CSC 431

EcoDex

System Architecture Specification (SAS)

Team 03

Michael Castellucci	Scrum Master
Carter Falkenberg	Developer
Phuong Uyen Dang	Developer

Version History

Version	Date	Author(s)	Change Comments
1.0	4/4	Michael, Carter, Uyen	First draft
1.0	4/5	Uyen	Continuing first draft

Table of Contents

Table of Figures

1. System Analysis

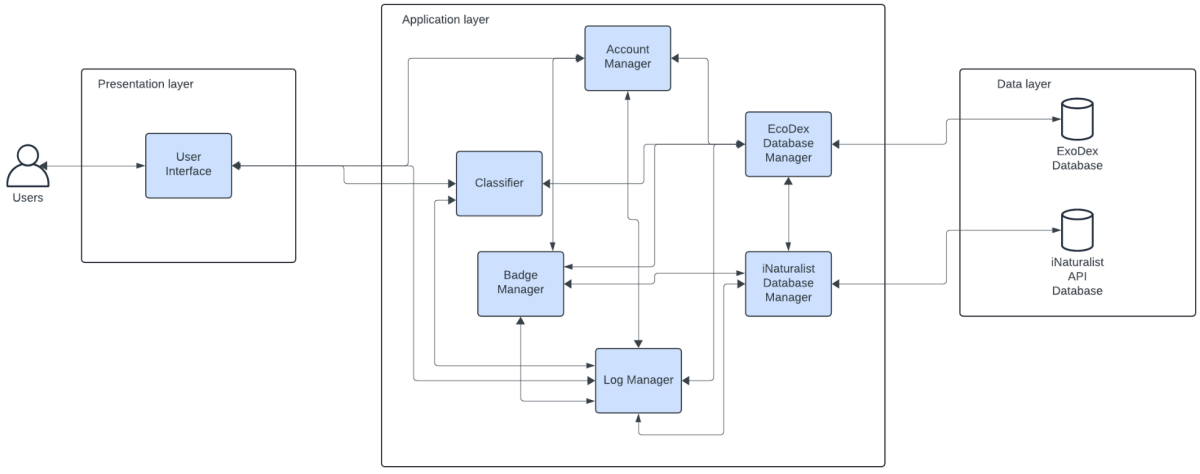
1.1 System Overview

This document describes the architecture design and specification of the EcoDex flora and fauna recognition and social game app. EcoDex aims to educate and raise awareness about species of flora and fauna for the protection of ecosystems. The application has the following main functionalities:

- Identifying a species of flora or fauna by either a photo upload of it or via the in-app camera feature.
- Connecting users to other users who might share the same interests.
- Displaying a logs map of regional flora and fauna.
- Informing users about interesting and relevant facts about a flora or fauna (such as its uses in medical history, etc.)

The application uses a three-tier architecture, including a presentation, application, and data tier. The presentation will include a mix of a social media style feed as well as a Pokémon style deck of unlocked species. There will also be a map consisting of species that have been identified around the globe. The application will consist of the logic, utilizing the machine learning model to classify, retrieving the proper photos for the feed, and linking the map to the correct location and species data. The data tier will store any data required for the app to function. This includes but is not limited to user information, stored photos, weights for the machine learning model, friends lists, and more.

1.2 System Diagram



1.3 Actor Identification

Actors interacting with the system include:

- New user: The user who has not previously had an account with EcoDex.
- Registered user: The user who has established an account with EcoDex and can therefore use it to log into the system.
- Moderation team: EcoDex moderators who are tasked with handling reported messages and posts. Their responsibilities regarding these reported items are evaluating whether deletion, bans, or suspensions are necessary, and performing these actions as required.
- Platform server: The central database containing the machine learning algorithm, photos, maps, and metadata associated with user accounts of the application.

1.4 Design Rationale

1.4.1 Architectural Style

EcoDex will utilize a 3 tier architectural system.

1) *Presentation tier:* This tier is responsible for handling the user interface elements of the system, including the style of messages, feed presentation, and logs maps.

2) *Application tier:* This tier is where the logic of the system operates and data manipulation is executed. An example of an operation within this tier would be the identification of the species of a flora or fauna by a photo submitted by the user.
3) *Data tier:* This tier houses the application-related components (ML algorithm, species information, etc.) and user-specific data (usernames, encrypted passwords, etc.). It also manages the storage and retrieval of this information.

1.4.2 Design Pattern(s)

The design patterns that would be considered for implementation are as follows.

1) *Factory Method:* There might arise instances for the creation of objects and components based on user actions or system requirements. The Factory Method design pattern can provide a way to delegate the object creation process to subclasses, allowing for the flexible management of a variety of object types.
2) *Façade:* Since the application encompasses underlying subsystems for message styles, feed presentation, data manipulation, and data storage, the Façade pattern can provide a simplified and unified interface to these subsystems which could be complex on their own. It can encapsulate the interactions with these components and present a unified interface to the higher-level application logic.

1.4.3 Framework

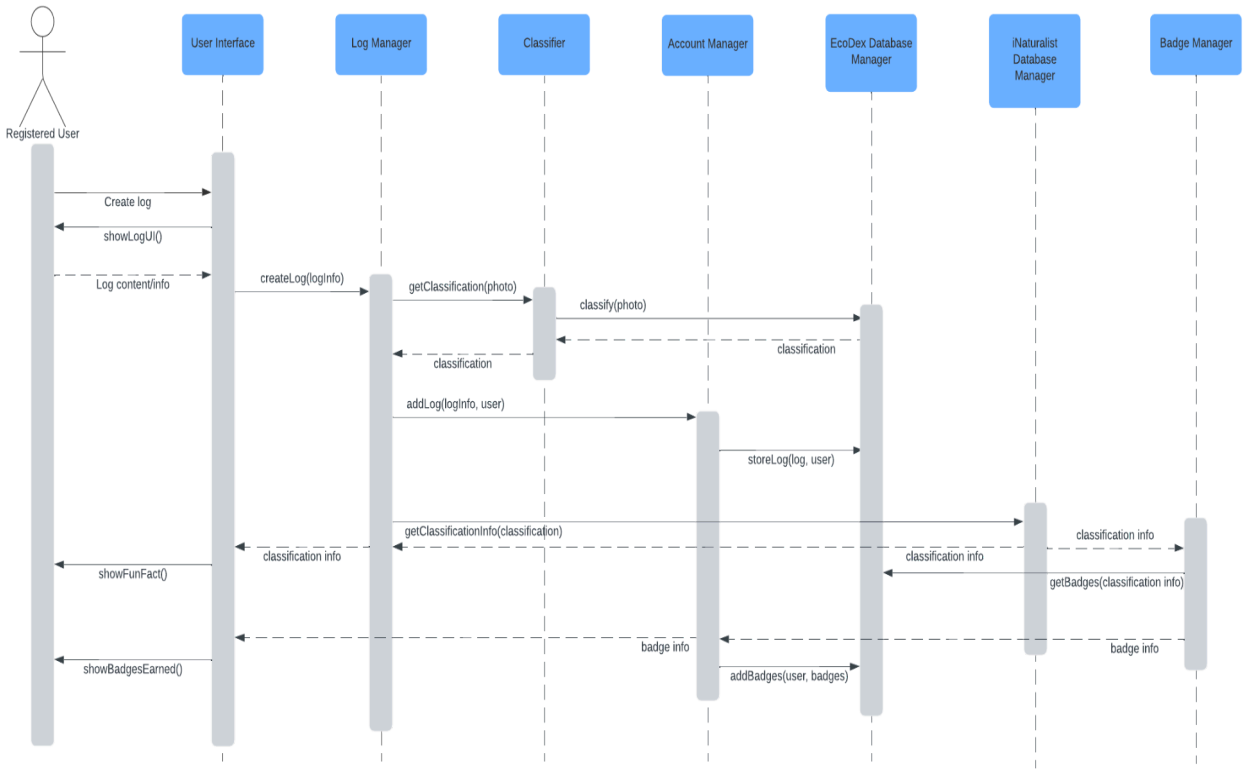
All frameworks we are using are free, due to limited budget. Therefore, we have selected frameworks that we believe align with our project requirements, budget constraint, and team expertise.

1) *Machine learning algorithm:* We will be using the PyTorch framework for the machine learning algorithm. This framework is open-source and allows for heavily customizable machine learning model architecture, which is ideal for the optimization of a CNN-based model for species recognition. Its rich ecosystem of libraries and tools will enable us to optimize our model architecture efficiently.
2) *Front end:* React will be used for the front end as it is versatile and robust, and allows for integrated use with Android and IOS. Since we do not have much experience with front end, we chose React due to the countless templates and rigorous documentation online, making it an ideal choice.

3) *Back end:* Django will be used for the back end as it allows for simple Pythonic integration with our machine learning model and is capable of scaling appropriately for the app. It will also be compatible with our machine learning algorithm, which will be implemented in Python using the PyTorch framework.
4) *Species references and information:* IPyNaturalist will be utilized for the access of species references and information as it leverages from the very popular app, INaturalist. Instead of directly using the INaturalist API, we are utilizing IPyNaturalist due to it being in Python, which integrates well with PyTorch and our skill sets as developers.

2. Functional Design

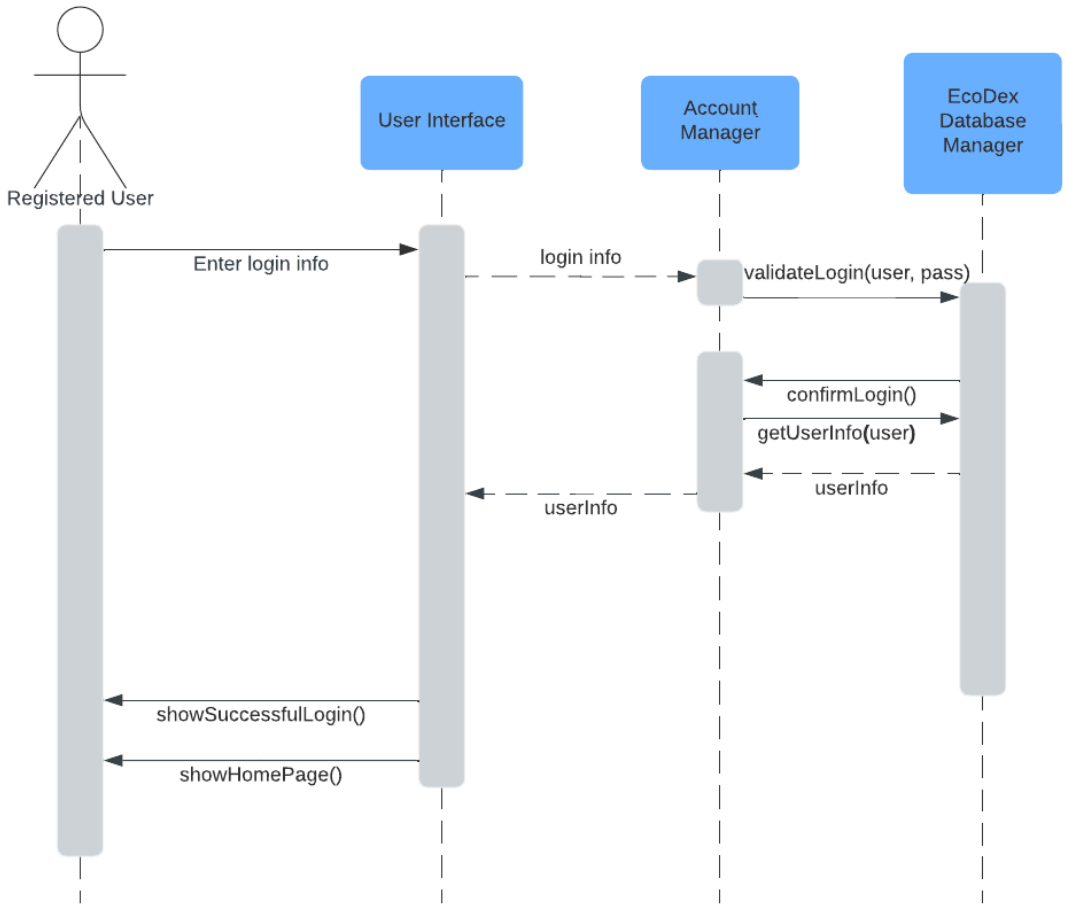
2.1 Create Log Sequence Diagram



- Creating Logs is one of the main functionalities of EcoDex, allowing users to create posts and earn badges from them.
- To begin, the user interacts with the User Interface (UI) to prompt a log creation which then shows the user the log creation UI by calling showLogUI().
- Once the user enters the information pertaining to their Log, the UI calls createLog() and sends information to the Log Manager.
- Once the Log Manager receives the information, it requests a classification for the user's photo by calling getClassification(). The Classifier then works with the EcoDex Database Manager to return a classification to the Log Manager.
- The Log is then added to the user's account when the Log Manager calls addLog(). This causes the Account Manager to store the log in the EcoDex Database using the EcoDex Database Manager
- Information about the identified organism (such as fun facts) are then requested by the Log Manager from the iNaturalist Database Manager. This information is distributed back to the Log Manager to be displayed to the user through the UI.

The classification info is at the same time distributed to the Badge Manager to see what badges the classification earns the user. This badge information is then distributed back to the Account Manager to update the user's badges. Afterwards, the new badge earned is displayed through the UI using showBadgesEarned().

2.2 Login Sequence Diagram



- Logging into EcoDex is required for all other functionality, so it is essential to the user experience.
- To begin, the user enters their login credentials for the UI to send to the Account Manager. From here, the Account Manager attempts to validate the login credentials with the EcoDex Database Manager. The EcoDex Database Manager then confirms or denies the login.
- If the login is confirmed, the Account Manager requests the user's account information, which is then sent to the User Interface. The User Interface then uses this information to display the user's home page after showing the login was successful.

3. Structural Design