data analyzer shall be able to request raw data from the database through “Request Raw Data” button, filtering which tables or columns they want.

**req09:** The interface shall make the data available for download in a .db file through “Download Data” button upon request of data analyzer.

**req10:** The data analyzer shall be able to upload processed data to system through “Upload Processed Data” button. The uploaded data shall be in a file format that is parseable by the system.

# Admin Panel Interface

**req11:**

**req12:** To be able to send the raw data to a user, IT staff shall be able to access raw data through “Extract raw data” button. The raw data shall be able to be filtered before extraction, according to the user’s request.

**req13:** Database shall be queried through the Admin Panel Interface.

* + 1. **Statistical Data Exploration Interface (Explore tab) req14:** Visualizations such as graphs and maps generated by using Graph Drawing Software shall be available for users to see under different sections under the “Explore” tab on website.

**req15:** User shall be able to see “Trend Graphs” by first selecting the species of the bird and selecting the year after that.

**req16:** User shall be able to see the tabulated data according to state and year through “Bird summaries” section.

**req17:** Distribution of a bird species in a selected year shall be shown on a real map using Google Maps API. This option shall be under “Map Room” section.

**req18:** User shall be able to select among the various options of visualizations from the “Explore” tab.

# Blog interface

**req19:** A blog writer shall be able to post a new blog entry by completely filling the parts of the form such as name, surname and also uploading a file which contains the article.

**req20:** While posting a blog entry, Category shall be optional. **req21:** User shall be able to filter the blog posts by their publishing month and categories.

**req22:** User shall be able to post a comment to a blog post by filling Comment, Name and Email and optionally user’s website information.

**req23:** The Email address of the user who commented on a post shall not be published publicly and user shall be informed about this.

# FeederWatch ID Confirmation Interface

**req24:** User shall be already logged in to use this interface. **req25:** To be able to set the authorization level of the account, the FeederWatch ID shall be sent to the Cornell lab account

management system or Bird Studies Canada account management system according to the selection of the user.

**req26:** After submitting the ID the user shall be informed about the response of the account management system.

# Warehouse Order Management Interface

**req27:** Cornell Lab account management system shall be able to use the functionality provided by this interface by requesting FeederWatch system to send an order.

**req28:** When requested, Warehouse Order Management Interface shall inform Warehouse order management system.

**req29:** Cornell Lab account management system shall be informed about the result of the process.

**req30:** If the research kits are not enough to complete the order, Warehouse Order Management Interface shall command the Cornell Lab account management system to send new kits to the Warehouse.

# Account Authorization Interface

**req31:** When the users submit their FeederWatch ID’s, account authorization interface shall pass the ID to Cornell Lab or Bird Studies Canada according to the platform the user selected.

# Google API

**req32:** When user requests to see Map room through Statistical Data Exploration Interface, the map shall be provided by Google Maps API.

# Cornell Login Authorization Interface

**req33:** FeederWatch system shall use this interface whenever the functionality the user wants to use needs authorization.

**req34:** This interface shall use the cookie that was given when the user first logged in to authorize the user afterwards.

**req35:** FeederWatch system shall not store any information about the user but shall get it from Cornell Lab when needed, using this interface.

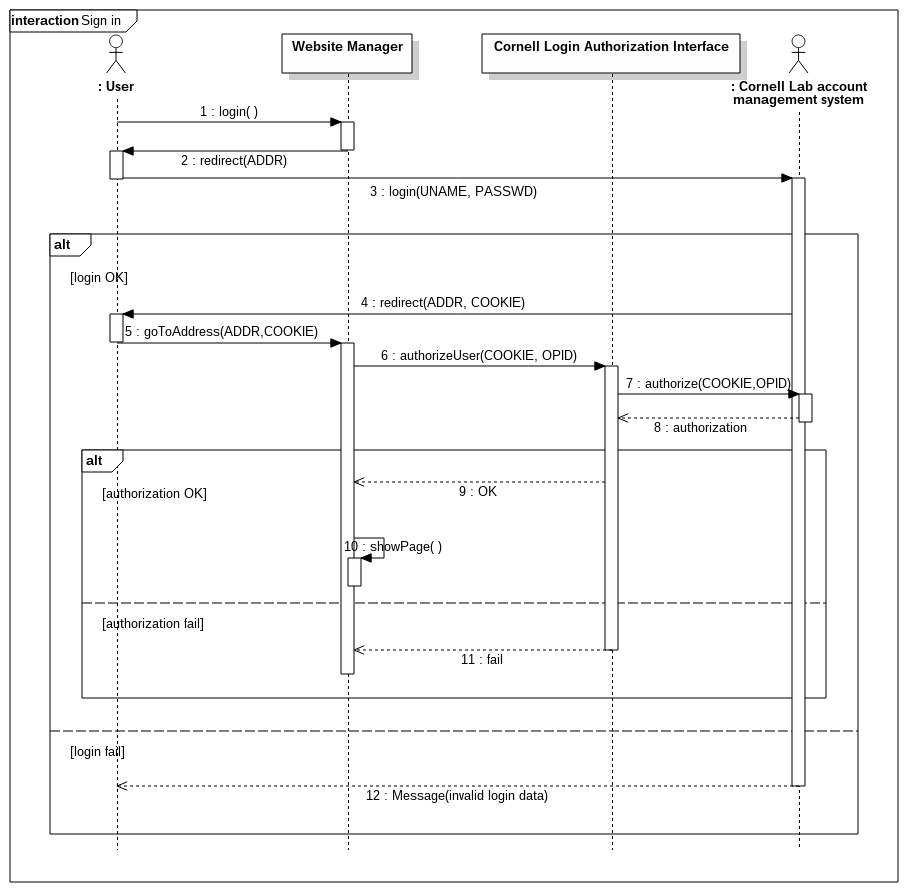


Figure 12: Login Sequence Diagram

# Functions

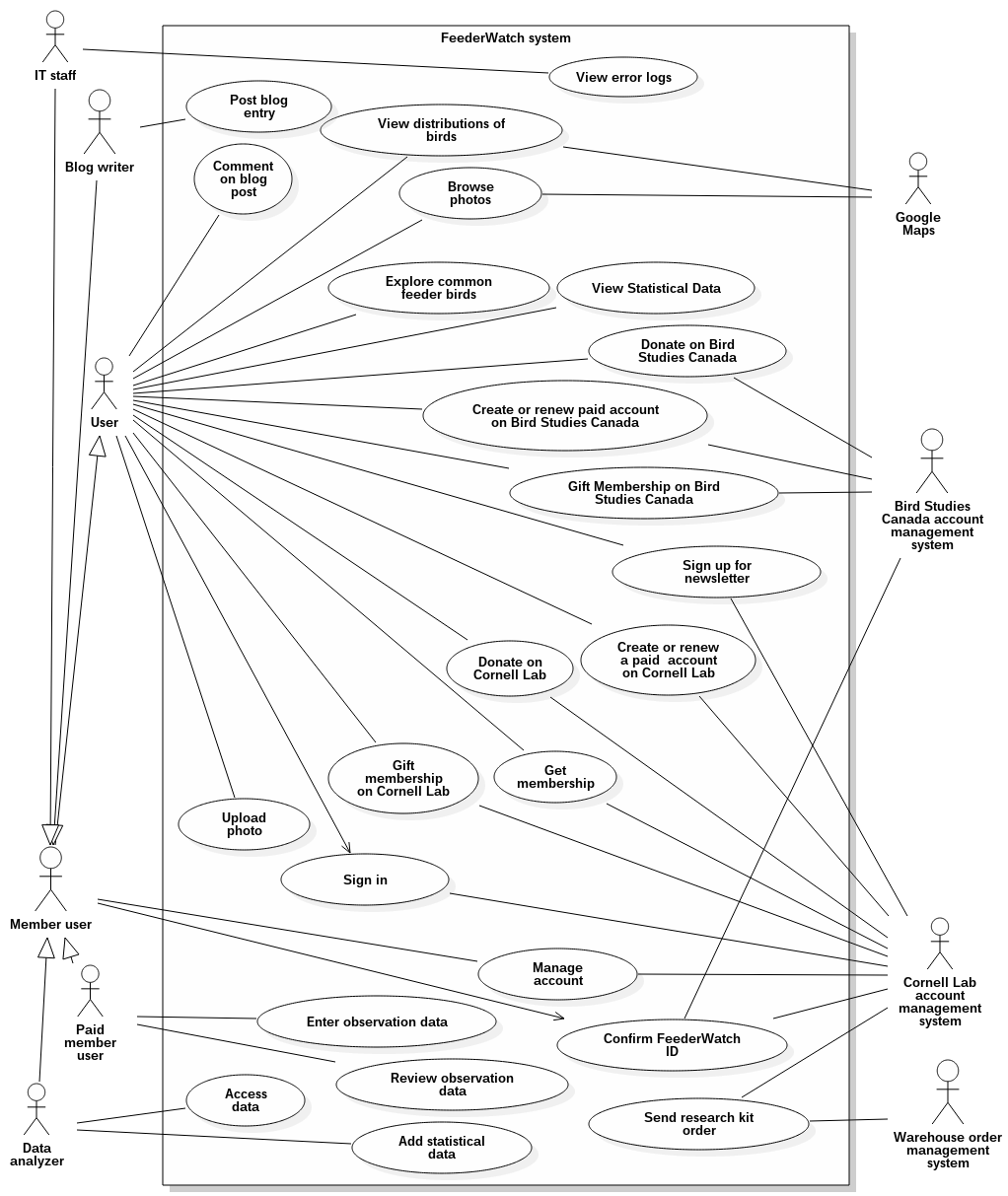


Figure 13: Use Case Diagram

|  |  |
| --- | --- |
| **Use case name** | Send research kit order |
| **Actors** | Cornell Lab account management system, Warehouse order management system |
| **Description** | Research kits are sent to new paid members of the project. These research kits are kept in a warehouse which the account management system alerts to ship out research kits whenever a new person joins the project.  The warehouse then ships out the orders. |
| **Data** | How many research kits are going to be sent, which addresses the research kits are going to be sent to |
| **Preconditions** | - |
| **Stimulus** | Cornell Lab account management system notifies FeederWatch system when someone enters all the required data and clicks join. |
| **Basic Flow** | Step 1 - FeederWatch system is notified by the Cornell Lab account management system.  Step 2 - The address and quantity data received is sent to the Warehouse order management system through FeederWatch Warehouse System Interface.  Step 3 - A confirmation that the kits will be shipped with the next party of shipment from the Warehouse is received. |
| **Alternative Flow** | Step 3 - If there are no kits left at the warehouse, FeederWatch system is notified about this and the shipment is postponed in the Warehouse system to when new kits have arrived.  Step 4 - The Cornell lab is notified and takes care of the delivery of new kits to the Warehouse. |
| **Exception Flow** | If the authorization of FeederWatch system at Warehouse order management system fails, this is saved in Error log file. |
| **Postconditions** | A research kit is shipped out to the new participant. |

Table 3: Send research kit order

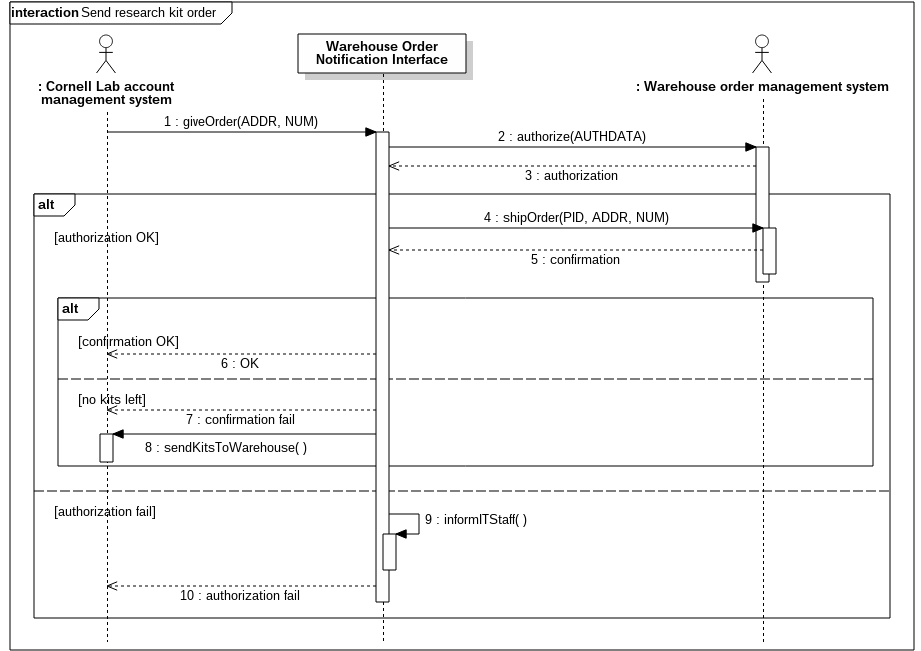


Figure 14: Send research kit order sequence diagram

|  |  |
| --- | --- |
| **Use case name** | Confirm FeederWatch ID |
| **Actors** | Member user, Cornell Lab account management system, Bird Studies Canada account management system |
| **Description** | Member user can enter the FeederWatch ID to get authorized as a paid member user and use paid member privileges. |
| **Data** | FeederWatch ID, which platform the member user joined the system through, authorization data |
| **Preconditions** | Member user should be logged in before confirming FeederWatch ID. |
| **Stimulus** | Member user presses “Enter” button. |
| **Basic Flow** | Step 1 - ID confirmation request is received from the Member user.  Step 2 - If Cornell lab is selected as the platform, and if Cornell Lab account management system authorizes the |

|  |  |
| --- | --- |
|  | Member user, then FeederWatch System authorizes the Member user to be Paid member user. |
| **Alternative Flow** | Step 2 - If Canada Bird Studies is selected as the platform, and if Canada Bird Studies account management system authorizes the Member user, then FeederWatch System authorizes the Member user to be Paid member user. |
| **Exception Flow** | If Cornell Lab account management system or Canada Bird Studies does not authorize the Member user, then show an error message to the Member user. |
| **Postconditions** | Member user may be authorized as Paid member user or not authorized by the Cornell Lab account management system or Canada Bird Studies account management system and an error message is shown. |

Table 4: Confirm FeederWatch ID

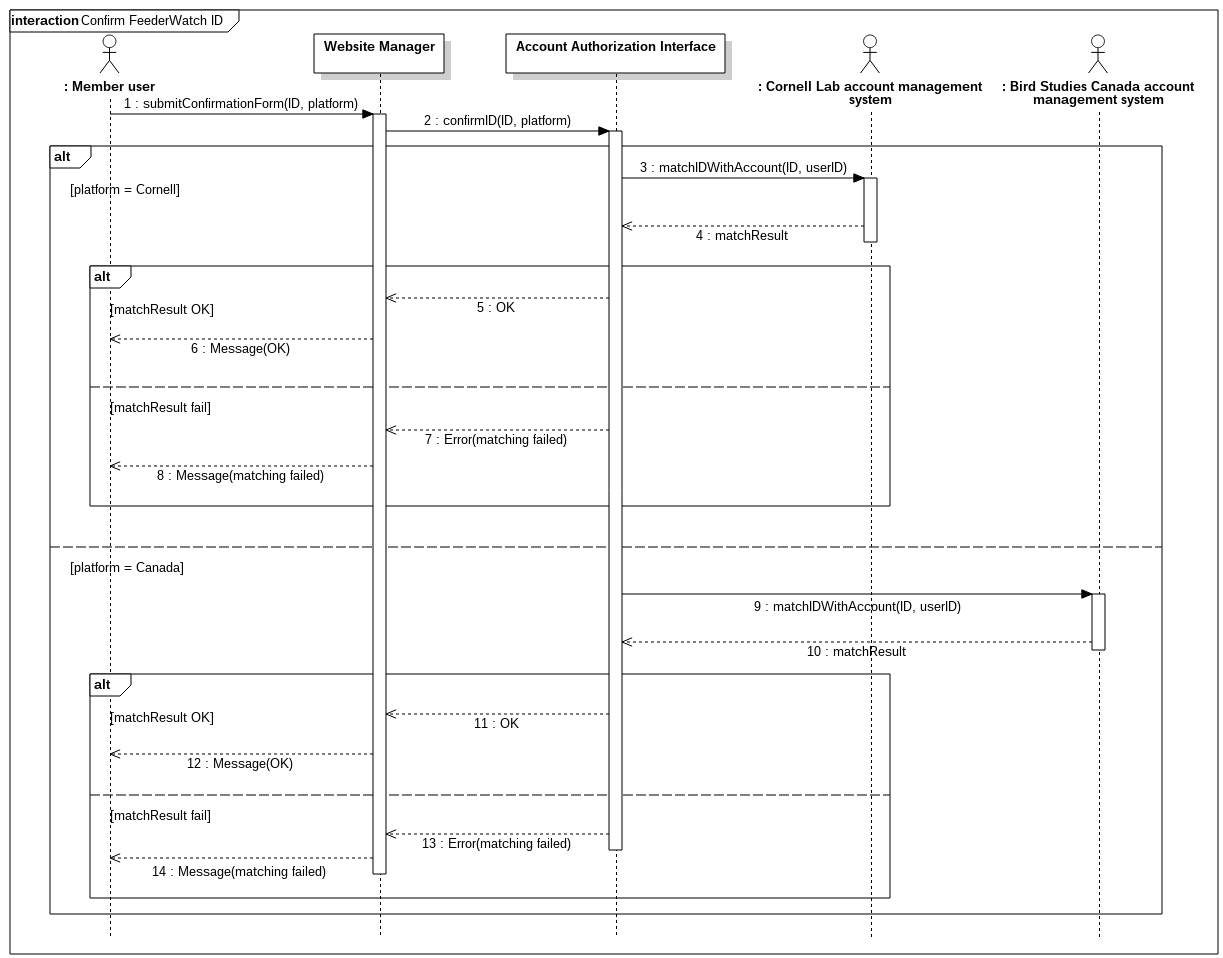


Figure 15: Confirm FeederWatch ID sequence diagram

|  |  |
| --- | --- |
| **Use case name** | View error logs |
| **Actors** | IT staff, Cornell Lab account management system |
| **Description** | IT staff can view the errors logs happened in the FeederWatch system. |
| **Data** | Error logs |
| **Preconditions** | User should be logged in as IT staff. |
| **Stimulus** | IT staff presses “View error logs” button |
| **Basic Flow** | Step 1 - Request is received by FeederWatch system. Step 2 - Error logs are shown to the IT staff on a special screen. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | Error logs are shown on the screen. |

Table 5: View error logs

|  |  |
| --- | --- |
| **Use case name** | Donate on Cornell Lab |
| **Actors** | User, Cornell Lab account management system |
| **Description** | Users in the United States can donate to FeederWatch by using Cornell Lab account management system.  FeederWatch redirects User to Cornell Lab to manage payment. |
| **Data** | - |
| **Preconditions** | - |
| **Stimulus** | User presses the “Donate” button. |
| **Basic Flow** | Step 1 - User clicks Donate button.  Step 2 - FeederWatch system redirects User to Cornell Lab account management system’s donate page. |
| **Alternative Flow** | - |
| **Exception Flow** | - |

|  |  |
| --- | --- |
| **Postconditions** | The user shall be in the Cornell Lab account management system’s donate page. |

Table 6: Donate on Cornell Lab

|  |  |
| --- | --- |
| **Use case name** | Donate on Bird Studies Canada |
| **Actors** | User, Bird Studies Canada account management system |
| **Description** | Users in Canada can donate to FeederWatch by using Bird Studies Canada account management system.  FeederWatch redirects User to Bird Studies Canada to manage payment. |
| **Data** | - |
| **Preconditions** | - |
| **Stimulus** | User presses the “Donate” button. |
| **Basic Flow** | Step 1 - User clicks Donate button.  Step 2 - FeederWatch system redirects User to Bird  Studies Canada account management system’s donate page. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The user shall be in the Bird Studies Canada account management system’s donate page. |

Table 7: Donate on Bird Studies Canada

|  |  |
| --- | --- |
| **Use case name** | Post blog entry |
| **Actors** | Blog writer |
| **Description** | Blog writer is a special member user type who can post their articles to FeederWatch system’s blog. By posting blog entry, their article is shown at the blog of FeederWatch system. |
| **Data** | Post title, Post content |
| **Preconditions** | Blog writer shall be authenticated to write a blog post. Post title shall not be empty. |

|  |  |
| --- | --- |
|  | Exactly one file with proper type should be uploaded as Post content. |
| **Stimulus** | Blog writer presses “Submit” button after uploading post content and writing title. |
| **Basic Flow** | Step 1 - Extract the text from the uploaded post content document.  Step 2 - Insert post title, post content and publisher name to database.  Step 3 - Publish post on FeederWatch blog. |
| **Alternative Flow** | - |
| **Exception Flow** | If the text cannot be extracted from the document, FeederWatch system sends and error to Blog writer. |
| **Postconditions** | The entry which Blog writer posted is saved in the database and published on the site. |

Table 8: Post blog entry

|  |  |
| --- | --- |
| **Use case name** | Access Data |
| **Actors** | Data analyzer |
| **Description** | Through the Access Data functionality of the system, all statistical raw data are available for download for the data analyzers who are responsible for providing processed statistical data for the website’s Explore tab. |
| **Data** | Statistical data previously entered by users |
| **Preconditions** | The user should be logged in as data analyzer to be able to request data access from the system. |
| **Stimulus** | The data analyzer selects which tables they want to access and presses “Get Data” button. |
| **Basic Flow** | Step 1 - The request made by the data analyzer for certain data is received by the system.  Step 2 - The system creates a new database file with the tables the data analyzer requested.  Step 3 - The database file is available for download for the analyzer. |
| **Alternative Flow** | - |

|  |  |
| --- | --- |
| **Exception Flow** | - |
| **Postconditions** | The requested data is available for the data analyzer to download. |

Table 9: Access Data

|  |  |
| --- | --- |
| **Use case name** | View distributions of birds |
| **Actors** | User, Google Maps |
| **Description** | User can see the distribution of a specific bird species according to dates on a map which is provided by Google Maps. |
| **Data** | The name of the bird species, date |
| **Preconditions** | Only valid dates should be presented to User. |
| **Stimulus** | User presses “Go to map” button. |
| **Basic Flow** | Step 1 - Get the name of the bird species, and date from User.  Step 2 - Request the map user wanted using Google Map’s API and show it to user. |
| **Alternative Flow** | - |
| **Exception Flow** | Show an apology message to user if Google Maps is down. |
| **Postconditions** | The map user wanted is shown to the User. |

Table 10: View distributions of birds

|  |  |
| --- | --- |
| **Use case name** | Comment on blog post |
| **Actors** | User |
| **Description** | User can comment on a blog post. |
| **Data** | Comment, Name, Email, and Website of the User |
| **Preconditions** | The blog post that will be commented on should exist in the database. |
| **Stimulus** | User presses “Post Comment” button. |

|  |  |
| --- | --- |
| **Basic Flow** | Step 1 - Comment, Name, Email from User and Post Name, and optionally Website information is received. Step 2 - FeederWatch inserts this information to the database. |
| **Alternative Flow** | **-** |
| **Exception Flow** | If at least one of the Comment, Name and Email sections are not provided, User sees an error message. The sections that were already provided are not erased in the error page and don’t have to be provided again. |
| **Postconditions** | The comment User posted is shown under the Post. |

Table 11: Comment on blog post

|  |  |
| --- | --- |
| **Use case name** | Browse photos |
| **Actors** | User, Google Maps |
| **Description** | User can filter photos that were previously uploaded by Member users. Filtering can be done by Season, by selection on the map provided by Google Maps, by activity, by Popular species, and by Popular tags. |
| **Data** | Filtering type selected by user |
| **Preconditions** | - |
| **Stimulus** | User selects any type of filter. |
| **Basic Flow** | Step 1 - Get the filtering information from the User. Step 2 - If user does not select filtering “On The Map” option, query the database accordingly.  Step 3 - Show the resulting images to User. |
| **Alternative Flow** | Step 2 - If user selects filtering “On the Map” option, the request to the Google Map API is done accordingly.  Step 3 - Show the user resulting area done by the filtering by using Google Map. Also allow for further filtering “On  the Map”. |
| **Exception Flow** | Show an apology message to user if Google Maps is down. |
| **Postconditions** | Filtered photos are shown to the user. |

Table 12: Browse photos

|  |  |
| --- | --- |
| **Use case name** | Explore common feeder birds |
| **Actors** | User |
| **Description** | Explore common feeder birds provides an environment to the newly registered paid users to improve their bird identification skills since they are going to provide FeederWatch data. Also normal Users can access this functionality. The birds can be categorized by their region, food type, and feeder type. |
| **Data** | Region, food type, and feeder type info if filtered, information about birds that will be shown to the User, the photos of the birds |
| **Preconditions** | - |
| **Stimulus** | User accesses the link and also selects a filtering option. |
| **Basic Flow** | Step 1 - Query the database to show all common birds. Step 2 - Show all common birds to the user without any filtering. |
| **Alternative Flow** | Step 2 - If user selects any filtering option, query the database accordingly and show the user the resulting birds. |
| **Exception Flow** | - |
| **Postconditions** | Resulting birds are shown to the user according to the filtering done by the User. |

Table 13: Explore common feeder birds

|  |  |
| --- | --- |
| **Use case name** | View statistical data |
| **Actors** | User |
| **Description** | In the Explore tab of the FeederWatch website, User can view statistical data that is present in the FeederWatch system that were interpreted by Data analyzers. It can be categorized by bird species or by regions and filtering can be selected by the User. |
| **Data** | Filtering options selected by user, statistical data requested. |
| **Preconditions** | - |

|  |  |
| --- | --- |
| **Stimulus** | User requests data by clicking the according links. |
| **Basic Flow** | Step 1 - Give User the option of selecting the category of which data will be shown.  Step 2 - Query the database and show the user results accordingly.  Step 3 - Give User the option of filtering the results. |
| **Alternative Flow** | If User requests to filter the results, again query the database accordingly and show the results to User. |
| **Exception Flow** | - |
| **Postconditions** | User can see the filtered results or all results of the category. |

Table 14: View statistical data

|  |  |
| --- | --- |
| **Use case name** | Sign up for newsletter |
| **Actors** | User, Cornell Lab account management system |
| **Description** | User fills a form in FeederWatch website to subscribe for the newsletter of the Cornell Lab eNews or Project FeederWatch. |
| **Data** | Email address of the user and the selection of User whether User wants to register Cornell Lab eNews or Project FeederWatch. |
| **Preconditions** | - |
| **Stimulus** | User presses “Sign Up” button under the newsletter form. |
| **Basic Flow** | Step 1 - Email address of the user is fetched together with the selection of which newsletter User wanted to subscribe.  Step 2 - Send fetched information to Cornell Lab account management system if the User selected Cornell eNews option. If the User selected “Project FeederWatch” option, then subscribe the user to the newsletter subsystem of FeederWatch.  Step 3 - Show the User a message that shows the subscription process is done without any error. |
| **Alternative Flow** |  |
| **Exception Flow** | If the user does not select any of the newsletter systems |

|  |  |
| --- | --- |
|  | or does not type valid email, then show a related error message to the user. |
| **Postconditions** | User subscription details are sent to Cornell Lab account management system if the option is selected. Likewise, the User is subscribed to Project FeederWatch, if selected. |

Table 15: Sign up for newsletter

|  |  |
| --- | --- |
| **Use case name** | Create or renew a paid account on Cornell Lab |
| **Actors** | User, Cornell Lab account management system |
| **Description** | The User who wants to create a paid account on FeederWatch website and is in the United States will be redirected to Cornell Lab account management system to handle the creation process. |
| **Data** | - |
| **Preconditions** | - |
| **Stimulus** | The User presses the “Join or Renew” button. |
| **Basic Flow** | Step 1 - Joining or renewing request is received from the User.  Step 2 - FeederWatch system redirects User to Cornell Lab account management system’s join or renew page. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The user shall be in the Cornell Lab account management system’s “Join or Renew” page. |

Table 16: Create or renew a paid account on Cornell Lab

|  |  |
| --- | --- |
| **Use case name** | Create or renew a paid account on Bird Studies Canada |
| **Actors** | User, Bird Studies Canada account management system |
| **Description** | The User who wants to create a paid account on FeederWatch website and is in Canada will be redirected to Bird Studies Canada account management system to handle the creation process. |

|  |  |
| --- | --- |
| **Data** | - |
| **Preconditions** | - |
| **Stimulus** | The User presses the “Join or Renew” button. |
| **Basic Flow** | Step 1 - Joining or renewing request is received from the User.  Step 2 - FeederWatch system redirects User to Bird  Studies Canada account management system’s join or renew page. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The user shall be in the Bird Studies Canada account management system’s “Join or Renew” page. |

Table 17: Create or renew a paid account on Bird Studies Canada

|  |  |
| --- | --- |
| **Use case name** | Get membership |
| **Actors** | User, Cornell Lab account management system |
| **Description** | The User who selects to “Create an Account” on FeederWatch website will be redirected to Cornell Lab account management system to handle the creation process. |
| **Data** | - |
| **Preconditions** | - |
| **Stimulus** | The User presses the “Create an Account” button. |
| **Basic Flow** | Step 1 - User clicks “Creating account request is received from the User.  Step 2 - FeederWatch system redirects User to Cornell Lab account management system’s account creation page. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The user shall be in the Cornell Lab account management system’s account creation page. |

Table 18: Get membership

|  |  |
| --- | --- |
| **Use case name** | Gift membership on Cornell Lab |
| **Actors** | User, Cornell Lab account management system |
| **Description** | When User presses the “Give membership as a Gift” button and is in the United States, User will be redirected to Cornell Lab account management system’s gift membership site. |
| **Data** | - |
| **Preconditions** | - |
| **Stimulus** | The User presses the “Give membership as a Gift” button. |
| **Basic Flow** | Step 1 - Gifting a membership request is received from the User.  Step 2 - FeederWatch system redirects User to Cornell Lab account management system’s membership gifting page. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The user shall be in the Cornell Lab account management system’s membership gifting page. |

Table 19: Gift membership on Cornell Lab

|  |  |
| --- | --- |
| **Use case name** | Gift membership on Bird Studies Canada |
| **Actors** | User, Bird Studies Canada account management system |
| **Description** | When User presses the “Give membership as a Gift” button and is in Canada, User will be redirected to Bird Studies Canada account management system’s gift membership site. |
| **Data** | - |
| **Preconditions** | - |
| **Stimulus** | The User presses the “Give membership as a Gift” button. |
| **Basic Flow** | Step 1 - Gifting a membership request is received from the User. |

|  |  |
| --- | --- |
|  | Step 2 - FeederWatch system redirects User to Bird Studies Canada account management system’s membership gifting page. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The user shall be in the Bird Studies Canada account management system’s membership gifting page. |

Table 20: Gift membership on Bird Studies Canada

|  |  |
| --- | --- |
| **Use case name** | Sign in |
| **Actors** | User, Cornell Lab account management system |
| **Description** | User gets redirected to Sign In page of Cornell Lab account management system. After Cornell Lab grants authorization, FeederWatch System also authorizes the User. |
| **Data** | Authorization data |
| **Preconditions** | - |
| **Stimulus** | User presses “Your Data” link in FeederWatch system. |
| **Basic Flow** | Step 1 - User requests signing in.  Step 2 - User is redirected to Cornell Lab account management system to sign in.  Step 3 - If Cornell Lab account management system authorizes the User, also FeederWatch system authorizes the User and redirected to FeederWatch website. |
| **Alternative Flow** | If Cornell Lab account management system does not authorize the User, the User will not be redirected to FeederWatch website and will remain in Cornell Lab account management system’s sign in page. |
| **Exception Flow** | - |
| **Postconditions** | The user will be authorized in both Cornell Lab account management system and FeederWatch system. |

Table 21: Sign in

|  |  |
| --- | --- |
| **Use case name** | Upload photo |
| **Actors** | User |
| **Description** | Users can upload a photo of a bird to FeederWatch system in “Upload your photos” segment under the Participant Photos. |
| **Data** | Username, submission name, email address, uploaded photos, submission title, city or state, description, category, species, tags, approval of terms and conditions |
| **Preconditions** | - |
| **Stimulus** | User presses “Submit” button of Upload photo segment. |
| **Basic Flow** | Step 1 - If the User is a member and already authorized beforehand, then some of the form will be readily filled and username will be a required field.  Step 2 - Member user requests submission after uploading a photo and entering necessary information. Step 3 - FeederWatch system gets the information and inserts them into the database.  Step 4 - The photo uploaded by the User is shown on the Browse photos segment. |
| **Alternative Flow** | If the user is not a member or not authorized readily, the  form wouldn’t be readily filled and username field won’t be required. |
| **Exception Flow** | If the Member user presses the “Submit” button before filling the required information, error message is shown to user without deleting the information filled. |
| **Postconditions** | The photo Member user uploaded is shown under Browse photos segment. |

Table 22: Upload data

|  |  |
| --- | --- |
| **Use case name** | Manage account |
| **Actors** | Member user, Cornell Lab account management system |
| **Description** | Member user can give personal information other than the necessary information gathered during the registering process of Cornell Lab account management system. |
| **Data** | - |

|  |  |
| --- | --- |
| **Preconditions** | User should be Member user. |
| **Stimulus** | User presses “Manage Account” button. |
| **Basic Flow** | Step 1 - Managing account request is received from the Member user.  Step 2 - Member user is redirected to Cornell Lab account management system. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The Member user is redirected to account management page of the Cornell Lab account management system |

Table 23: Manage account

|  |  |
| --- | --- |
| **Use case name** | Enter observation data |
| **Actors** | Paid member user |
| **Description** | Paid member users can submit their observation data about the birds. |
| **Data** | Observation data |
| **Preconditions** | Paid member user should be logged in and also the account should have been confirmed by using a FeederWatch ID. |
| **Stimulus** | User presses “Submit Tally” button. |
| **Basic Flow** | Step 1 - Submitting the observation data is received from the Paid member user.  Step 2 - Information about the observation of the Paid member user is fetched by FeederWatch system and it is inserted into the database. |
| **Alternative Flow** | - |
| **Exception Flow** | - |
| **Postconditions** | The observation data submitted by the Paid member user is saved into the database. |

Table 24: Enter observation data

|  |  |
| --- | --- |
| **Use case name** | Review observation data |
| **Actors** | Paid member user |
| **Description** | Paid member users can review the observation data that they entered before. |
| **Data** | Observation data that was entered before by the Paid member user, corrected observation data if any correction was made |
| **Preconditions** | Paid member user should be authorized by Cornell Lab account management system beforehand. |
| **Stimulus** | Paid member user presses “Enter and Review Your Data” button. |
| **Basic Flow** | Step 1 - Reviewing the observation data request is received from the Paid member user.  Step 2 - Paid member user is shown the observation data that they provided before.  Step 3 - If the Paid member user thinks that there is an error in the entry, then they can edit the observation data. Step 4 - After editing the observation data, Paid member user can press the “Submit Tally” button to update the database of FeederWatch system. |
| **Alternative Flow** | If the Paid member user does not do any changes in the observation data, then nothing is changed in the database of FeederWatch system. |
| **Exception Flow** | - |
| **Postconditions** | If any change is made, then database is changed accordingly. |

Table 25: Review observation data

|  |  |
| --- | --- |
| **Use case name** | Add statistical data |
| **Actors** | Data analyzer |
| **Description** | Data analyzer inserts interpreted statistical data into the database of FeederWatch system. This interpreted data is to be shown to users afterwards. |
| **Data** | Interpreted statistical data |
| **Preconditions** | Data analyzer should be authorized by Cornell Lab |

|  |  |
| --- | --- |
|  | account management system beforehand. |
| **Stimulus** | Data analyzer presses “Submit” button. |
| **Basic Flow** | Step 1 - Request of adding statistical data from the Data analyzer is received.  Step 2 - Files uploaded by the Data analyzer in a specific style which can be interpreted by FeederWatch system are fetched.  Step 3 - Uploaded files are parsed and interpreted by the FeederWatch system. |
| **Alternative Flow** | - |
| **Exception Flow** | If the uploaded files cannot be parsed, Data analyzer gets an error message with a little explanation of what went wrong. |
| **Postconditions** | The data that Data analyzer submitted is ready to be shown to Users. |

Table 26: Add statistical data

# Usability requirements

**req36:** A user shall be able to reach any function of the system using at most 3 buttons.

**req37:** Font size shall be big since the users of FeederWatch includes elder people.

**req38:** Long pages shall include “back to top” button to allow fast access to the top of the page.

**req39:** A search bar shall exist in all pages.

**req40:** A clickable site map shall be placed under all pages.

# Performance requirements

**req41:** The system shall respond to at least 1000 users simultaneously. **req42:** The system shall generate statistical graphs in at most 150ms. **req43:** Internet speed shall be fast enough to download a copy of the full database in at most 1 hour without interfering with normal website activities.

# Logical database requirements

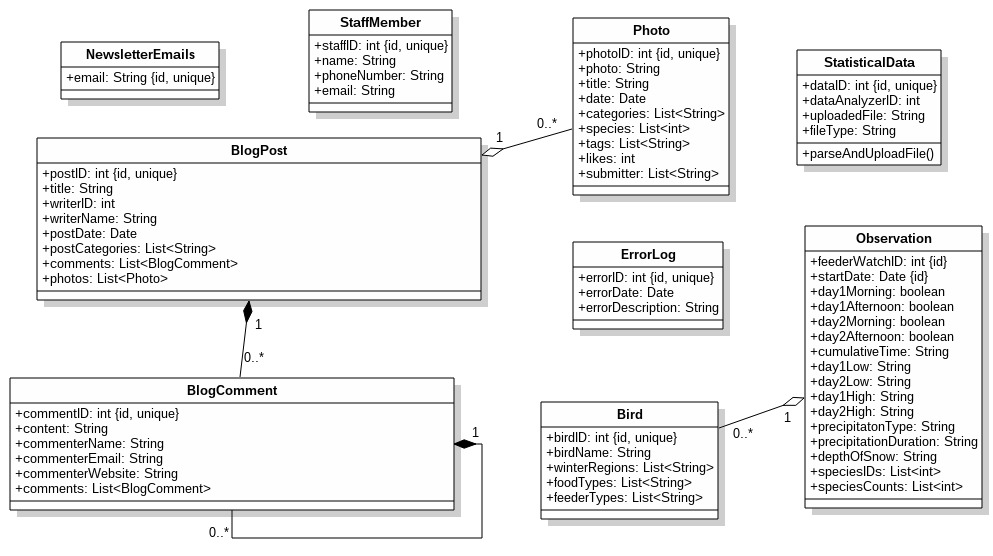


Figure 16: Logical Database Class Diagram

**req44:** Observation table shall be used when Tally Sheet is submitted or reviewed by the user.

**req45:** Only Data analyzer and IT Staff shall be able to access and filter Observation table and download the result.

**req46:** Observation table shall be in a one-to-many relationship with Bird table.

**req47:** All Users shall be able to search through and filter Bird table while browsing common feeder birds.

**req48:** Entries to the ErrorLog table shall be added by informITStaff function. This function shall be called only by the internal objects of the FeederWatch system when encountered errors.

**req49:** Only ITStaff shall be able to view the results of ErrorLog table and filter them with respect to date.

**req50:** NewsletterEmails table shall keep only the subscriptions to the FeederWatch newsletter.

**req51:** An entry to the NewsletterEmails table shall be added when any user registers their email address to newsletter and FeederWatch option is selected.

**req52:** Photo table shall store all photos used in FeederWatch system including photos in Browse Photos and Blog sections.

**req53:** BlogPost table shall be in a one-to-many relationship with Photo table since blog posts may contain photos in them.

**req54:** BlogComment shall be a weak entity since a blog comment cannot exist without its blog post or its blog comment.

**req55:** A BlogComment entry shall be able to have other blog comments. **req56:** An entry to a BlogPost table shall be added when a Blog writer adds a new post.

**req57:** An entry to BlogComment table shall be able to be added by any User.

**req58:** The information about staff that appears in Admin Panel shall be stored in StaffMember table.

**req59:** All integrity constraints shall be checked by the DBMS and the high-level language used in the implementation of FeederWatch. **req60:** Database backup shall be done at least once a week.

# Design constraints

**req61:** The owner of the photograph in Participant Photos section shall be stated explicitly.

**req62:** All articles and comments posted in the blog section shall be stored for legal purposes.

**req63:** The project shall comply with ornithology rules and regulations.

# Software system attributes

* + 1. **Reliability**

**req64:** Probability of data corruption on failure shall not exceed

0.042. This can be done by using two hard disks and the proper DBMS.

**req65:** Mean time to failure shall be at least one week.

# Availability

**req66:** Time to restart after failure shall not exceed 20 minutes.

**req67:** The system shall always be available except for between 3

a.m. and 5 a.m. Central Time (GMT-5) which is when any maintenance needed will be done.

**req68:** The unavailability of the system due to failures shall not exceed 24 hours in a year.

# Security

**req69:** The project shall be invulnerable to SQL injection since the data FeederWatch acquires is the core part of the project.

**req70:** The account management system shall be handled by Cornell Lab.

**req71:** While getting the authentication information from Cornell Lab, the system shall use HTTPS to ensure security.

# Maintainability

**req72:** The project shall be documented well to allow newly recruited members to understand it well.

**req73:** Implementation of the project shall be in a modular fashion to make it easier to change the structure of classes.

# Portability

**req74:** The project shall support different operating systems other than linux, which the system will run on, to allow changes in the future.

**req75:** The system shall not depend on operating-system-specific libraries.

**req76:** The implementation shall be in Java since JVM is supported on many operating systems.

# Supporting information

FeederWatch system fills the gap between people and the database which the statistical data collected. It is a project to understand the distribution of birds in winter season.

System has a user interface which is a website and a database which stores data submitted by paid users. The database also contains statistical data which were interpreted by data analyzers.

Accounts and payment are fully done by Cornell (and also Bird Studies Canada in registration process) and when a user wants to do something FeederWatch always asks Cornell Lab whether they can do it. Thus, FeederWatch does not store any user information.

# Verification

# Appendices

# Assumptions and dependencies

# Acronyms and abbreviations