Project FeederWatch

Software Design

Description

METU CENG350 2016-2017 Spring

Group 47	
Name	**Student ID**
Hazan Anayurt	2098747
Sezai Artun Özyeğin	2036515

CHANGE HISTORY

Version	**Date**	**Status**
1.0	23.04.2017	Released
1.1	04.05.2017	Released

TABLE OF CONTENTS

FIGURES

TABLES

1. Introduction

1.1. Purpose

The purpose of this document is to provide a baseline for the project on which the system can be implemented. The document specifies the detail of which components are to be built to realize Project FeederWatch and in which way these components shall be built.

1.2. Scope

This document describes the system at an architectural level, with use cases and their detailed step-by-step descriptions, the components and subcomponents of the system with the hardware these components will be working on, the interface classes and how the components of the system interact with each other or external systems or users interact with the system through these interfaces, and a complete design of the database with how create, read, update and delete operations are performed on the classes that are mapped onto tables in this database.

1.3. Stakeholders and their concerns

Users: Users are people from all over the world who are simply visitors of the website. One of their concerns is to get more informed about bird species and statistics about their movement in winter with a good presentation with graphical interfaces. They also want a platform to share the bird photos they take to be seen by more people who are interested in birds.

Paid member users: These are the users that are from North America and have paid either Cornell Lab or Bird Studies Canada to become a paid member. On top of the concerns of regular users, their concern is to have more guidance while bird watching and collecting data, and to have somewhere to submit these data to so their observations can be useful to Ornithology studies and help save bird species. They also want to donate to Cornell Labs to help fund their studies.

IT staff of FeederWatch: IT staff is who will be maintaining the system and handling technical issues after the platform is put to work. Their concern is to have a simple interface for an admin panel that contains all the functionality needed for them to view errors and to make necessary changes on the system to resolve these errors while being efficient, and a system that is overall easy to maintain.

System Developer: These are the programmers who develop the software that will lie under the FeederWatch system. Their concern is having no ambiguity with the software that they are going to develop. Therefore, they are highly dependent on SRS and SDD.

Cornell Lab of Ornithology: Cornell Lab is essentially the customer that issued this project. Their concern is having a platform for collecting data of feeder birds specifically with effort from only regular citizens and have a much broader area of study thanks to users from all over North America. Since Cornell Lab already has an account management system and wants to connect every project with a single account database, they want a system that can work with this already existing account system.

Bird Studies Canada: Bird Studies Canada is the platform users that live in Canada can sign up through. Their concern is for Canadian users to be able to take part in bird watching and data collection in North America while only being a part of the account confirmation, membership and donations.

Data Analyzer: These are the statisticians who interpret the observation data collected by platform users. Their concern is to access data easily and fast in a simple interface and be able to create graphs from the data they interpreted without too much manual work.

Blog Writer: These are the special type of platform users who are authorized to write article for the blog. Their concern is to have a blog where they will get the opportunity to share their knowledge with as many users as possible with an easy-to-use interface with a photo-uploading feature and categorising posts for easier access.

2. References

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

IEEE standard for information technology--systems design--software design descriptions. (2009). New York, NY: Institute of Electrical and Electronics Engineers.

Other sources:

Dong Ngo March 29, 2013 1:02 PM PDT @riceandstirfry. (2012, December 03). Digital storage basics, Part 3: Backup vs. redundancy. Retrieved May 2, 2017, from https://www.cnet.com/how-to/digital-storage-basics-part-3-backup-vs-redundancy/

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

3. Glossary

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
DBMS	Database Management System
URL	Uniform Resource Locator, web address
AUTHDATA	Authorization data
RAID	Redundant Array of Independent Disks
HDD	Hard Disk Drive
SATA	Serial ATA, a computer bus interface

Table 1: Glossary

4. Architectural Views

4.1. Context view

In this viewpoint all use cases of the system ares specified with detailed step-by-step descriptions of the basic flow, along with an alternative flow and/or an error flow if applicable. These description tables describe in detail how system services should work in different scenarios and are intended to act as a guideline while these functions are being implemented. The Context Diagram below shows the general exchanges between actors and the system, while the Use Case Diagram shows how different actors, namely different types of users or external systems interact with FeederWatch System through use cases.

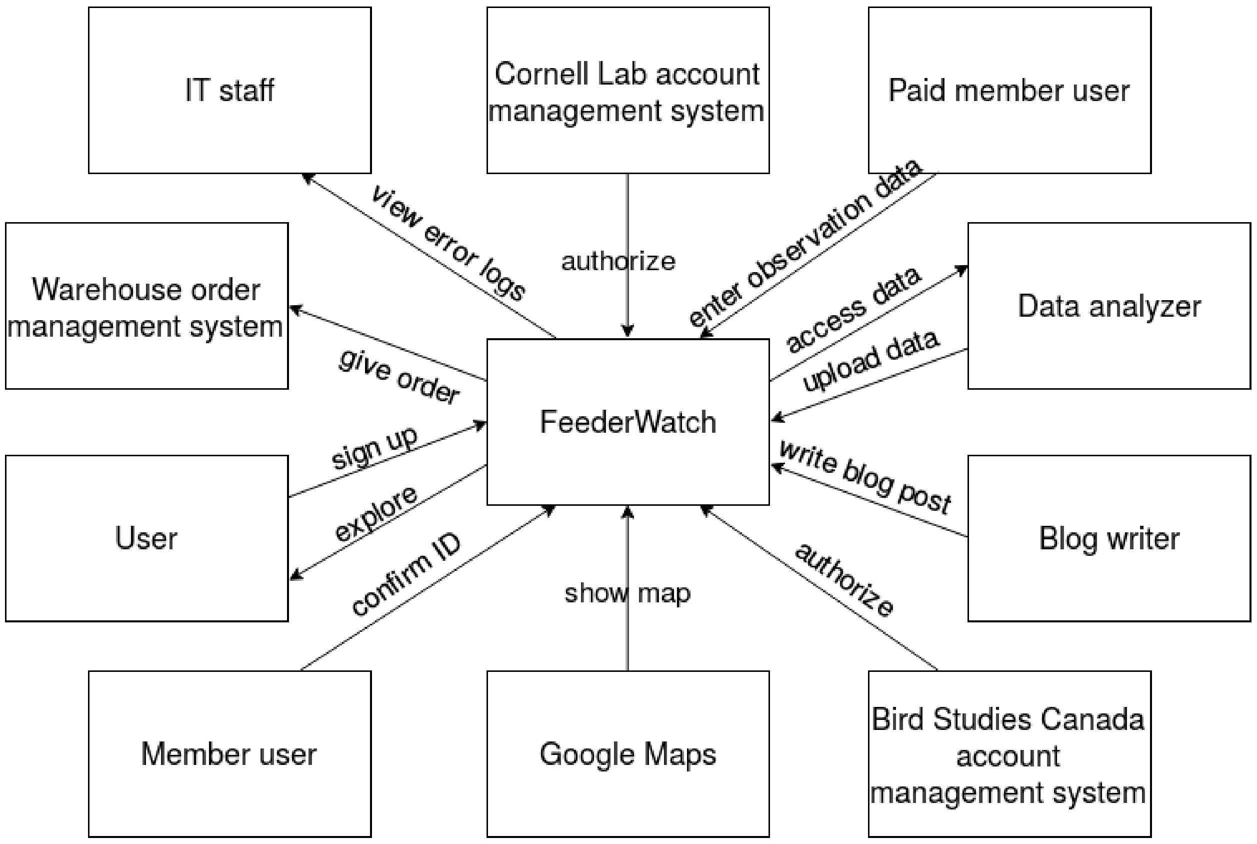


Figure 1: Context diagram

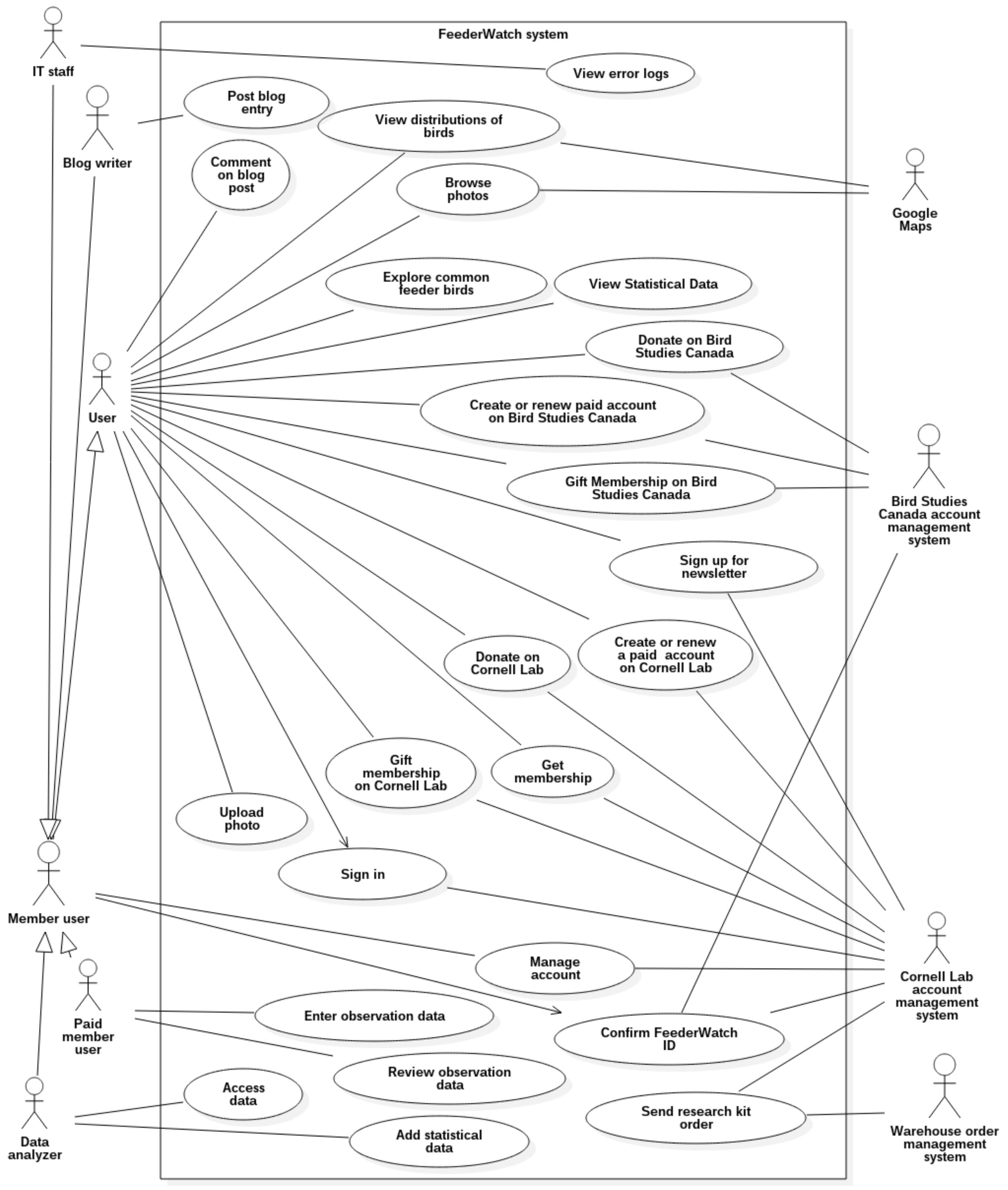


Figure 2: Use Case Diagram

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 2: Send research kit order

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 3: Confirm FeederWatch ID

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 4: 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 5: 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 6: 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 7: 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 8: 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 9: 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 10: 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 11: 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 12: 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 13: 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: 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 20: 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 21: Upload photo

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 22: 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 23: 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 24: 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 25: Add statistical data

4.2. Composition view

In this viewpoint, components of the system such as subsystems and the functionalities of these components will be shown from a top-level point of view. Design entities and their relationships, with the hardware these entities are mapped to can be seen in the composition and deployment diagrams below.

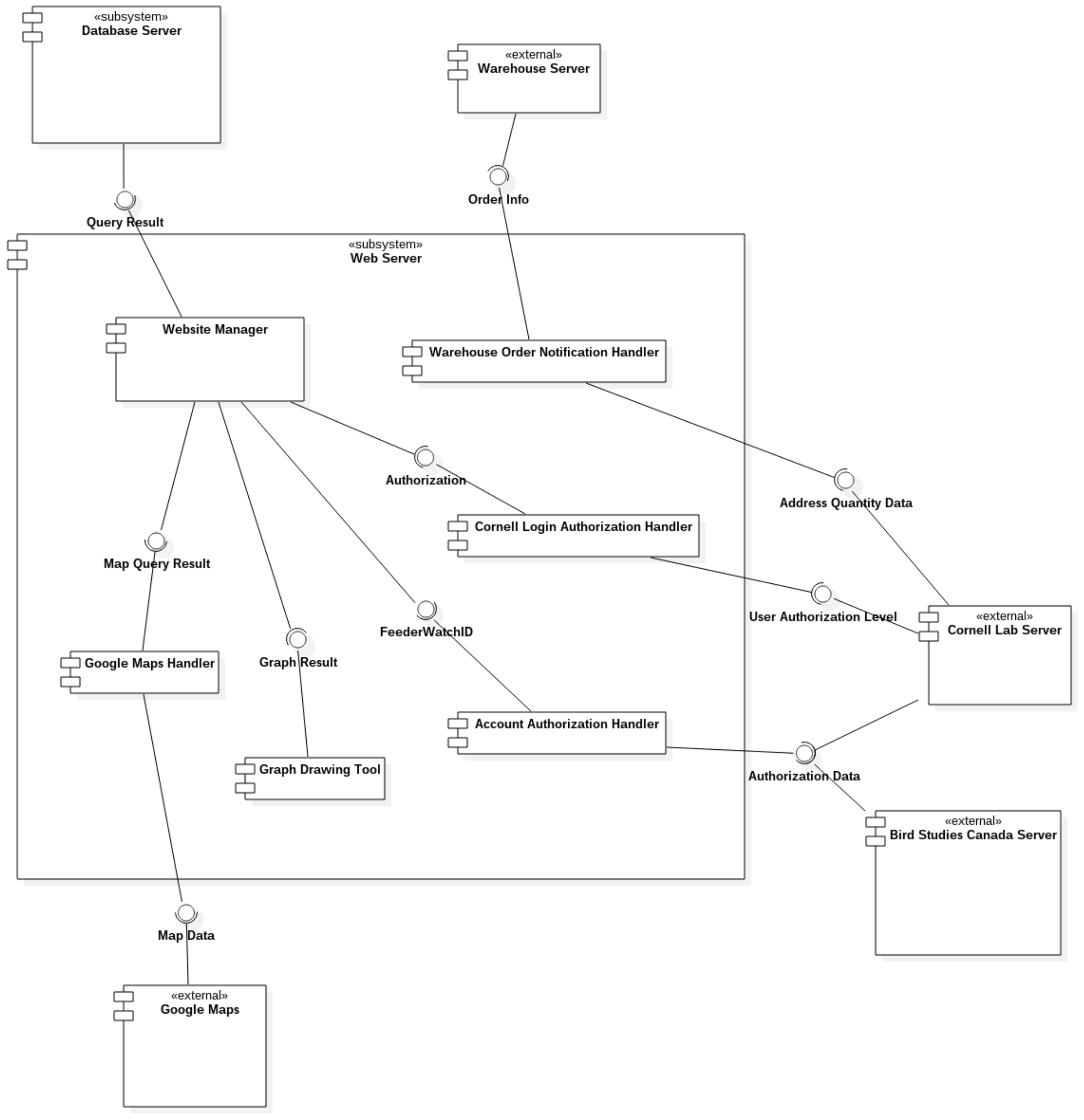


Figure 3: Component Diagram

Design Rationale:

- All of FeederWatch website's functionalities, namely the different user interfaces employed in different pages, modifying the database upon user input and querying the database in order to display data, are done by the component named Website Manager. The purpose for having just one component to handle all of the user interfaces was because they have similar functionalities such as displaying data, taking input and reaching the database.
- Web Server subsystem has different components to communicate with different external systems to make Website Manager simpler and make tasks more organized.
- After querying the database to acquire needed data, Web Server sends data to Google Maps Handler component to be transformed into a suitable format and sent to Google Maps, allowing the data to be shown on map.
- FeederWatch system is responsible for contacting the Warehouse for ordering research kits as this isn't directly related to account management and therefore Cornell Lab. For this purpose, there is the Warehouse Notification Handler component which can be contacted by Cornell upon registration of a new user and notifies the Warehouse system in turn.
- Users can buy membership via Cornell Lab or Bird Studies Canada, but the activation of memberships is done through FeederWatch system in both cases. For this, the Account Authorization Handler component provides an interface to both Cornell Lab and Bird Studies Canada to send ￼required data for authorization.
- Since account data are stored in Cornell Lab and for not to have to keep any user data in the FeederWatch system, every time a user wants to perform an action, their authorization is verified by Cornell Lab via Cornell Login Authorization Handler.
- As a Graph Drawing Tool, an off-the-shelf application is used so it isn't a part of Website Manager but a separate component that runs on Web Server and is used by Website Manager.

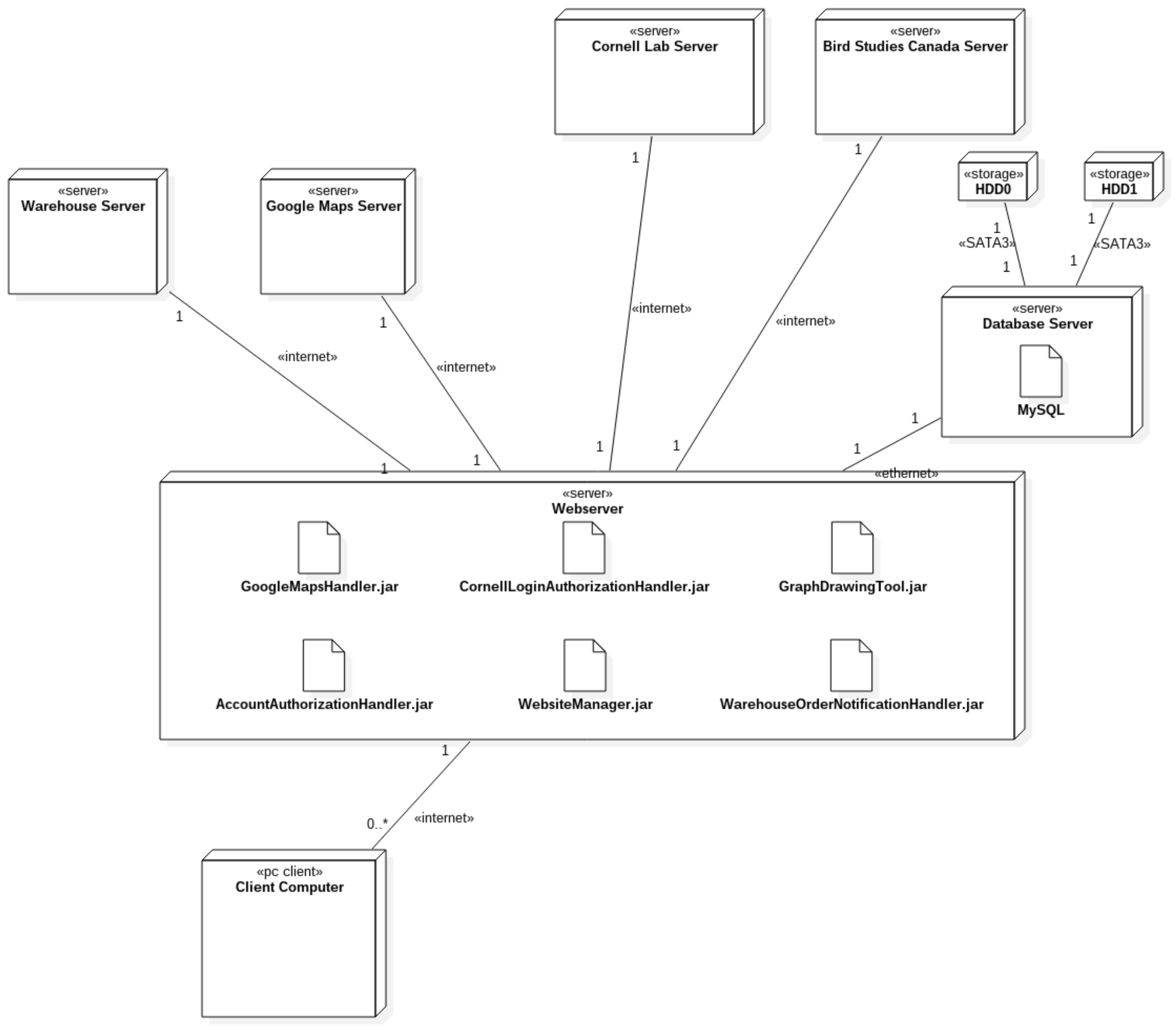


Figure 4: Deployment Diagram

Design Rationale:

- The Database Server is separate from the Web Server to make the system scalable, to be able to accommodate the future need for a bigger Database Server without changing the Web Server.
- Database Server has 2 hard disks for increasing the reliability of the system. RAID 1 will be used to mirror the data in hard drives. By mirroring, although it will be slower than RAID 0, the probability of data corruption will be minimized.

- As a relational database management system, MySQL is used because it is open-source and free.

4.3. Information view

In this view, persistent information that is kept in the database and how these map into classes with methods and attributes is shown in a class diagram, along with the relationships of these classes with each other. Create, read, update and delete functionality that is assigned to each database component or class is also specified.

4.3.1. Service Interfaces

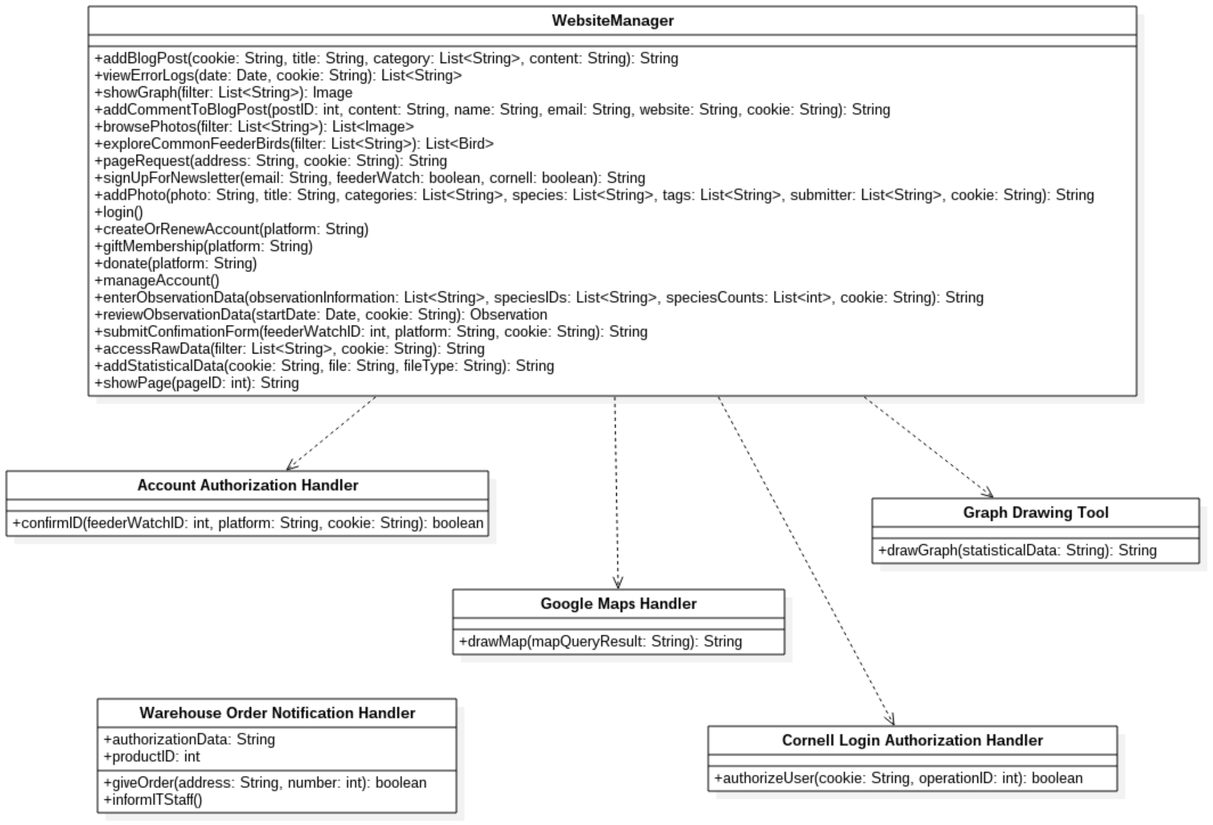

WebsiteManager
+addBlogPost(cookie: String, title: String, category: List<String>, content: String): String
+viewErrorLogs(date: Date, cookie: String): List<String>
+showGraph(filter: List<String>): Image
+addCommentToBlogPost(postID: int, content: String, name: String, email: String, website: String, cookie: String): String
+browsePhotos(filter: List<String>): List<Image>
+exploreCommonFeederBirds(filter: List<String>): List<Bird>
+pageRequest(address: String, cookie: String): String
+signUpForNewsletter(email: String, feederWatch: boolean, cornell: boolean): String
+addPhoto(photo: String, title: String, categories: List<String>, species: List<String>, tags: List<String>, submitter: List<String>, cookie: String): String
+login()
+createOrRenewAccount(platform: String)
+giftMembership(platform: String)
+donate(platform: String)
+manageAccount()
+enterObservationData(observationInformation: List<String>, speciesIDs: List<String>, speciesCounts: List<int>, cookie: String): String
+reviewObservationData(startDate: Date, cookie: String): Observation
+submitConfimationForm(feederWatchID: int, platform: String, cookie: String): String
+accessRawData(filter: List<String>, cookie: String): String
+addStatisticalData(cookie: String, file: String, fileType: String): String
+showPage(pageID: int): String
Account Authorization Handler
+confirmID(feederWatchID: int, platform: String, cookie: String): boolean
Graph Drawing Tool
+drawGraph(statisticalData: String): String
Google Maps Handler
+drawMap(mapQueryResult: String): String
Warehouse Order Notification Handler
+authorizationData: String
+productID: int
+giveOrder(address: String, number: int): boolean
+informITStaff()
Cornell Login Authorization Handler
+authorizeUser(cookie: String, operationID: int): boolean

Figure 5: Service Interfaces Class Diagram

Operation	**Description**
addBlogPost	Adds a new blog post to the blog.
viewErrorLogs	ITStaff is able to see the error logs of FeederWatch system filtered by date.
showGraph	When a user requests to see the data that the Data Analyzers entered before, this operation first queries the database and sends the graphs to the user which were drawn by the drawGraph operation of Graph Drawing Tool according to the data.
addCommentToBlogPost	A user can post a comment to a specific blog post.
browsePhotos	A user is able to browse photos and filter them.
exploreCommonFeederBirds	Users are able to explore common feeder birds and they can filter the results with the given filter options.
pageRequest	This operation checks the authorization level of the user by using Cornell Login Authorization Handler and uses showPage operation when it is appropriate.
signUpForNewsletter	A user is able to sign up for FeederWatch's or Cornell Lab's newsletter with their email addresses.
addPhoto	Users can upload a photo and also provide its title and can specify its category, the species in it and tag it.
login	This operation just redirects the user to Cornell Lab's login site since FeederWatch System does not store any user information.
createOrRenewAccount	When this operation is used, the user is redirected to the website of the selected platform.
giftMembership	According to the selected platform, the user requested to use this operation is redirected to that platform's membership gifting page.
donate	Users are redirected to the donation page of the platform they selected.
manageAccount	The user requested this operation is only redirected to Cornell Lab's account management site since FeederWatch system does not store any user information.
enterObservationData	A paid member user can enter their observation data and provide the information discussed in Interface viewpoint, Tally Sheet Submission Interface.

reviewObservationData	A paid member user is able to review the observation data they entered before by providing their FeederWatchID and the start date of the observation.
submitConfirmationForm	With this operation a member user is able to match their account with their FeederWatchID. This operation uses authorizeUser operation of Account Authorization Handler to communicate with Cornell Lab Server or Bird Studies Canada Server.
accessRawData	This operation can be used by ITStaff and Data Analyzers to obtain the raw data given by the Paid Member Users.
addStatisticalData	Data Analyzers can add new statistical data they interpreted from the raw data. This added data is used for showing users some refined statistical data when they requested with showGraph operation.
showPage	When the user requested a page with the operation pageRequest, this operation provides the page and sends that to the user.
confirmID	Website manager requests to confirm a FeederWatch ID when it is requested by a member user, and this operation of Account Authorization Handler send the ID to the Cornell Lab Server or Bird Studies Canada Server.
authorizeUser	This operation communicates with Cornell Lab Server to decide whether a user is authenticated to do a specific operation. This operation is used for every functionality of the FeederWatch System.
drawMap	This operation of Google Maps Handler communicates with Google Maps using its API. While doing that, it uses the query result given by the Website Manager and parses it before directly using it on Google Maps API.
giveOrder	This operation can be user only by Cornell Lab and it is for giving research kit orders to Warehouse. Since the authorization data and product ID stored in Warehouse Order Notification Handler, this has to be done through giveOrder operation.
informITStaff	When the Warehouse does not have enough research kits left, it replies accordingly and Warehouse Order Notification Handler informs the ITStaff.

drawGraph	According to the Statistical Data input sent from the Website Manager, draws a line graph and returns it in String form to be shown on website.

Table 26: Service interfaces operation descriptions

Operation	Inputs	Outputs	Exceptions
addBlogPost	Authorization cookie Blog entry Title Category	Message operation OK or failed	Authorization Failed Database server not available
viewErrorLogs	Authorization cookie Date	Error list	Authorization Failed Database server not available
showGraph	Filter options	Resultant image	Not existing bird species Database server not available
addCommentToBlogPost	postID Content Name Email Website Authorization cookie	Message operation OK or failed	Invalid email address Database server not available
browsePhotos	Filter options	Resultant images after querying the database	Database server not available
exploreCommonFeederBirds	Filter options	Resultant information about common feeder birds after querying the database	Database server not available
pageRequest	URL Authorization cookie	Requested page	Authorization failed
signUpForNewsletter	Email FeederWatch choice	Message operation OK or failed	Cornell Lab server not available

	Cornell Lab choice		
addPhoto	Image Title Categories Species Tags Submitter Authorization cookie	Message operation OK or failed	Image type is not supported Database server not available
login	-	-	Cornell Lab server not available
createOrRenewAccount	Platform	-	Cornell Lab server not available Bird Studies Canada server not available
giftMembership	Platform	-	Cornell Lab server not available Bird Studies Canada server not available
donate	Platform	-	Cornell Lab server not available Bird Studies Canada server not available
manageAccount	-	-	Cornell Lab server not available
enterObservationData	Authorization cookie Observation information Species IDs Species Counts	Message operation OK or failed	Authorization failed Database server not available
reviewObservationData	Authorization cookie	Observation information	Authorization Failed

	Start date of the observation		Database server not available
submitConfirmationForm	Authorization cookie FeederWatch ID Platform	Message confirmation result	Cornell Lab server not available
accessRawData	Authorization cookie Filter options	Raw Data	Authorization failed Database server not available
addStatisticalData	Authorization cookie Statistical data file File type	Message operation OK or failed	Authorization failed Database server not available
showPage	Page ID	Requested page	Page not found Database server not available
confirmID	FeederWatch ID Authorization cookie Platform	Confirmation status	Cornell Lab server not available Bird Studies Canada server not available
authorizeUser	Authorization cookie Operation ID	Authorization status	Cornell Lab server not available
drawMap	Query result	Map URL	Google Maps down
giveOrder	New users' addresses Quantity of research kits	Confirmation status	Authorization failed
informITStaff	-	-	Database server not available
drawGraph	Statistical data	Graph result	Data corrupted

Table 27: Service interfaces operation design

Design rationale:

- There is a handler class to communicate with each external system the FeederWatch system interacts with since these operations didn’t directly deal with users.
- The Website Manager is responsible for querying the database when data is needed for different user operations as having too many classes that directly interact with data would have complicated the design.
- Every operation that needs authorization takes an authorization cookie as input because no user data is kept in FeederWatch database and Cornell Lab Account Authorization System needs to be asked if the user is allowed to do every action they are trying to do.
- As all user-account-related data is kept on Cornell Lab Servers or Bird Studies Canada Servers, every operation that is related to user accounts is just redirection to either platform, therefore only takes platform data as input.

4.3.2. CRUD Operations

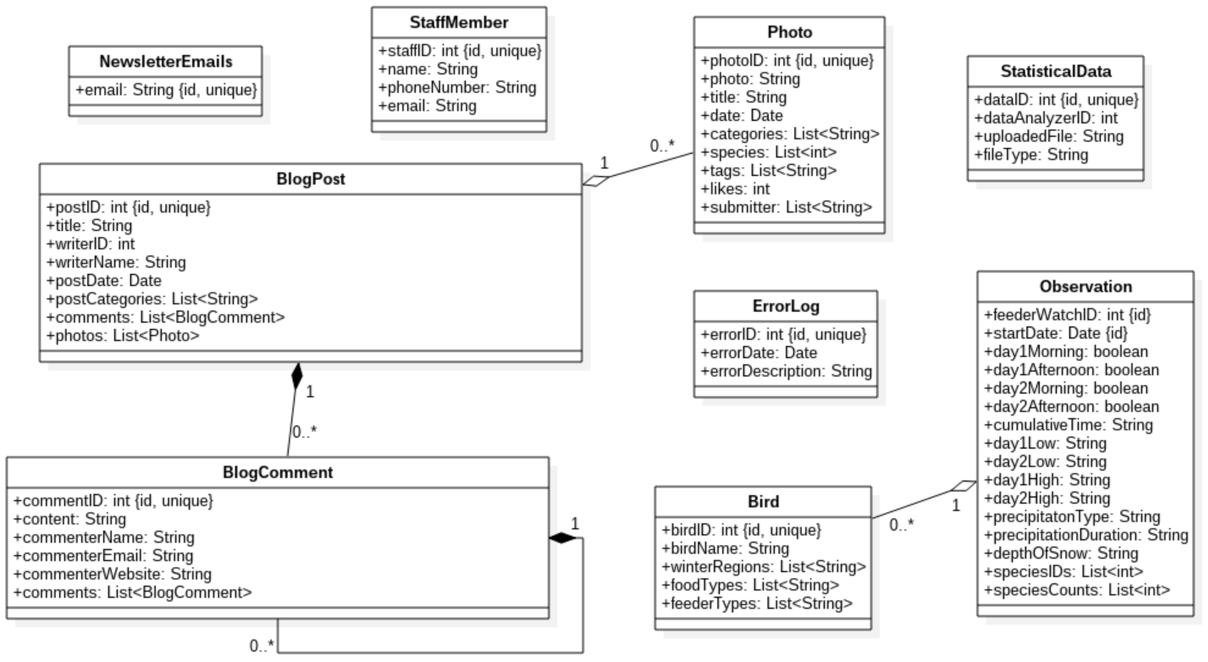


Figure 6: Database Class Diagram

Operation	**CRUD Operations**

addBlogPost	CREATE - BlogPost
	READ -
	UPDATE -
	DELETE -

viewErrorLogs	CREATE -
	READ - ErrorLog
	UPDATE -
	DELETE -

showGraph	CREATE -
	READ - StatisticalData
	UPDATE -
	DELETE -

addCommentToBlogPost	CREATE - BlogComment
	READ -
	UPDATE - BlogPost
	DELETE -

browsePhotos	CREATE -
	READ - Photo
	UPDATE -
	DELETE -

exploreCommonFeederBirds	CREATE - BlogPost
	READ - Bird
	UPDATE -
	DELETE -

signUpForNewsletter	CREATE - NewsletterEmail
	READ -
	UPDATE -
	DELETE -

addPhoto	CREATE - Photo
	READ -
	UPDATE -
	DELETE -

enterObservationData	CREATE - Observation
	READ -
	UPDATE -
	DELETE -

reviewObservationData	CREATE -
	READ - Observation
	UPDATE - Observation
	DELETE - Observation

accessRawData	CREATE -
	READ - Observation
	UPDATE -
	DELETE -

addStatisticalData	CREATE - StatisticalData
	READ -
	UPDATE -
	DELETE -

showPage	CREATE -
	READ - BlogPost, BlogComment, Photo, Bird, StatisticalData, Observation
	UPDATE -
	DELETE -

informITStaff	CREATE - ErrorLog

	READ - UPDATE - DELETE -

Table 28: CRUD Operations

Design rationale:

- MySQL was chosen as a relational DBMS to ensure all integrity constraints are satisfied.
- No user data is kept in the FeederWatch Database because Cornell Labs are responsible for keeping account information and instead of session-based data being kept in the FeederWatch servers to be able to authenticate users Cornell can be asked whenever a user tries to perform an action to make the database simpler.
- Even though no user data is kept in the database, the staff member information is an exception to this in case of emergencies when a staff member needs to be contacted and Cornell Servers cannot be reached.
- BlogComment is a weak entity (shown with composition relation in the class diagram) because a blog comment cannot exist on its own without a BlogPost.
- Photo is in an aggregation relation with BlogPost because a photo can exist either in its own or within a blog post.

4.4. Interface view

In this view, different internal interfaces between components of the FeederWatch system, external interfaces between FeederWatch System and other systems or different types of users, the definitions of these interfaces and use cases they relate to are specified.

4.4.1. Internal Interfaces

The Interface Between the Database Server and the Website Manager: The Website Manager queries the Database Server for every operation in the system that is related to the Database. The Website Manager send the query to the Database Server in the form of string that is written in SQL. Then, the Database Server tries to run that query using MySQL DBMS. If the SQL query generates error, then the error message is sent to the Website Manager. Otherwise, the result of the query is sent.

Design Rationale:

- Since the whole project shapes around the data, the database storing it is in a separate system only devoted to it; thus, an interface is needed between the Website Manager and the Database server.
- The sent data is the SQL query string that is generated according to the operation the Website Manager tries to do. SQL query is selected since the DBMS is MySQL.

The Interface Between the Google Maps Handler and the Website Manager: The Website Manager sends the query results of the database to the Google Maps Handler to convert it to a map data so that Google Maps API can understand. After the communication with the Google Maps, Google Maps Handler sends the resultant link back to the Website Manager.

Design Rationale:

- The direct query result cannot be understood by Google Maps API; therefore, the Website Manager does not handle with its conversion and uses Google Maps Handler’s interface to communicate with Google Maps.

The Interface Between the Graph Drawing Tool and the Website Manager: To provide the graphs in Explore Tab, Website Manager uses Graph Drawing Tool’s interface. Website manager sends the graph query results to Graph Drawing Tool. Graph Drawing Tool creates graphs and sends it back to the Website Manager.

Design Rationale:

- Because Graph Drawing Tool is an off-the-shelf software, it is separate from the Website Manager and it provides an interface to be used.

The Interface Between the Account Authorization Handler and the Website Manager: When the user enters FeederWatch ID and chooses a platform on the website, the Website Manager sends this information along with the cookşe that came from the user to Account Authorization Handler through this interface for the information to be confirmed by whichever platform was selected. After Account Authorization Handler gets the match result from one of the platforms, the result is sent back to Website Manager for the user to be shown an appropriate message on the website.

Design Rationale:

- The Website Manager only needs to send the Cookie and the FeederWatch ID and the Cornell Lab Server or Bird Studies Canada Server will be able to recognize which user is the owner of the cookie since the cookie was initially generated by the Cornell Lab Server itself.

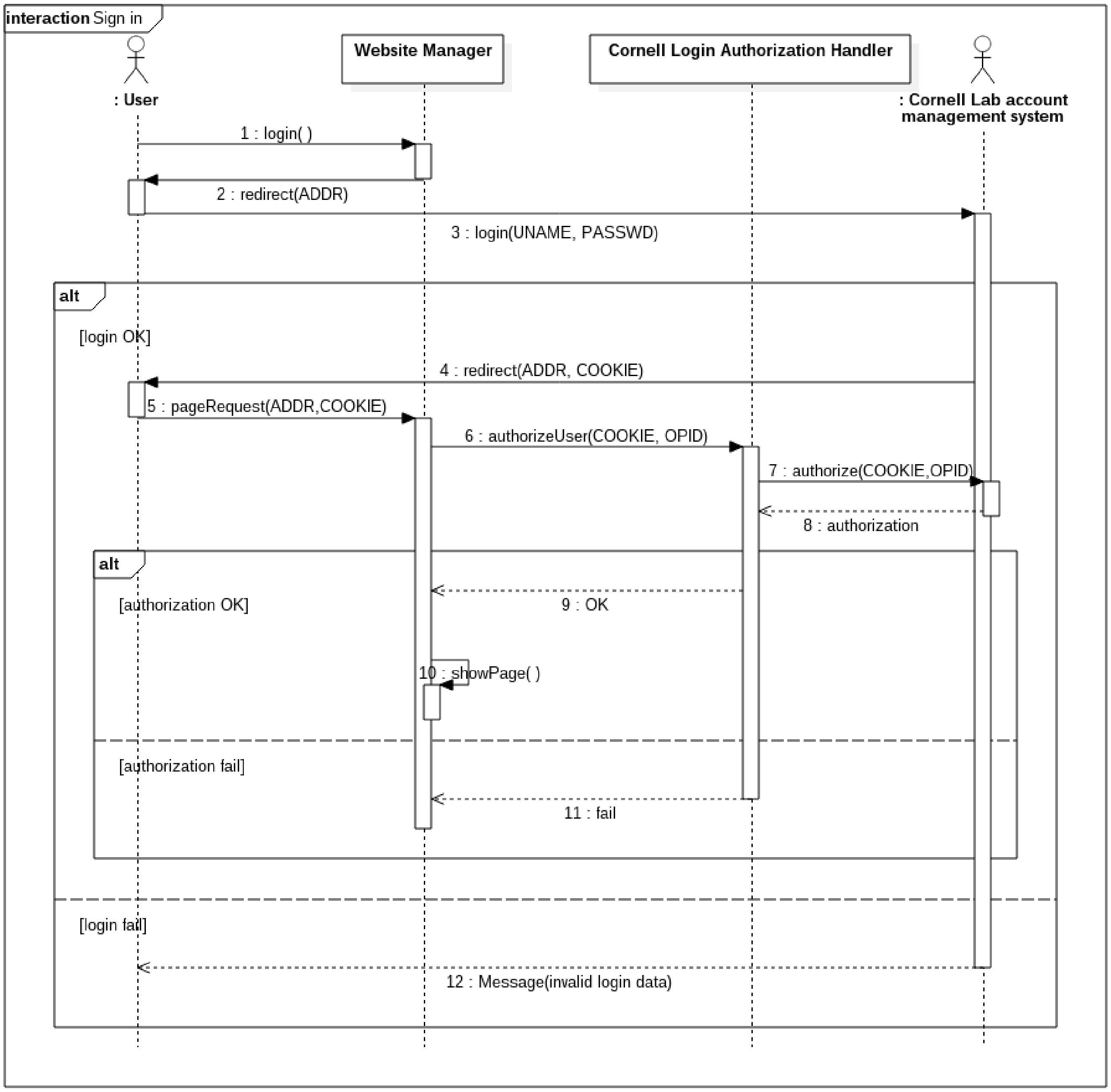

interaction Sign in
: User
Website Manager
Cornell Login Authorization Handler
: Cornell Lab account management system
1 : login( )
2 : redirect(ADDR)
3 : login(UNAME, PASSWD)
alt
[login OK]
4 : redirect(ADDR, COOKIE)
5 : pageRequest(ADDR,COOKIE)
6 : authorizeUser(COOKIE, OPID)
7 : authorize(COOKIE,OPID)
8 : authorization
alt
[authorization OK]
9 : OK
10 : showPage( )
[authorization fail]
11 : fail
[login fail]
12 : Message(invalid login data)

Figure 7: Sign in use case diagram showing the internal interface between Website Manager and Cornell Login Authorization Handler components of the system

The Interface Between the Cornell Login Authorization Handler and the Website Manager: Every time a user requests to do some action related to FeederWatch system, their authorization level is asked to Cornell Lab Server. This is done by a separate component named Cornell Login Authorization Handler. Website Manager uses Cornell Login

Authorization Handler’s interface just by sending the Cookie that came from the user and the ID of the operation which user requested. The authorization information is then sent back to Website Manager.

Design Rationale:

- Since FeederWatch system does not store any information about users, it has to be asked to Cornell Lab Server. For every operation this has to be done; therefore, there is a need for a separate component which only handles that.
- The only two information that is sent from Website Manager to Cornell Login Authorization Handler is the cookie, which were initially given to user by Cornell Lab Server, and the ID of the operation since Cornell Lab Server already knows the owner of the cookie.

4.4.2. External Interfaces

4.4.2.1. User Interfaces

All user interfaces of Project FeederWatch are accessed through the FeederWatch website. This includes the user interface for IT staff who can access the Admin Panel by signing in with their staff IDs and data analyzers who can access the Data Access and Analysis Interface the same way. The website provides a different array of user interfaces for different levels of membership. All interfaces are explained in more detail below with mockups provided for the most critical ones. The website is designed so that the menu is accessible from everywhere.

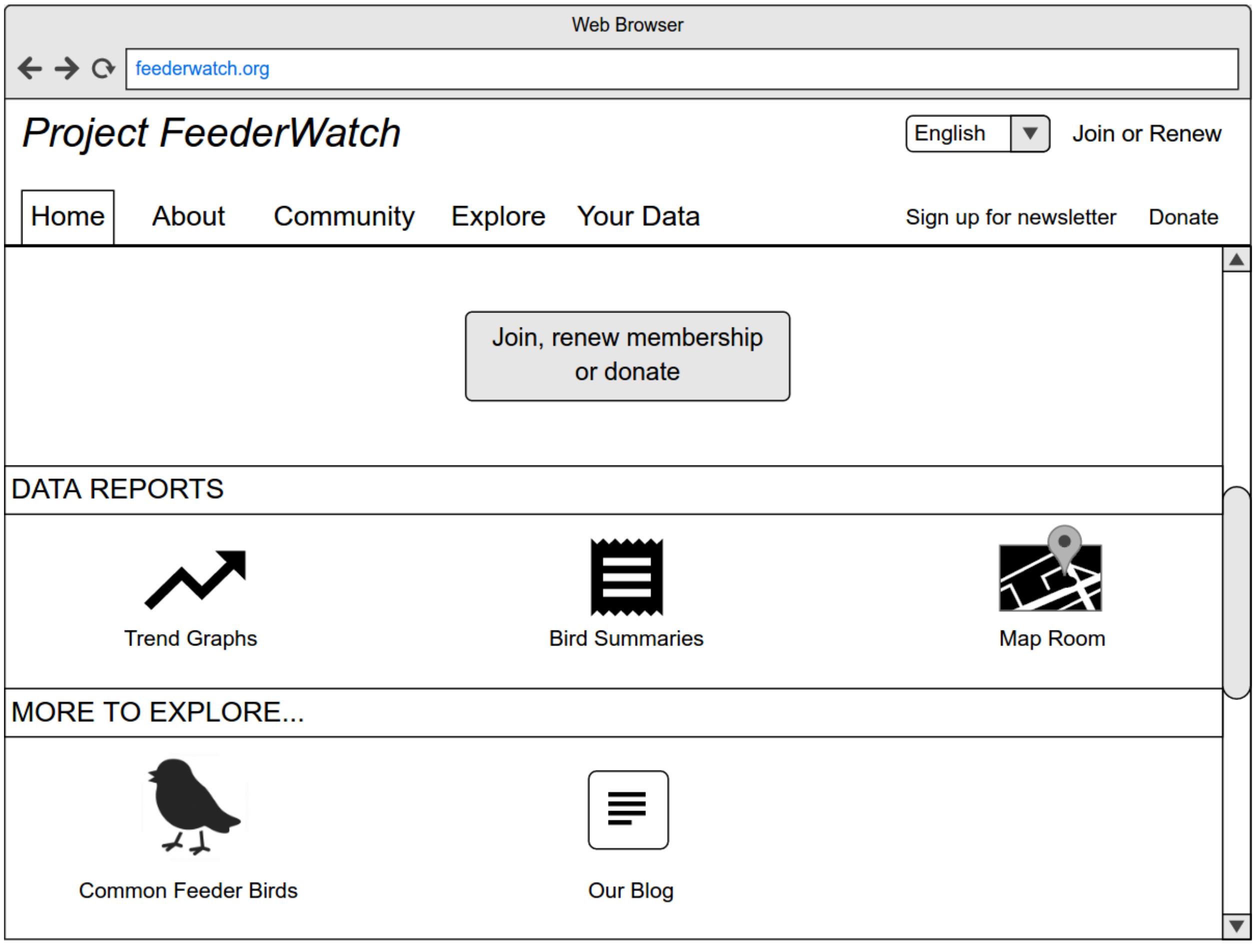
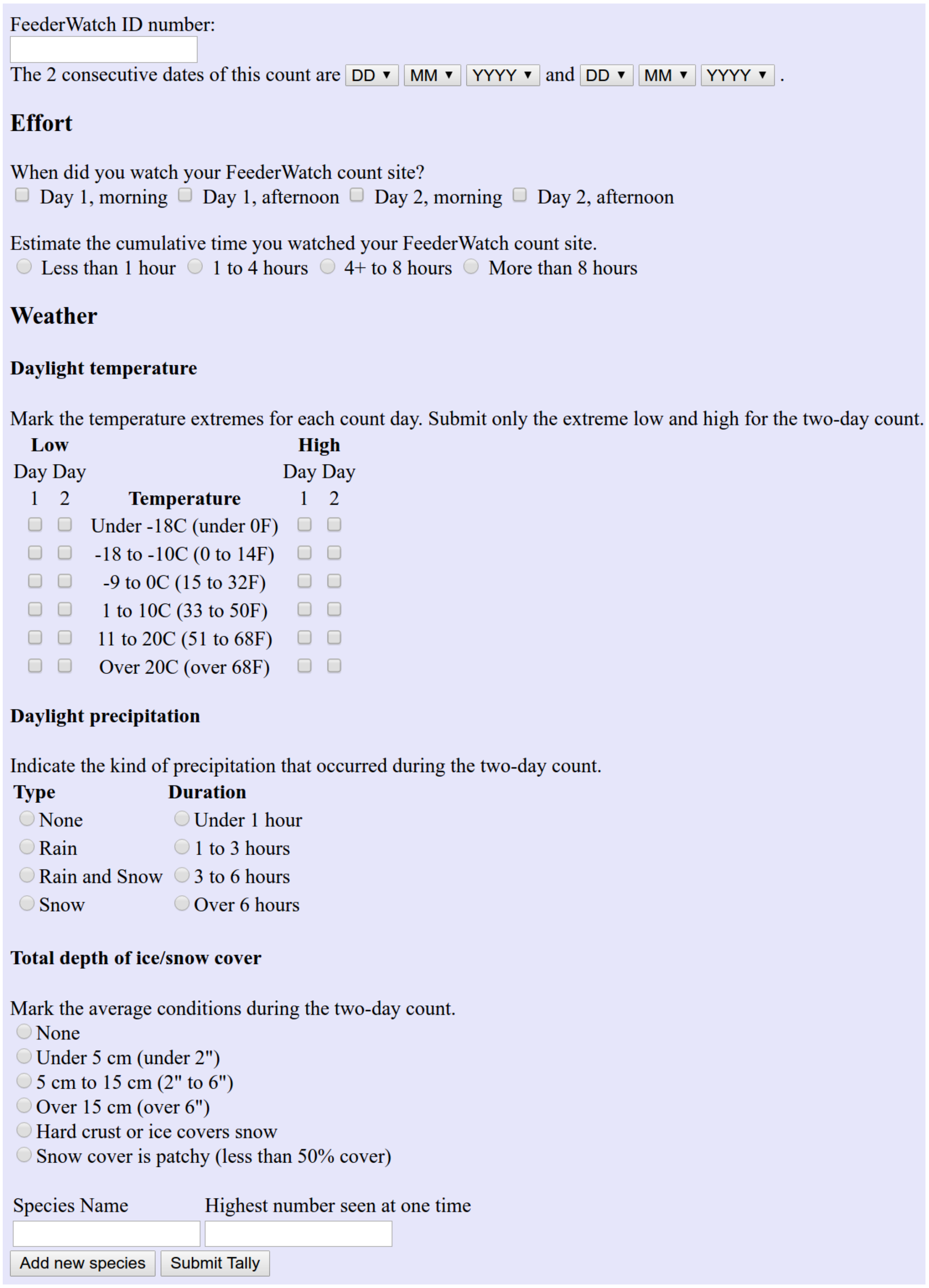


Figure 8: Home page of the FeederWatch Website

Home page: The most used functionalities of FeederWatch are directly accessible from the home page without using the main menu. All functionalities are accessible through the tabular menu which always sits on the top of the website. Clicking the “Join, renew membership or donate” button redirects to the page where the user can choose to join, renew their account, donate or gift someone a membership through Bird Studies Canada or Cornell Lab. This is implemented in “Create or renew paid account on Bird Studies Canada”, “Donate on Bird Studies Canada”, “Gift membership on Bird Studies Canada”, “Create or renew paid account on Cornell Lab”, “Donate on Cornell Lab” and “Gift membership on Cornell Lab” use cases.

Design Rationale:

- The main menu is designed in a tabular fashion so that every functionality of the website can be reached using at most 3 buttons.
- The “Join, renew membership or donate” button is in the center of the page since we want new users to be able to easily join FeederWatch Project.
- The site is offered in English and French. The language of the site can be changed in every page. The reason of supporting also French is that there can be users living in Canada who speak French.



FeederWatch ID number:

The 2 consecutive dates of this count are DD ▾ MM ▾ YYYY ▾ and DD ▾ MM ▾ YYYY ▾ .

Effort

When did you watch your FeederWatch count site?
☐ Day 1, morning ☐ Day 1, afternoon ☐ Day 2, morning ☐ Day 2, afternoon

Estimate the cumulative time you watched your FeederWatch count site.
○ Less than 1 hour ○ 1 to 4 hours ○ 4+ to 8 hours ○ More than 8 hours

Weather

Daylight temperature

Mark the temperature extremes for each count day. Submit only the extreme low and high for the two-day count.

Low			High	
Day 1	Day 2	Temperature	Day 1	Day 2
☐	☐	Under -18C (under 0F)	☐	☐
☐	☐	-18 to -10C (0 to 14F)	☐	☐
☐	☐	-9 to 0C (15 to 32F)	☐	☐
☐	☐	1 to 10C (33 to 50F)	☐	☐
☐	☐	11 to 20C (51 to 68F)	☐	☐
☐	☐	Over 20C (over 68F)	☐	☐

Daylight precipitation

Indicate the kind of precipitation that occurred during the two-day count.

Type	Duration
○ None	○ Under 1 hour
○ Rain	○ 1 to 3 hours
○ Rain and Snow	○ 3 to 6 hours
○ Snow	○ Over 6 hours

Total depth of ice/snow cover

Mark the average conditions during the two-day count.
○ None
○ Under 5 cm (under 2")
○ 5 cm to 15 cm (2" to 6")
○ Over 15 cm (over 6")
○ Hard crust or ice covers snow
○ Snow cover is patchy (less than 50% cover)

Species Name   Highest number seen at one time

Figure 9: Tally Sheet Submission Interface

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. 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. Paid member users are able to use “Enter observation data” and “Review observation data” use cases through this interface.

Design Rationale:

- The objective with this interface is to have a clean and simple form which is easy to use by all user profiles. There aren’t any complicated menus or buttons.
- The interface collects all needed information while dividing the data into categories to make the storage of data easier.

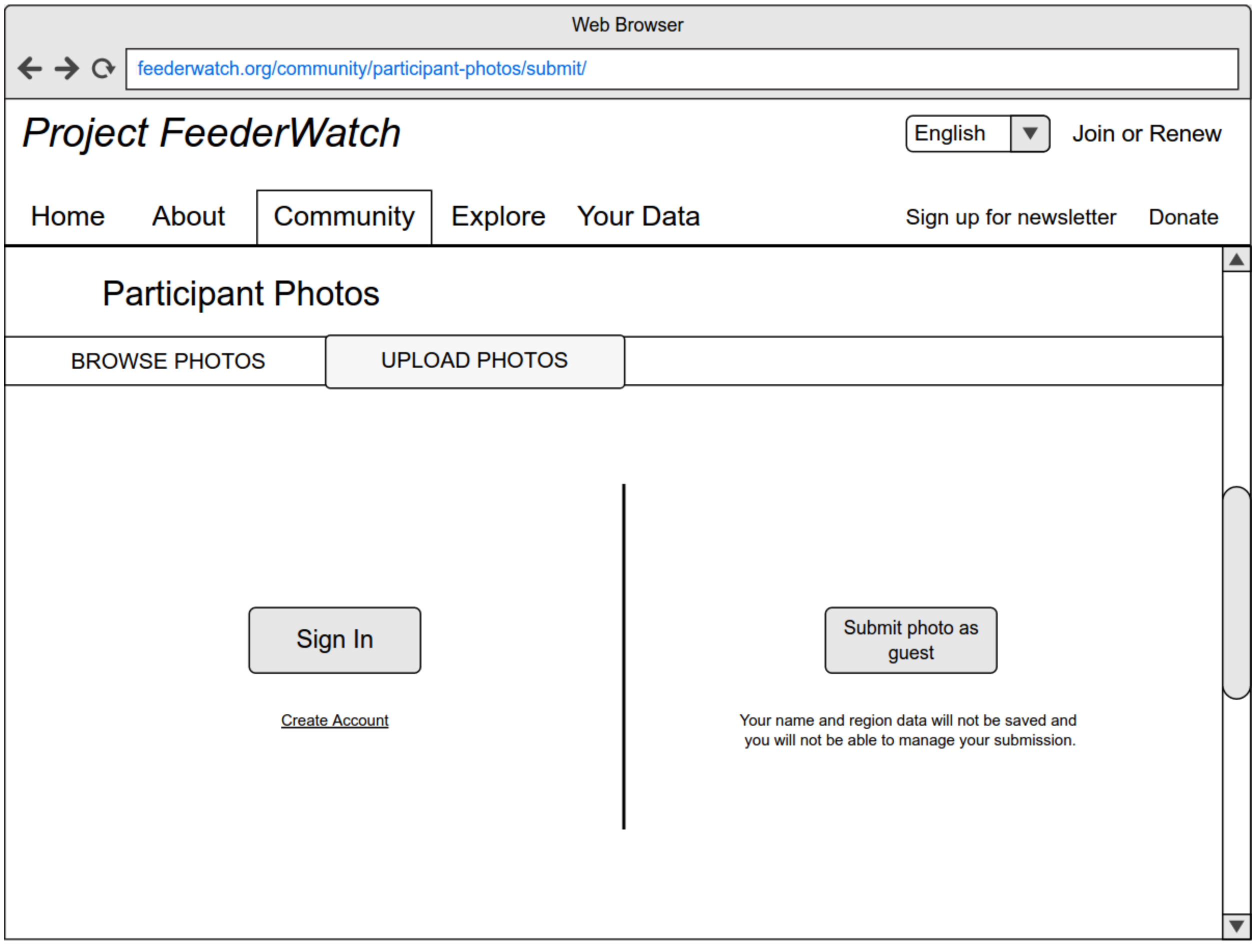


Figure 10: Uploading a photo as a member user or as a guest in “Participant Photos” section on “Community” tab

Photo Browsing and Uploading Interface: Users can upload their photos to FeederWatch website through this interface. 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. These two tabs correspond to the use cases “Browse photo” and “Upload photo” and “Sign In” option corresponds to “Sign in” use case.

Design Rationale:

- Since these two functionalities, browse and upload photos, are very related, users can pass from one to the other by changing the tab sits just below the header of the page.
- Providing already filled forms while uploading photos encourages the users to create accounts. This corresponds to “Get membership” use case.
- “Back to top” button is available to the users when they are browsing photos since there may be a big amount of photos and it may get hard to go back to the top of the page.

Data Access and Analysis Interface: This interface can only be reached when a user is signed in as a data analyzer. Through this interface, data analyzers can filter and download 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 which are handled by “Access data” and “Add statistical data” use cases.

Design Rationale:

- The functionality of this interface is fairly simple; download raw data from database and upload processed data to website, so the design reflects this.
- As this interface is only available for staff members, the aesthetic details aren’t of importance.
- The filtering of raw data is done graphically through the interface considering statisticians might not be able to use SQL to query the database themselves.

Admin Panel Interface: This interface is only available to users who sign is with appropriate staff ID. This is where IT staff can see error logs, which is implemented by the “View error logs” use case. Staff can also 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.

Design Rationale:

- Like the Data Access and Analysis Interface, looks aren’t a big issue for the admin panel as it will also be used by staff members only.
- Also like the Data Access and Analysis Interface, filtering and extracting data, and querying the database is done graphically through the system for ease of use and to avoid the possible data corruption that could result from the staff having a direct access to the database.

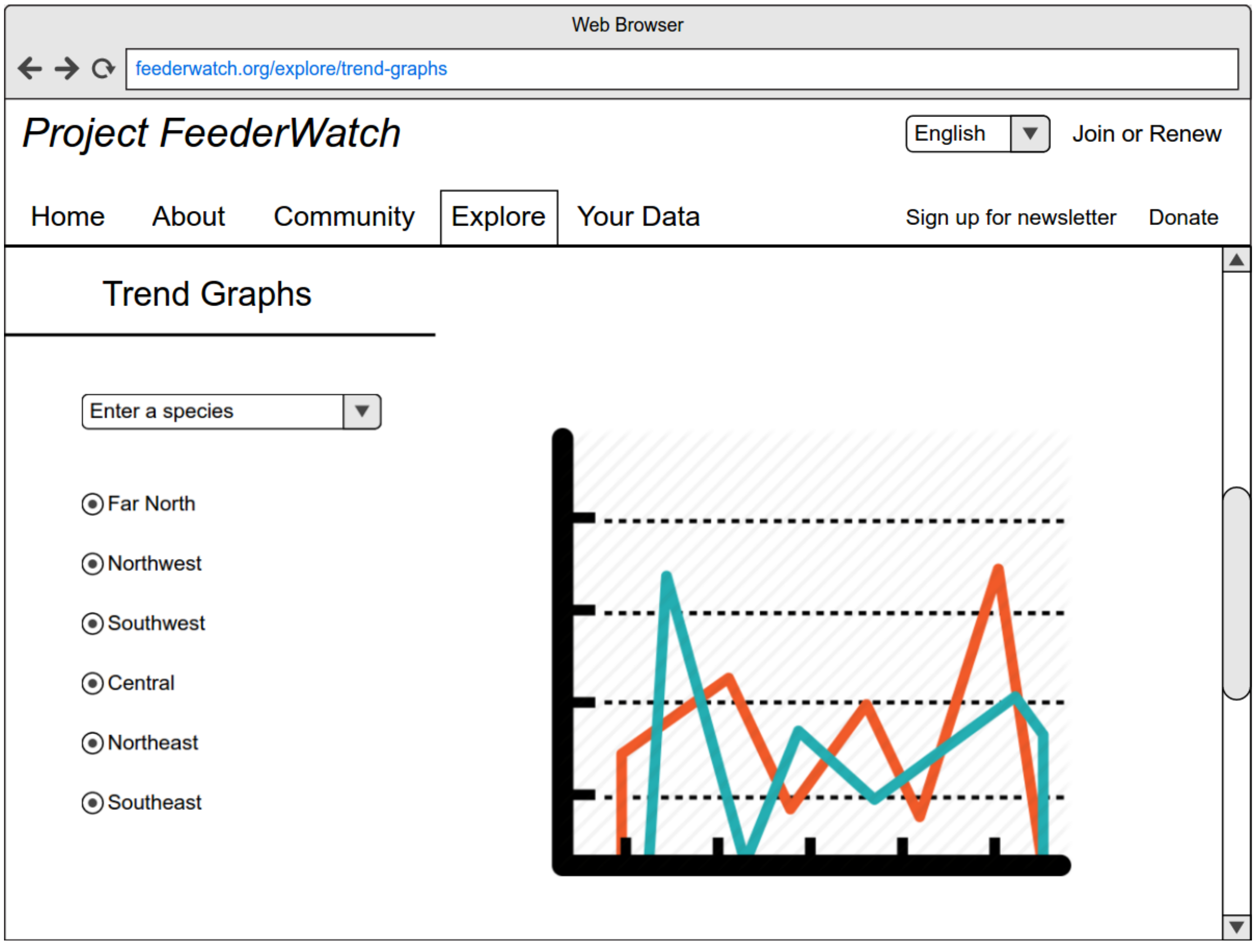


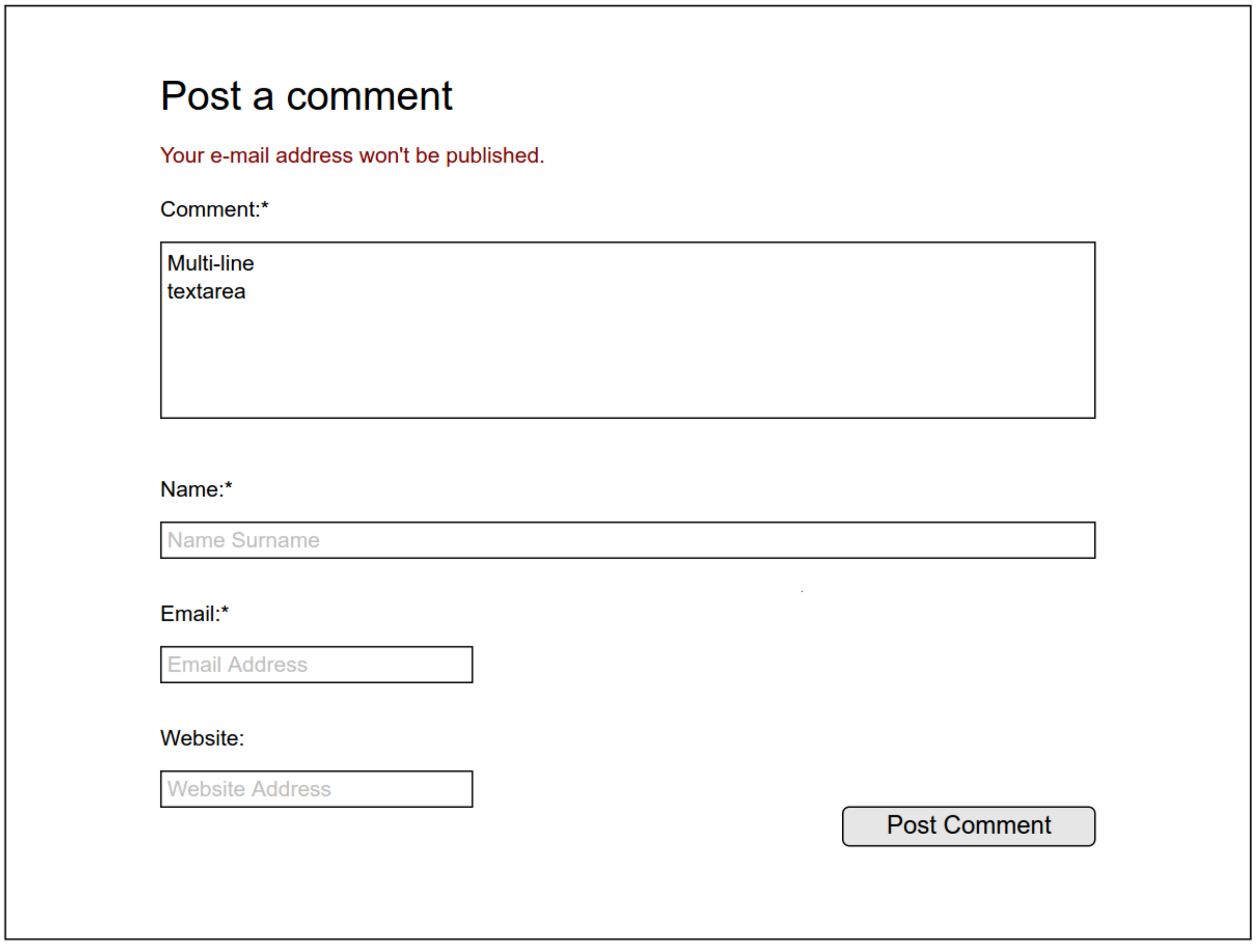
Figure 11: Viewing and filtering “Trend Graphs” on “Explore" tab

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. Users can see line graph representation of number of birds changing through the years in “Trend Graphs” section (Figure 11). These graphs are created by Data Analyzers. Users can filter these line graphs by species and region. In “Bird summaries section”, users are able to see the data of birds according to state and year in tabular form. In “Map room” section users can see distribution of a bird species in a selected year on a real map. The map in this section is provided by Google Maps and communication with its API is done by “Google Maps

Handler". The use case lies under all of these sections is "View statistical data".

Design Rationale:

- The reason of putting all of these sections into one tab, namely "Explore" tab, is that they are all related to the same use case, and being under the same use case is because they all query the database tables which were filled with the data provided by Data Analyzers.
- Being put into the same tab also simplifies the usage and therefore, users will be able to learn the website with ease.


Post a comment

Your e-mail address won't be published.

Comment:*

Multi-line
textarea

Name:*

Name Surname

Email:*

Email Address

Website:

Website Address

Figure 12: Posting comment to a blog post in "Our Blog" section on "Community" tab

Blog Interface: This is a simple blog interface where Blog writers can post their articles and Users can view and comment on them. “Post blog entry” use case can be used by Blog Writers to do this. While posting blog entries, Blog Writers can see the fields they should fill and they can also see the “Upload File” box through which they can upload their articles. Users can filter posts by their publishing month and categories. Users are also able to post comments to a blog post and this is provided by “Comment on blog post” use case. While posting a comment users are expected to provide their names and email addresses aside from the comment. They can also give their website URL optionally.

Design Rationale:

- “*” characters are written next to the required fields and without filling these fields, users or blog writers cannot proceed with their process.
- The email address of the user who posts a command is not shown to the other users due to regulatory policies. And user is informed about this.

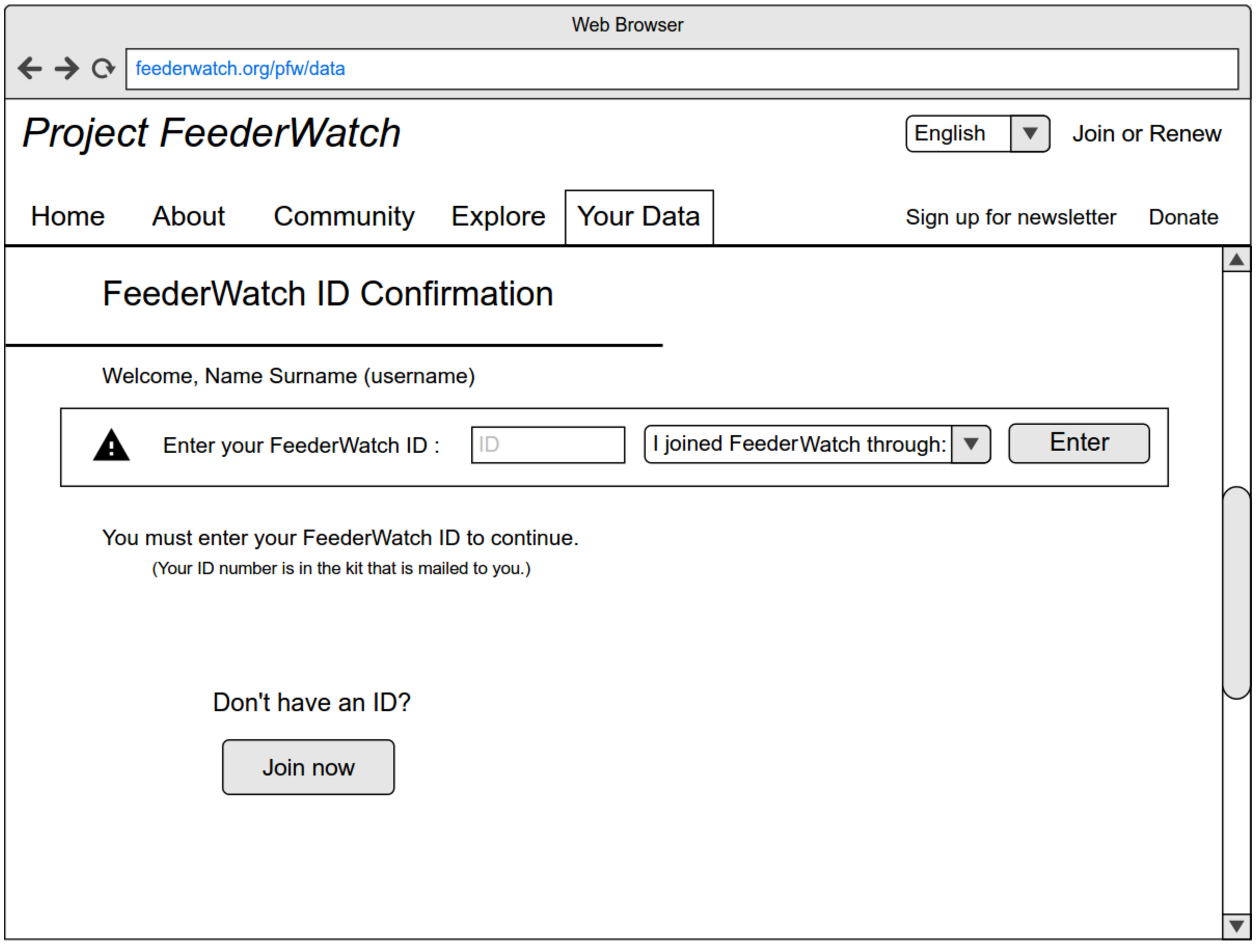

Web Browser
feederwatch.org/pfw/data
Project FeederWatch
English
Join or Renew
Home
About
Community
Explore
Your Data
Sign up for newsletter
Donate
FeederWatch ID Confirmation
Welcome, Name Surname (username)
Enter your FeederWatch ID :
ID
I joined FeederWatch through:
Enter
You must enter your FeederWatch ID to continue.
(Your ID number is in the kit that is mailed to you.)
Don't have an ID?
Join now

Figure 13: FeederWatch ID Confirmation on “Your Data” tab

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. This interface corresponds to the “Confirm FeederWatch ID” use case.

Design Rationale:

- As “Your Data” is the only tab a user needs to be a paid member user to use, they will only be prompted to confirm their paid membership if they try to access this tab. If they are a member user and they are signed in they will be able to see the FeederWatch ID Confirmation Interface. Otherwise they will be redirected to Cornell Labs website to sign in.

- The user can select the platform through which they joined the project in the dropdown menu as users can join through Bird Studies Canada or Cornell Labs.
- What a FeederWatch ID is and what the user needs to do to acquire one is also explained on the page to make things clear.

4.4.2.2. System Interfaces

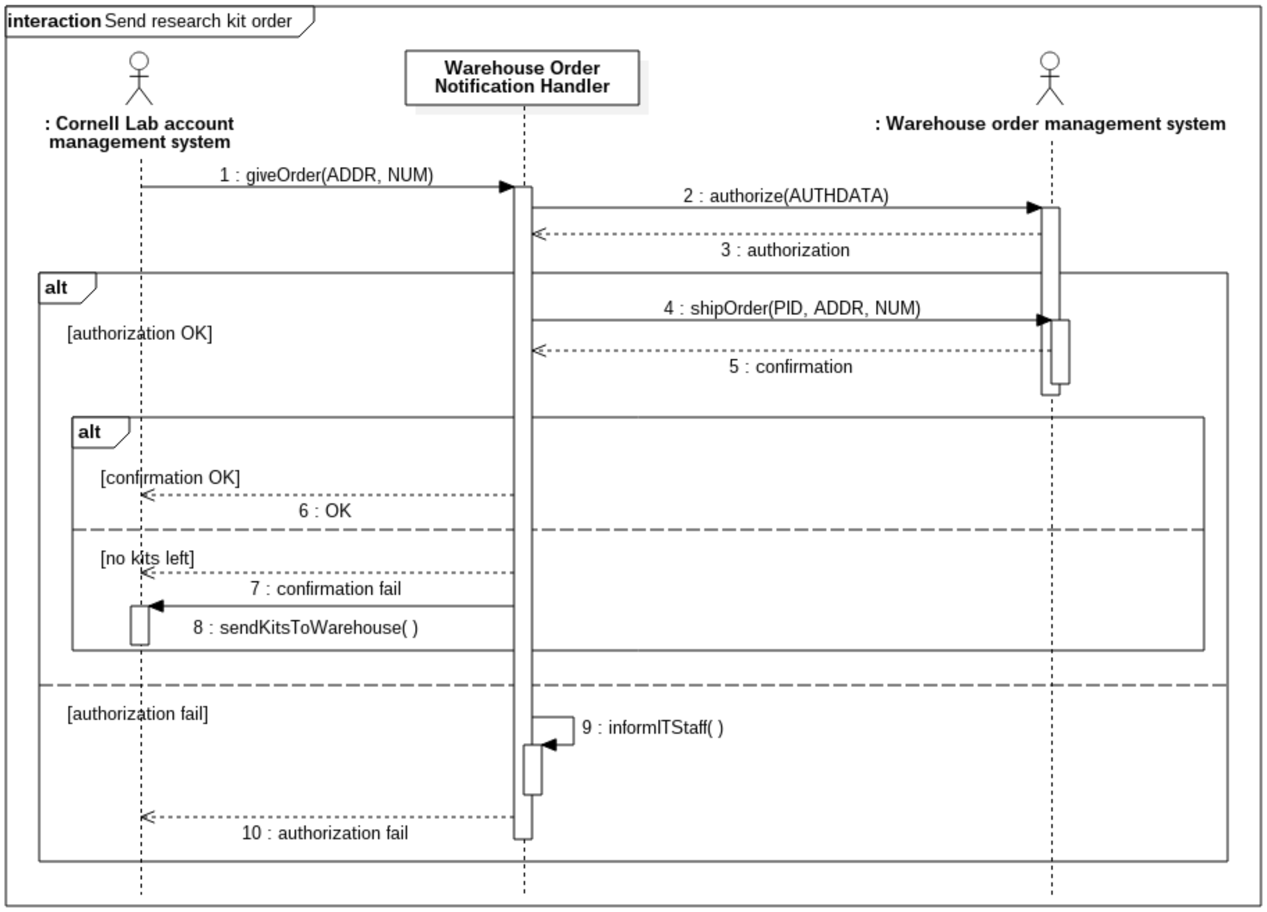


Figure 14: Send research kit order sequence diagram showing the interfaces between Warehouse Order Notification Handler and external systems Cornell Lab Account Management System and Warehouse Order Management System

The Interface Between the Warehouse Order Notification Handler and the Cornell Lab Server: Cornell Lab notifies the Warehouse Order Notification Handler component of the FeederWatch system when a new users join the project for the users' research kits to be sent out. Address and quantity data is provided by Cornell Lab in this exchange, for Warehouse Order Notification Handler to be able to notify the Warehouse where the research kits are kept. The result of the interaction of Warehouse Order Notification Handler and the warehouse is reported to Cornell for them to take action if there are no kits left at the warehouse. The related use case is "Send research kit order".

The Interface Between the Warehouse Order Notification Handler and the Warehouse System: The Warehouse Order Notification Handler component of the FeederWatch system communicates with the Warehouse Server through this interface. Upon the request of Cornell Lab, Warehouse Order Notification Handler first sends authorization data to Warehouse system to get authorized to make an order. If the authorization is OK, Warehouse Order Notification Handler sends the package ID of the kits and address and quantity data obtained from Cornell. If there are no problems with the order, the Warehouse System confirms; if there are no kits left at the warehouse, the Warehouse System informs The Warehouse Order Notification Handler. If the authorization fails, the Warehouse Order Notification Handler informs the IT Staff. The related use case is "Send research kit order".

Design Rationale:

- Cornell doesn't directly communicate with the Warehouse and this is handled by a component of the FeederWatch

System, as the research kits are only for FeederWatch Project.

- The authorization data that the system needs to be able to communicate with the warehouse is kept in the Warehouse Order Notification Handler component, the system only communicates with the warehouse through this component.
- Since Cornell Lab handles the money side of the project, they are responsible for ordering new research kits from the manufacturer to be sent to the warehouse when there are no kits left at the warehouse.

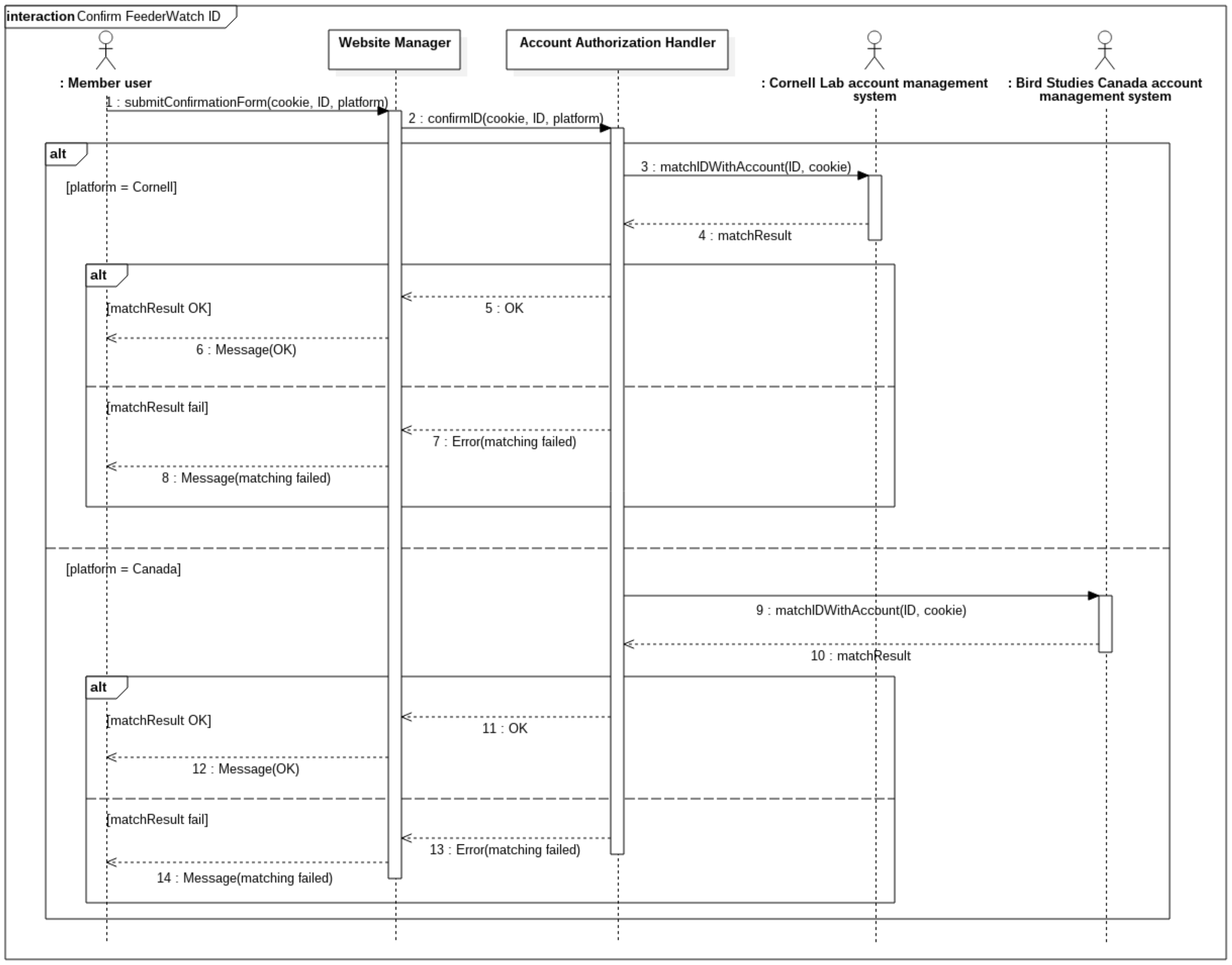

interaction Confirm FeederWatch ID
: Member user
Website Manager
Account Authorization Handler
: Cornell Lab account management system
: Bird Studies Canada account management system
1 : submitConfirmationForm(cookie, ID, platform)
2 : confirmID(cookie, ID, platform)
alt
[platform = Cornell]
3 : matchIDWithAccount(ID, cookie)
4 : matchResult
alt
[matchResult OK]
5 : OK
6 : Message(OK)
[matchResult fail]
7 : Error(matching failed)
8 : Message(matching failed)
[platform = Canada]
9 : matchIDWithAccount(ID, cookie)
10 : matchResult
alt
[matchResult OK]
11 : OK
12 : Message(OK)
[matchResult fail]
13 : Error(matching failed)
14 : Message(matching failed)

Figure 15: Confirm FeederWatchID sequence diagram showing the interfaces between Warehouse Order Notification Handler and external systems Cornell Lab Account Management System and Bird Studies Canada Account Management System

The Interface Between the Account Authorization Handler, the Cornell Lab Server and the Bird Studies Canada Server: Upon the ID confirmation request from the Website Manager and according to which platform was chosen by the user, Account Authorization Handler either contacts with Bird Studies Canada or Cornell Labs through this interface. The Account Authorization Handler matches FeederWatch ID with account by sending the information received from Website Manager with a cookie. Cornell Lab Server or the Bird Studies Canada send the match result back for the Account Authorization Handler to inform back the Website Manager. This is implemented in “Confirm FeederWatch ID” use case.

Design Rationale:

- Since FeederWatch System doesn’t store user data, the FeederWatch ID needs to be passed to Cornell Lab or Bird Studies Canada through this interface for the ID to be matched with their records and confirmed.

The Interface Between the Cornell Login Authorization Handler and the Cornell Lab Server: Through this interface, the cookie and operation ID that was provided by Website Manager is sent to Cornell Lab Server to get authorized. Most of the use cases require use of this interface.

Design Rationale:

- Since the data needed for authorization isn’t kept in FeederWatch Servers, for every operation request made by the user, Cornell Login Authorization Handler must ask Cornell Lab Server.

The Interface Between the Google Maps Handler and Google Maps: Using the query result provided by Website Manager, Google Maps Handler makes Google Maps draw using Google Maps API with the Map Data it sends. Then, Google Maps returns the result. “View distribution of birds” use case is related to this interface.

Design Rationale:

- To be able to use the Google Maps to show statistical data on graphs on the website, Google Maps Handler provides communication with Google Maps using its API.