Module 5- Computer Systems (2023-24)

Project

UNIVERSITY OF TWENTE.

Project Name: Bird Activity Detection	***Team Members:*** Lucas Fuertes, Thomas van der Boon, Dinh Thuy Nhat Vy, Carmen Asbreuk, Dimitri von Benckendorff, Duong Thu Huyen
Team ID: 9	***Mentor(s):*** Denis Krylov, Reneta Trifonova

Sr. No.	Review Security Architecture	Put checkmark ✓ if you have completed the Review Security Architecture as suggested in the left column	Additional comments (If required)	Security Controls/Countermeasures	Put checkmark ✓ if you have completed the Security controls points as suggested in the left column	Additional comments (if required)
1	Check the Trust Boundaries: the interfaces between the raspberry pi and database, database and server, and server database-server and server-API interfaces	✓		Check the prevention criteria: use strong authentication and encryption for the rpi-database and database-server interfaces and use an open-source APIs for weather data, and more	✓	
2	Identify data flows: and design for secure data transfers between components that use data as input and/or output in our system.	✓		Check the mitigation criteria to reduce the impact of the risk/threat for the application: attackers are slowed by requiring users to already define strong passwords and the storage is more secure because we add a salt to all passwords before hashing and storing them.	✓	
3	Entry and Exit points of the system and its components.	✓		Make a data flow diagram to visualize and understand the data flow, input, output points, and trust boundary.	✓	

4	**Write the complete architecture in the SDD template. Review and approve among yourselves and by your assigned mentor(s).**	✓		**Analyse the cost involved to implement the security controls (if any).**	✓	
	Team members reviewed:	Thomas van der Boon, yes Dimitri von Benckendorff, yes Luca Fuertes Taweeapiradeemanee, yes Dinh Thuy Nhat Vy, yes Duong Thu Huyen, yes				

1. Introduction

Bird Activity Detector (BAD), as its name suggests, is a product aiming at detecting Birds and taking images of them. It is design in a way that would be beneficial to both hobbyists and researchers, through the ability to classify the birds, display images and display various statistics regarding the specific birds that have appeared. We also wish to upload the sightings to a third-party website called BirdWeather, which is a visualisation of various stations around the World.

2. Functional/Non-Functional Requirements

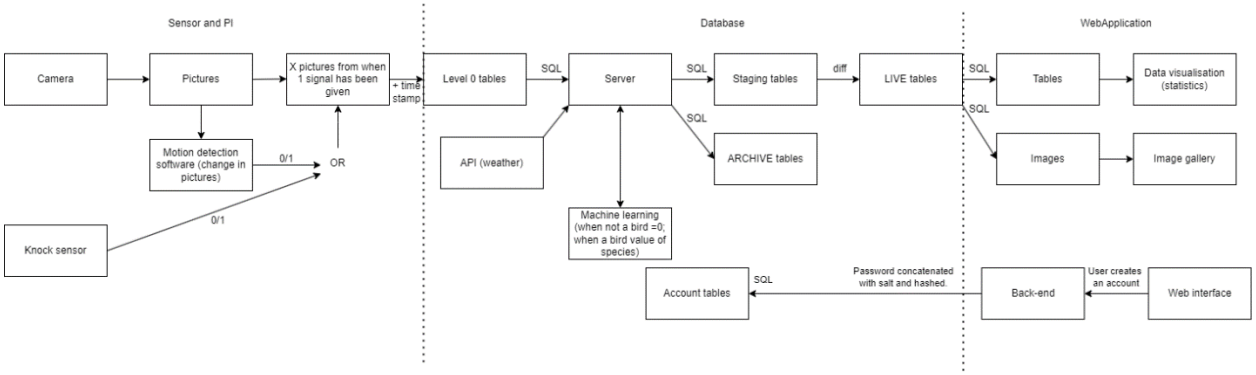
- **The functional requirements of an automated bird activity detector are:**
  - [**H**] Detect a movement near the bird feeder.
  - [**H**] Detect an object on the bird feeder platform.
  - [**H**] Autonomously capture and store images.
  - [**H**] Implement bird species classification from image input.
  - [**M**] Interactive web interface that allows a user to access relevant data of captured and classified birds.
  - [**M**] Visualize the data and enable a user to discern patterns in the web interface

- **The nonfunctional requirements of an automated bird activity detector are:**
  - [**M**] Allow the user to sort and filter through the image gallery on bird species, date, and time.
  - [**M**] Manage metadata including the date and time.
  - [**L**] Archive blurry images or images that do not contain a bird.
  - [**L**] Blur personally identifiable data for privacy of individuals captured on the media.
  - [**L**] Upload collected data to BirdWeather (or another bird sighting database) automatically.

- **The security requirements of an automated bird activity detector are:**
  - [**M**] An authentication layer (with username and password) for the web interface and database to store media in line with privacy.
  - [**M**] Secure storage of data. This includes protection measures for injection attacks and cross-site scripting.

3. Architectural Design

DATA FLOW DIAGRAM

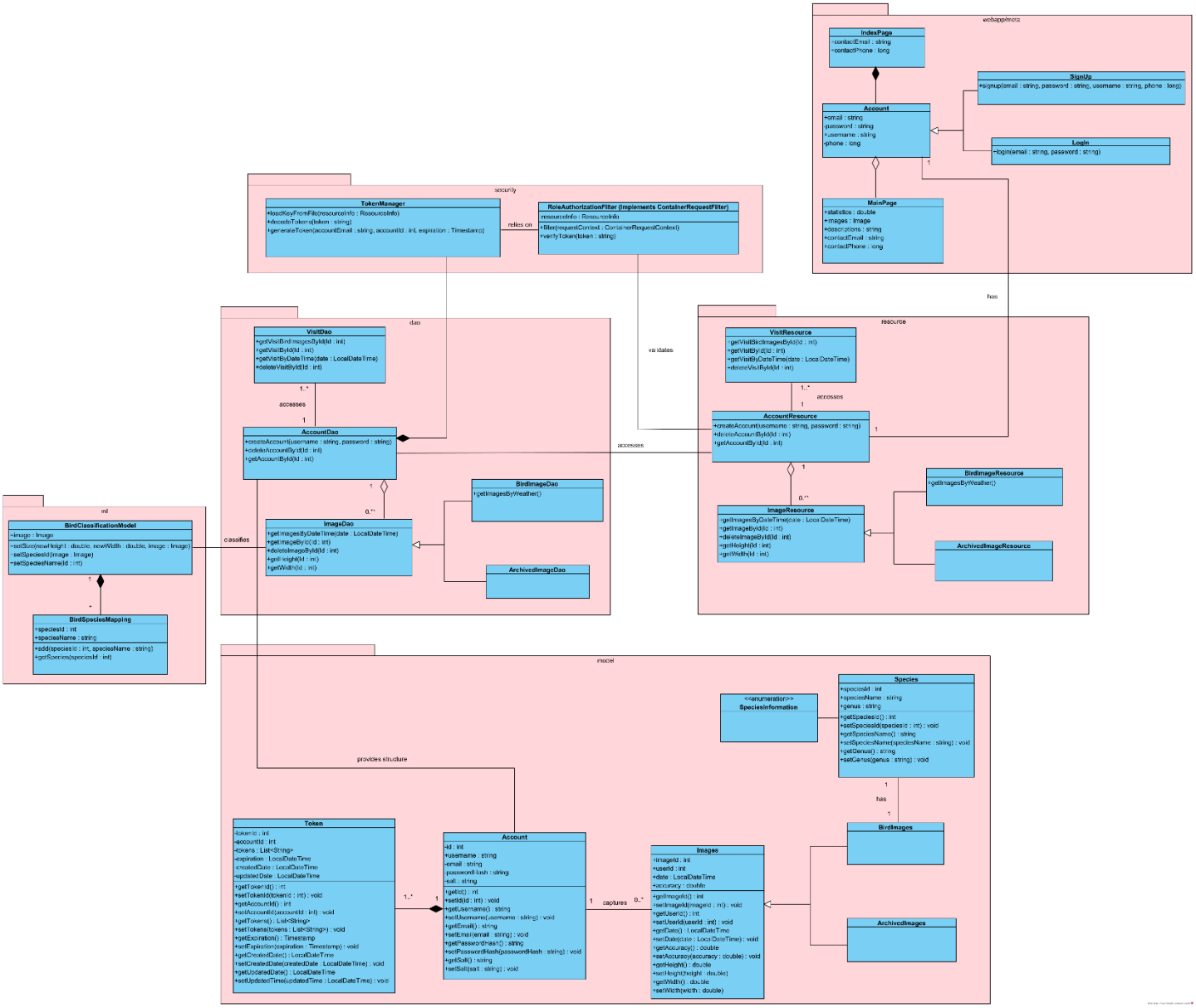


The data flow diagram can be divided by three main parts: Sensor and Pi, Database, and Web Application. The data mainly flows from left to right starting with the input from the camera and knock sensor. The knock sensor and indirectly also the camera give a 0/1 value to set the rest of the data flow in process. Multiple relevant pictures and the timestamp of their capture are sent to the database. A server supplements this data with weather data acquired from an open API and a classification outcome from a machine learning model. If the image quality is sufficient and it captures a bird according to the machine learning model, the image is stored in the staging schema of our database. Otherwise, the image is archived. The staging and live schema of our database is merged to become the new version of our live tables. These tables can be accessed by the web application. The web application takes the data and shows a gallery of images as well as some statistical information.

Furthermore, the security of user authentication is controlled in another data path that is followed when an account is created. The password is salted and hashed and then stored in the database.

With the focus of secure transfer in mind, we have the following thoughts and ideas on secure transfer of data between the raspberry pi, database, and servers. We will limit the raspberry pi such that it can access the database, but it cannot be accessed unless a wired connection is utilized. We will store the device in a safe location to ensure that the team can use a wired connection but nobody else. Accessing the database to read and/or write requires a valid login by the user or server. Accessing the weather data is through an open API. We will sanitize the APIs data to ensure that we add relevant, high-quality data to our database.

CLASS DIAGRAM



The class diagram has 3 primary subsystems, including webapp/meta for the front-end side of the system, database and security for the back-end side, and machine learning.

Firstly, regarding to the webapp/meta, users can create an account by signing up with email and password. Once they already had a valid account, they can login to the main webpage and upload the images. In the context of a web application, we have 4 main packages, which are DAO (also known as Data Access Object), model, resources, and security. For a typical flow, the resources will first receive an HTTP request and uses the DAO to perform data access operations, including retrieving and updating data in the underlying database. After that, DAO will directly interact with the model to retrieve the data to perform CRUD (Create, Read, Update, Delete) operations on the data source. Finally, the resource will use the retrieved data from the model to generate a response, which is sent

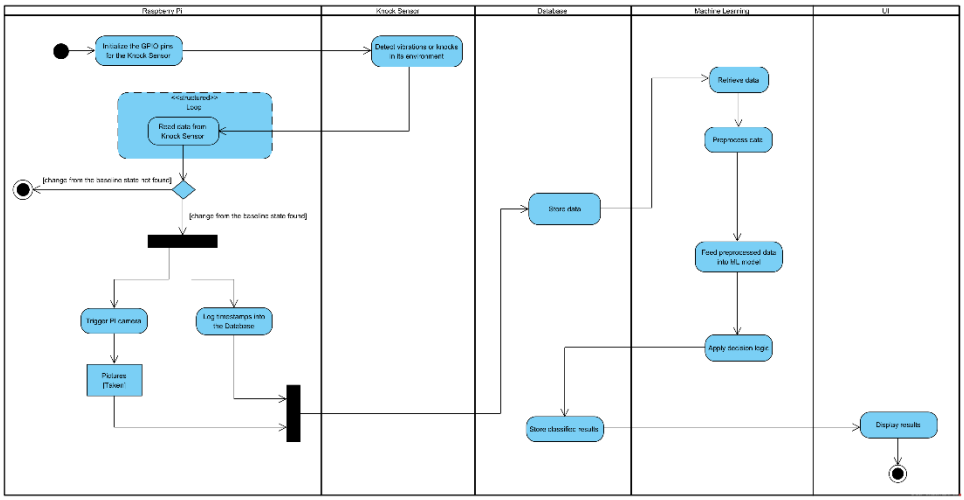
back to the client as the output from the web application. Here, the input data may contain images and their related information, timestamps, along with the information of users' accounts.

To classify the bird species and give more relevant information about the birds, a bird classification model will be trained and utilized, in which the classified results will be stored in the DAO and be displayed to the UI afterwards.

Finally, to have a secure application, "TokenManager" and "RoleAuthorizationFilter" are also implemented. While TokenManager will be responsible for the creation, validation and decoding of security tokens, RoleAuthorizationFilter intercepts incoming request before they reach the resources and enforces access control based on the roles or permissions assigned to the user. Hence, the RoleAuthorizationFilter will rely on the TokenManager to validate tokens and extract user-related information. Once it has enough information, it'll make access control decisions to determine whether the user has necessary permissions to access a specific resource.

ML: We add information to the images stored in our database using a machine learning algorithm. In our case, we want to identify whether there is a bird in the image. First, we need to rescale our image such that the classification model can handle the image as its input. Now we set the speciesId using the rescaled image and the trained classification model. The speciedId is an identifier that can be mapped to the speciesName. If there is no bird in the image, both the speciesId and speciesName will be null. Otherwise, the classification model will output the most probable bird species from the set of species that we have trained it on. So now we have an image with a width, height, speciesId, and speciesName. This is valuable information for our decision to consider the image for our gallery of bird images.

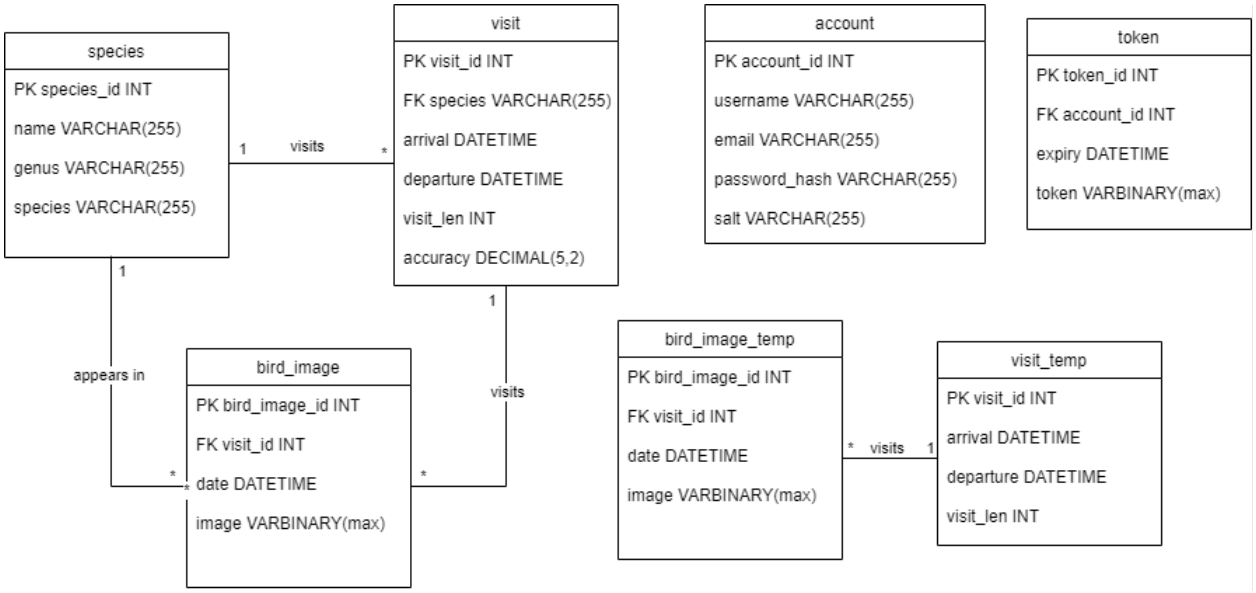
ACTIVITY DIAGRAM



The activity diagram has 5 main swim lanes, including the Raspberry Pi, Knock Sensor, Database, Machine Learning and UI. Firstly, we initialize the GPIO (also known as General Purpose Input/Output) pins from the Raspberry Pi for the knock sensor. When the sensor detects vibrations or knocks in its surrounding environment, the Pi will continuously read the data from sensor. If there exists a change from the baseline state (e.g., 0 to 1, or 1 to 0), Pi will trigger the Pi camera, which is responsible for

taking pictures, and log the timestamps to the database. When the database already stores enough data about taken image, machine learning will retrieve and pre-process the image. Once the processed data is ready, it'll be fed into the ML model. By applying the decision logic with a decent accuracy, the classified results will be stored back into the database. Finally, UI will display the output including bird species and more relevant information about the bird to the users.

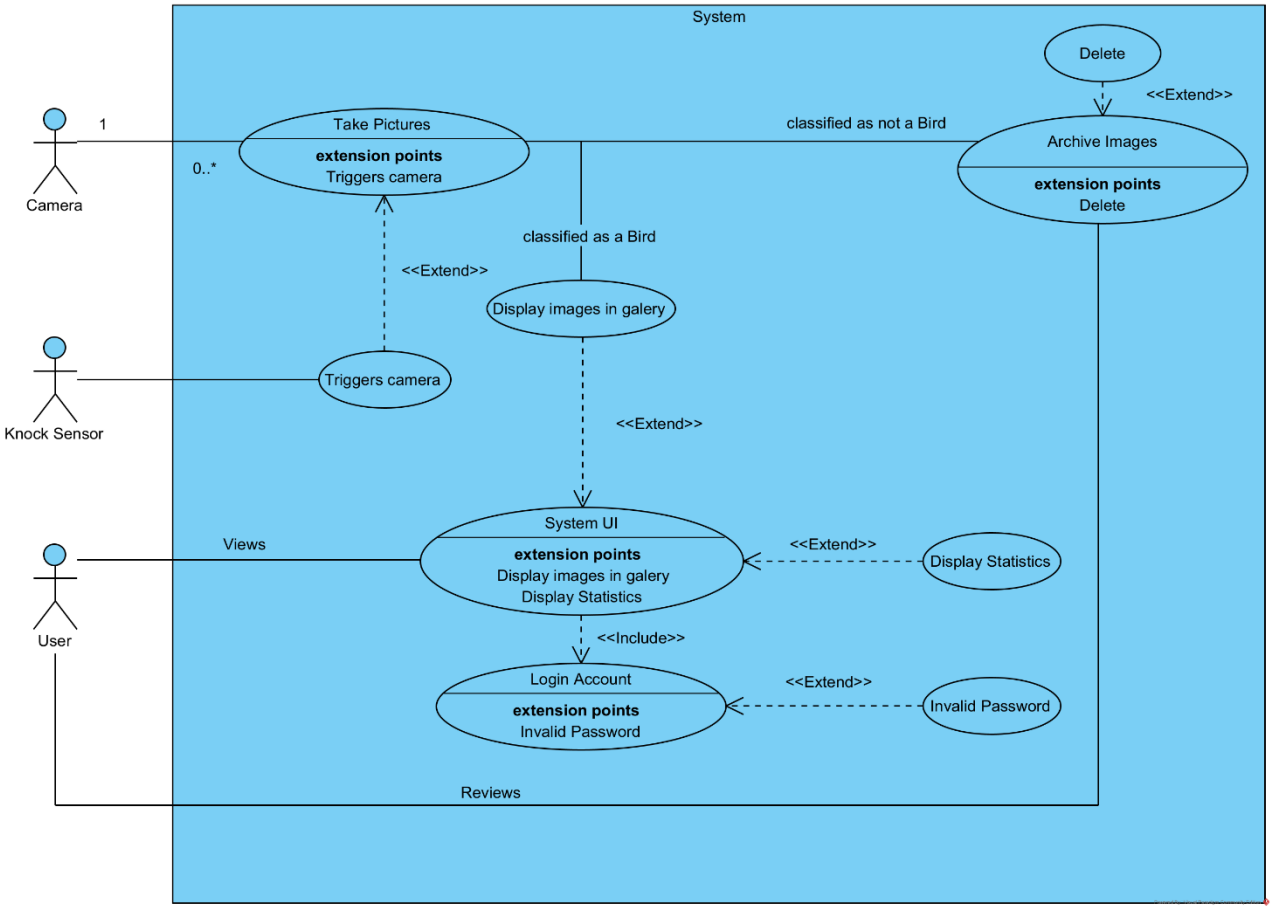
DATABASE ERD DIAGRAM



Some important notes about the design of the ERD:

- There are multiple bird_image and visit tables, with one being temporary and the other being the main table. The purpose of this is to allow the Pi to first send the collected metadata and images to the temporary tables (bird_image_temp and visit_temp). These temporary tables can function as a backlog where the server can run the images through the machine learning model so the species can be identified. Once this is done, the data can be moved to the main tables.
- Each visit can have multiple bird images, but each bird image belongs to a single visit.
- The account table contains a password hash and salt column. (Not storing plaintext passwords)
- The token keeps track of the currently logged in accounts. Expiry of the token will be enforced on the server side.

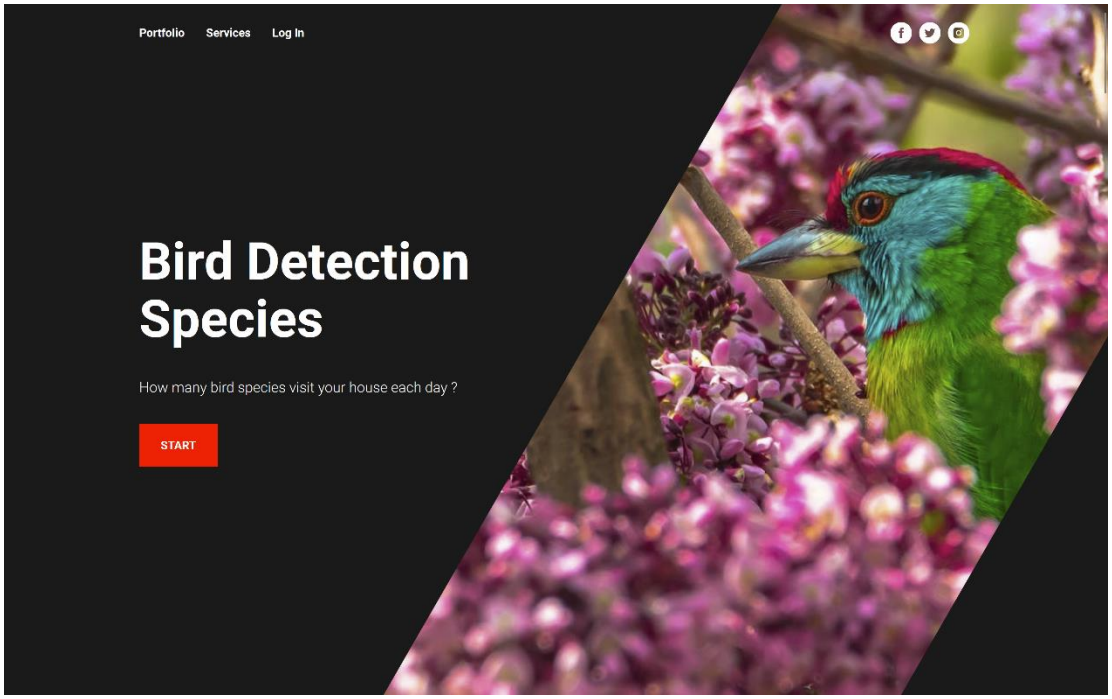
USE CASE DIAGRAM



The use case diagram shows how external users interact with the System. Two of the users are sensors, one of which may or may not be used, however both would activate the camera and take a picture of the bird based on the input they receive from the environment. The media would then be stored and wait to be classified (as can be seen in the activity and class diagrams). After classification it has two choices, one is to go into archive since the image did not contain a bird, and the other would be to receive a label and some other data, after which it will be stored in the gallery. The gallery is visible in the system UI, which also displays statistics, and can only be accessed by logging in with an account. The System UI will also be interacted with by the Users of the website.

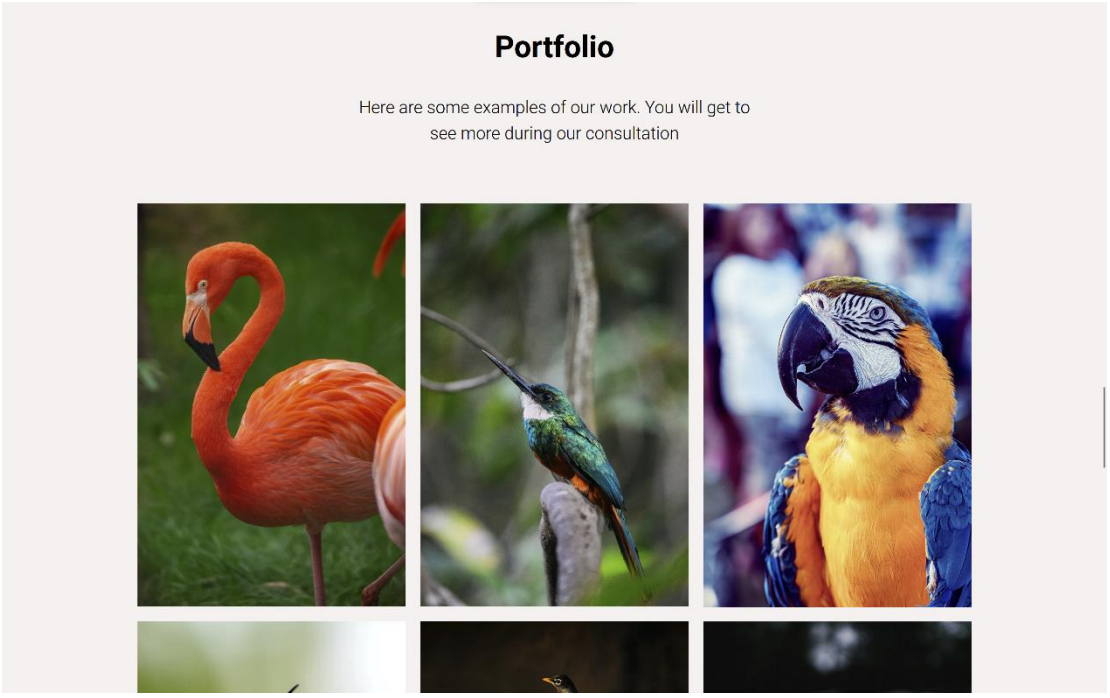
Prepared by: Dipti K. Sarmah

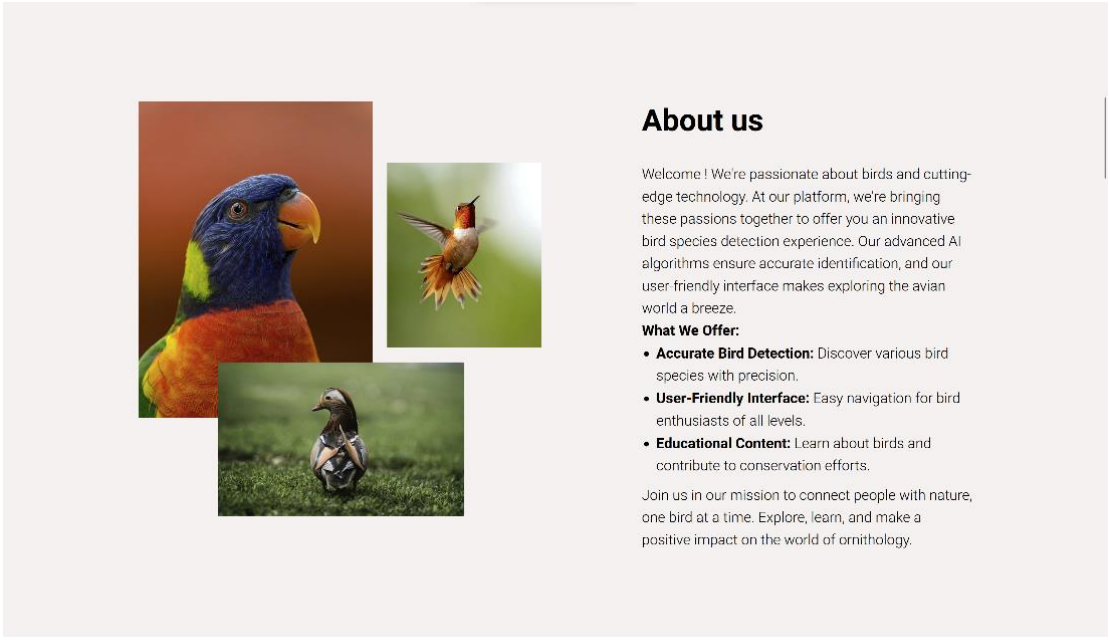
4. Product User Interface



Picture 1

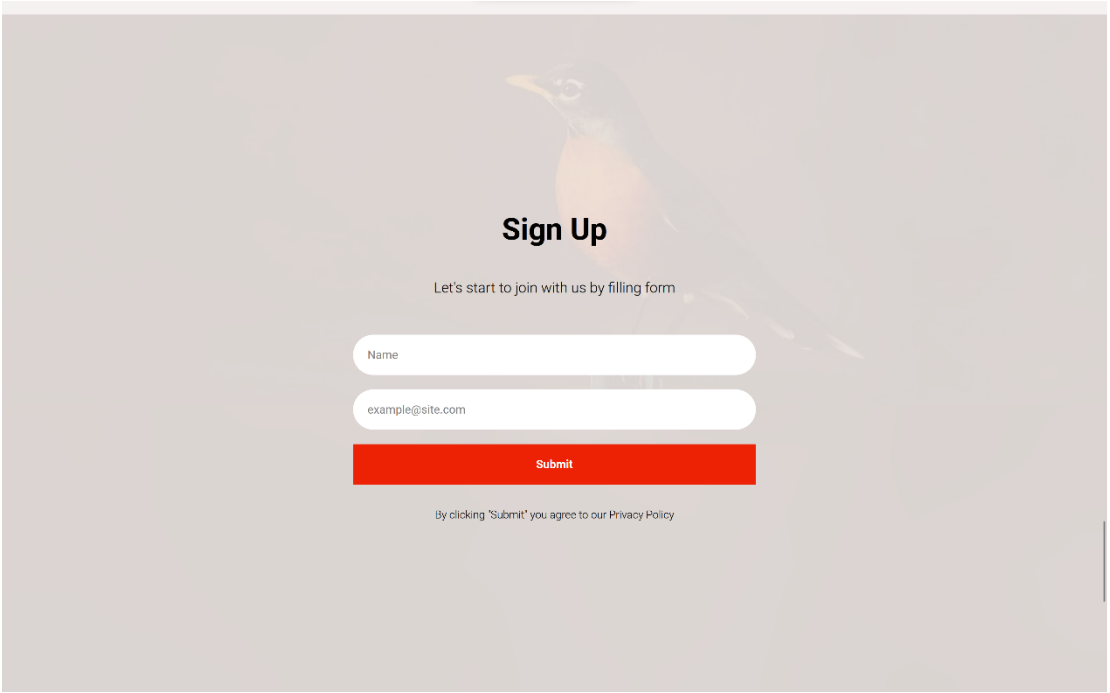
Picture 2



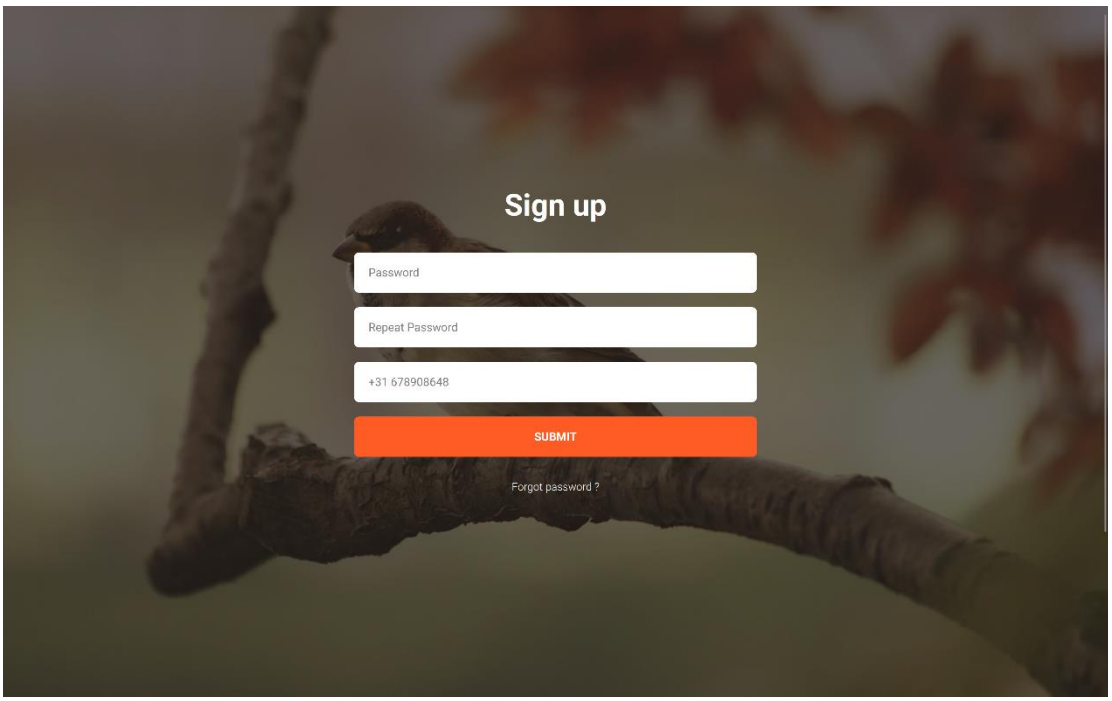


Picture 3

Picture 4

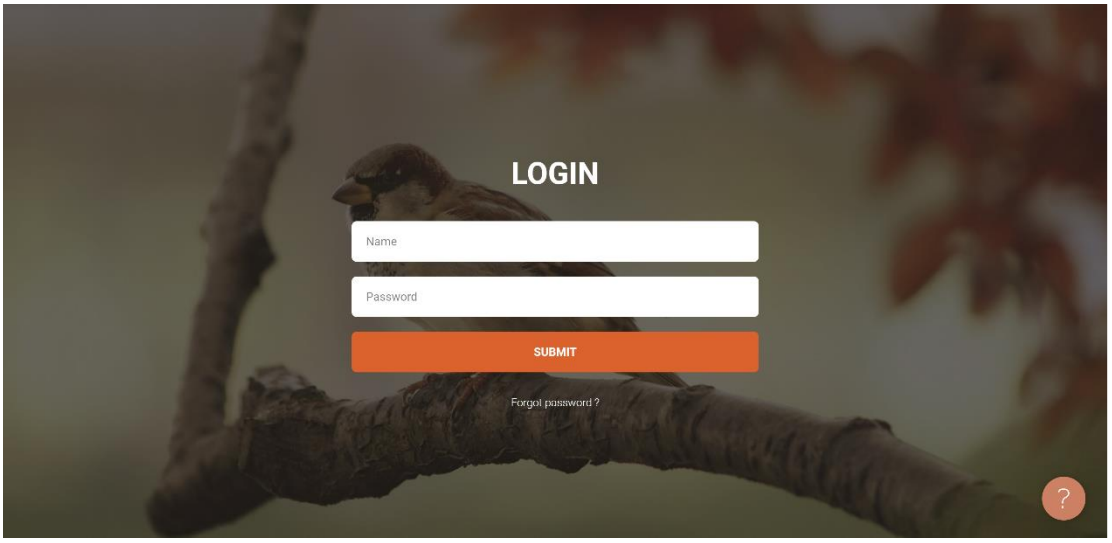


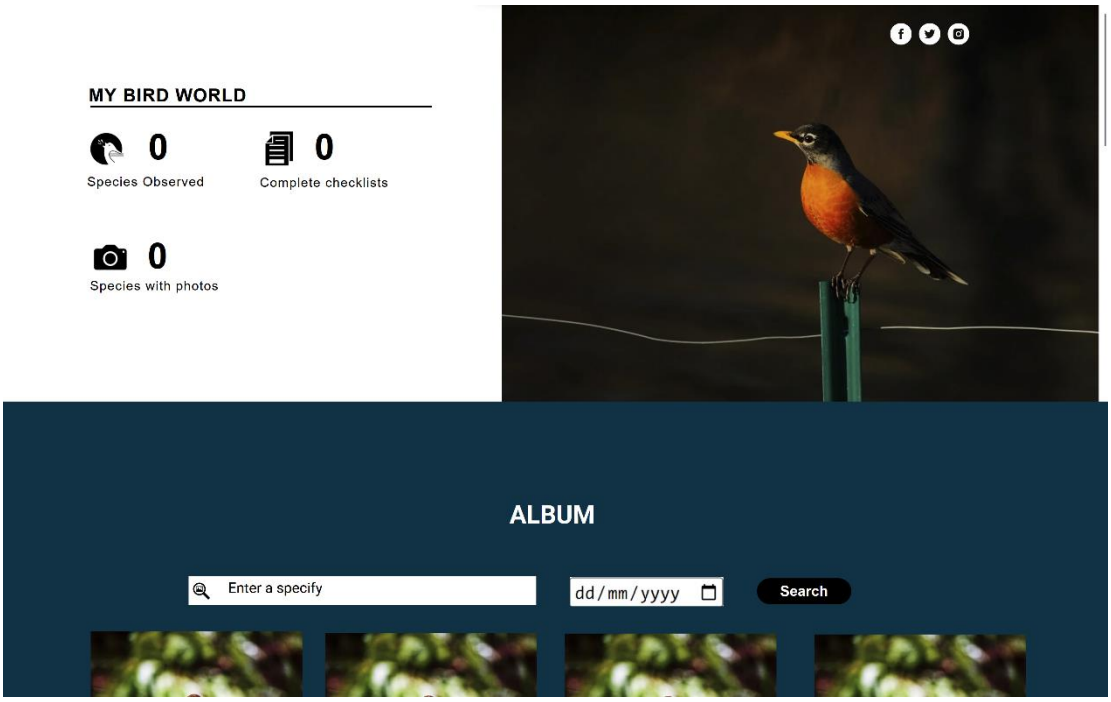
Prepared by: Dipti K. Sarmah



Picture 5

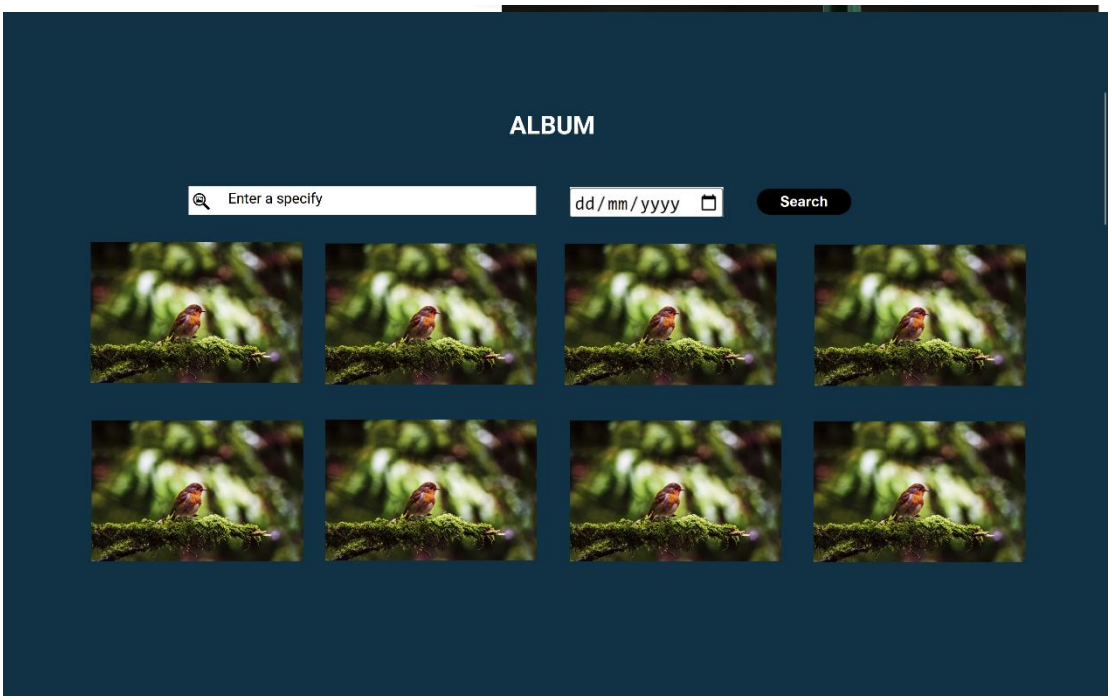
Picture 6



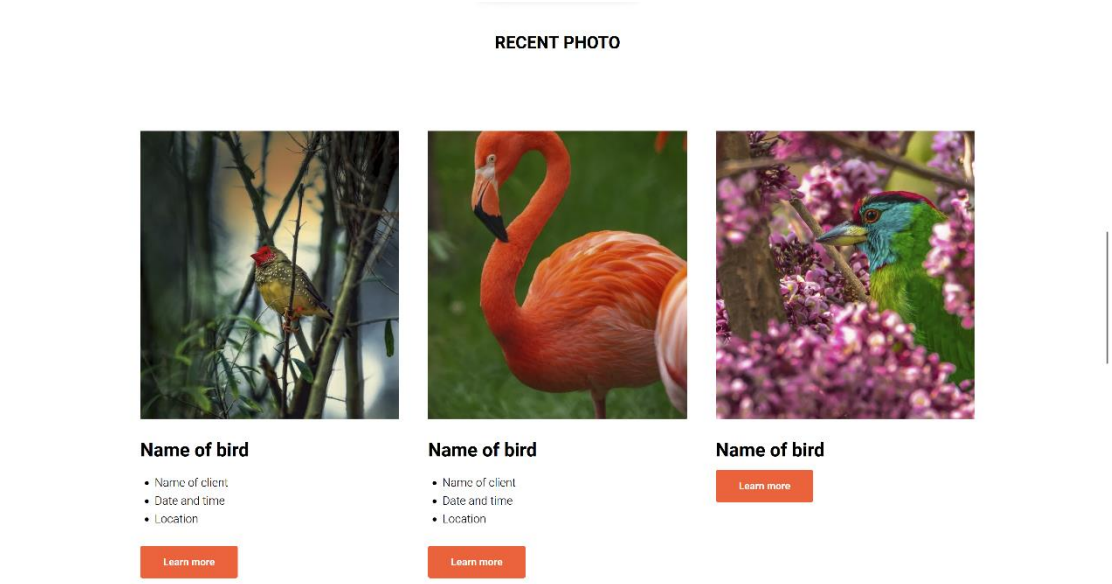


Picture 7

Picture 8

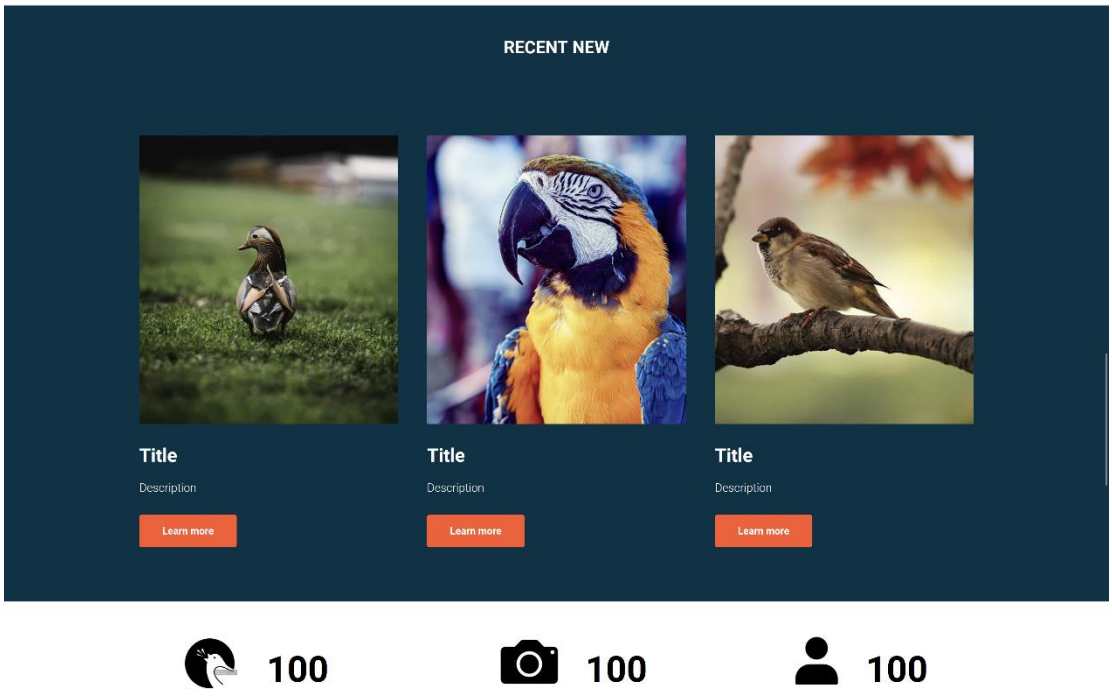


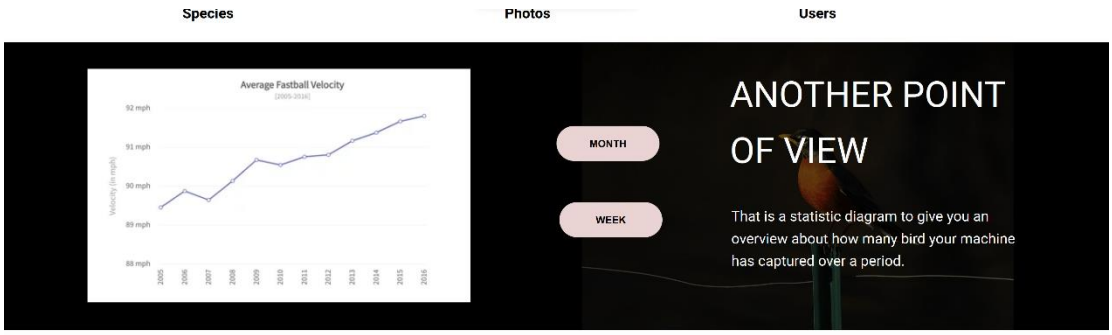
Prepared by: Dipti K. Sarmah



Picture 9

Picture 10





Picture 11

Description: The first picture is the home page where to discover the world of bird watching with our product. Click the "portfolio" button on the navigation to explore our extensive bird species portfolio (picture 2). Then Diving into our 'about us' section to see how we can enhance your bird-watching experience by clicking on "Services" (picture 3).

To start your journey, clicking on "Start" button to create your own account by filling the form with email and your name. After receiving a security email to set up your password by accessing a private link (picture 5). After successfully completing, you can easily log in to become a part of our vibrant community. (picture 6)

Once logged in, you can access an insightful overview that tracks your bird-watching progress. Explore statistics on the number of species observed, completed checklists, and the species documented with photos. Furthermore, you can easily navigate our extensive photo gallery by searching for specific bird species by name or applying date filters to find those memorable avian encounters. Moreover, you'll also have access to the latest pictures of each species and up-to-date news and information about them. Stay informed and connected to the ever-evolving world of bird species as you continue your bird-watching journey with us.

Finally, our platform features a user-friendly diagram that provides statistical insights into your bird-watching activities over a specified period. You can easily apply filters to view data by month or week, allowing you to track your observations with precision and flexibility. In our footer section, you'll find important contact details for our system, ensuring that you can reach out to us for any inquiries or assistance. We're here to support your bird-watching experience every step of the way. Explore, enjoy, and connect with nature through our comprehensive bird-watching system.

5. Prevention/Mitigation Criteria (Security Controls)

Prevention criteria

Security control	Criteria for prevention
Prevent SQL injection attacks by using prepared statements	Whenever any query is sent by the back end to the database, it should be done using prepared statements rather than by concatenating user input. This will ensure that user supplied data is treated as the content of a parameter and never as SQL code, which can be run. This will be done using the JDBC library's prepared statements in our Jersey web application.
Prevent cross-site scripting attacks by validating user input on arrival	Filters will be implemented on the server side to sanitize and validate all user inputs, including URL parameters, query strings, form inputs, and request headers. This filter will be implemented using Jersey features like the ContainerRequestFilter. Upon arrival of the request, the server will reject or sanitize input that contains suspicious characters for example, along with other potentially malicious inputs. The request will either be rejected, or the sanitized request will be passed to the server.
Prevent cross-site scripting attacks by encoding data on output.	Context-aware encoding will be used. When inserting user-generated content into HTML, use HTML encoding. When inserting content into JavaScript, use JavaScript encoding. The same is true for URLs. Jersey provides mechanisms for context-aware encoding such as the ContainerResponseFilter. The OWASP Java Encoder library will be used to encode the output in each of the different contexts.

Mitigation criteria

Security control	Criteria for mitigation
Mitigate unauthorized access to the system by using a login with username and password	The web interface will have a login page, where users input their credentials. If valid, a token is sent to the client by the server. This token is stored in the browser as a cookie and is used to authenticate subsequent requests to the server. The authorization tokens expire after a period of 1 day, after which the user must re-authenticate. The tokens will be JSON web tokens. There will be a password policy where passwords must be longer than 8 characters, contain a special character and a number.
Password hashing to stop attackers from using breached passwords.	When any account is registered, or a password is changed, the password must first be hashed before it is stored in the database. The hashing algorithm used will be SHA-256.
Salting of passwords before hashing to mitigate rainbow table attacks.	A randomly generated salt will be concatenated with the plaintext password before it is hashed and stored in the database. The salt will be 16 characters in length and stored alongside the hashed password in the database, so it can be used when authenticating the user's credentials. This mitigates rainbow table attacks which can occur in the event of a password breach.

6. The cost involved (if any):

Money should not be involved in any of the security controls since we will use software that is freely available for download. Certain security controls are simply practices we must follow while implementing the back end of our web application, meaning their time cost is minimal. An example of this is using prepared statements whenever we perform a query to the database.

The most time-consuming security control to implement will be validating user input and encoding data on output. This is because in certain contexts, different encodings must be used to ensure the output is secure. Additionally, validating user input can be difficult since there any multiple ways to circumvent input validation. Ensuring that validation and the output encoding is secure will be time intensive. Implementing password salting and hashing will also take time but can be done faster due to its simpler implementation details.

7. Conclusion:

From the diagrams, we can see that the database is the centre of our system. This means that it needs to be set up quickly and that the project falls when it does not work. We do, however, believe the database can be set up correctly and in time. The second important part of the project is the Machine learning which is an important output of the project. The machine learning needs to be built as soon as possible to have a functioning model by the end of this project. This will partly be ensured by using transfer learning, it massively decreases the time for training the model. The security part of the project is also of big importance, which is why some important decisions were made, like using salting and hashing.

Some noteworthy design decisions would include the decision to make an archive that can be accessed by the user. This allows for cases in which the user might want to retrieve an image that did not contain a Bird, such as a beautiful image of a sunset or a different animal that triggered the sensors such as a squirrel.

Another interesting choice was adding statistics to the accounts, this means that both hobbyists and Researchers can track various things, such as number of different species in their area, possibly something related to migration patters etc.