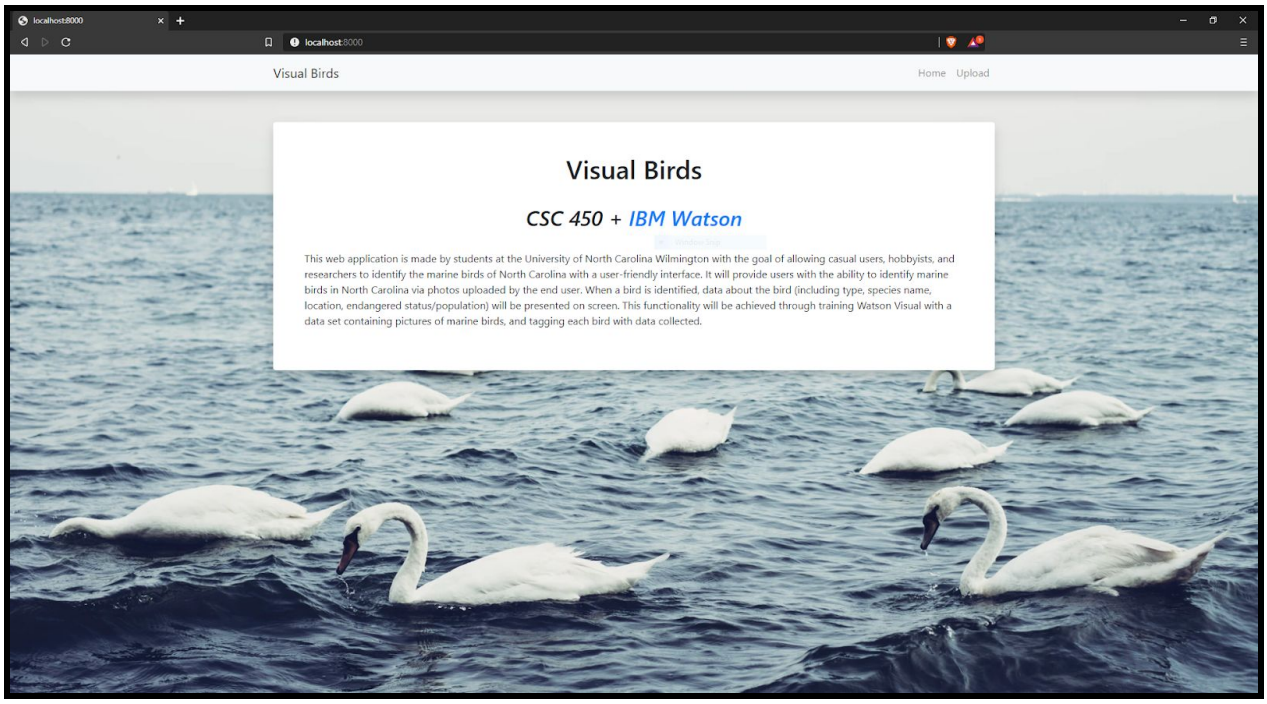
Visual Birds



Visual Birds

Project Documentation

https://github.com/tylerkramlich/visualbirds/

Adam Gray, Dat Tran, Dylan Mennen,

Ryan Bouffard, Tyler Killgore, Tyler Kramlich

CSC 450 - Software Engineering

Dr. Vetter

1. Customer Statement of Requirement

1.1 Problem Statement

Currently, there is an inability for general consumers to easily identify marine birds based on the appearance of the bird in question without previous knowledge of that bird species. This inability to identify the bird species may result in the misidentification of a bird species and/or the improper treatment of endangered species.

1.2 Solution

Utilizing IBM Watson Artificial Intelligence, this project will support users in identifying marine birds. The user will upload pictures to this application and be given relevant information when possible. Correct identification of birds will help inform general consumers of relevant information about the species and future versions of this project may support recording bird population in the area. This project will prioritize developing a prototype, limiting the scope of the project to the identification of a small subset of birds (10 species) local to North Carolina coastal areas for the initial release, allowing for the potential scope of bird species to be widened in future releases.

1.3 Summary

Noting the prevalence of smart-phones with photography and Internet capabilities in the United States market, this project utilizes Django, a high-level Python Web framework, in order for consumers to upload photos and send it to IBM Watson Visual Recognition via an Internet connection. IBM Watson Visual Recognition will utilize it's artificial intelligence tools, that depends on the current pattern analysis of existing images of bird species, in order to support identification of unknown marine bird photos. By choosing to implement this project as a web application, this application will be more widely available and able to be accessed via smart-phone while outdoors where necessary cellular coverage exists.

2. Software Project Plan

2.1 Introduction

This web application will provide users with the ability to identify marine birds in North Carolina via photos uploaded by the end user. When a bird is identified, data about the bird (including type, species name, location, endangered status/population) will be presented on

screen. This functionality will be achieved through training Watson Visual Recognition with a data set containing pictures of marine birds, and tagging each bird with data collected.

This project is a bird identification application, built by a team of UNCW students using IBM Watson, Django, and SQLite database technologies. This documentation is in accordance with Dr. Ronald Vetter's software project guidelines covered in CSC 450 during the Fall 2019 academic semester. Additionally, some IBM terminology may be used in accordance with IBM Watson documentation.

The system objectives are to identify a specific coastal bird based on user picture submission. The user is provided the name and description of the coastal bird as identified by IBM Watson Visual Recognition. If the confidence-level of IBM Watson Visual Recognition is lower than the required score, the user is then notified of a negative result. Some of the major software requirements will be that this website should accept a user's submission of a photo. The photo will be sent to IBM Watson Visual Recognition. Using existing pre-identified bird photos, IBM Watson Visual Recognition deciphers key features of each bird, allowing for user submitted photos to be analyzed for these key features. IBM Watson Visual Recognition, which responds using JSON prepared identifies a specific coastal bird based on user picture submission. Identifying the bird name within the JSON response from IBM Watson Visual Recognition, the user is provided a corresponding description within the application. The main constraints and issues this project may come across is the data set used by IBM Watson Visual Recognition which is finite and static. This prevents the IBM Watson Visual Recognition from improving it's identification capabilities. Additionally, the quality of user submitted photos may influence capability as well.

2.1.1 Project Scope

A user uploads a .jpg or .png file of a prospective bird that the user wants to identify. The photo uploaded sent to IBM Watson Visual Recognition and it's species is identified. The description, picture, and some facts of the identified species is provided if the software positively identifies the bird species. If the species cannot be identified, the user is told that the bird species in the photo uploaded is unable to be identified.

2.1.2 Major Software Functions

Utilizing IBM Watson Visual Recognition, this application will implement machine learning to identify patterns from pre-classified marine bird photos. This will allow user submitted photos to be analyzed and compared to pre-defined marine birds in order to provide the correct species name and relevant species information to the user. After proper identification of bird species, the user will be provided with relevant information about that bird.

2.1.3 Performance Issues/ Behavior Issues

This web application may be affected by the quality of pictures submitted by user, as a proper analysis and comparison may be difficult given improper lighting or focus on the targeted marine bird.

Given the specificity of target animals, marine birds, the slight differences between species ie. American Bittern and Least Bittern may result in longer processing time and/or inaccuracy. Application performance will be limited by internet connectivity of user, as time taken to upload the photo taken by the user will degrade performance.

Lastly, this software project is limited to the efficacy of IBM Watson being able to determine a specific picture's properties.

2.2 Project Estimates

2.2.1 Historical Data Used For Estimation

Some of our software team has had experience with larger projects, but not all. One such project included creating a database from scratch and interfacing it with a newly created website. Due to this experience we expect this project to take nearly as long, if not longer due to the complexity of the material.

Individual tasks such as training a model using IBM Watson are expected to take ~4-6 hours according to several software developer comments on IBM Watson's developer forum. However, given the basic knowledge of IBM tools we expect these tasks to be longer than average.

2.2.2 Estimation Techniques

While most professional project estimates utilize experienced estimators, those who are familiar with the estimation process, our team has no expertise on project estimation. Similarly, our team members have little to no experience on the development side of a full software project, so none of us have experience on that end to estimate either.

We intend to decompose the project into components once all team members have met to discuss the questions regarding the Watson Visual training videos through communication with IBM Watson advisor. While our purpose and the overall nature of the application is defined, the components have not been individually defined due to a lack of project development understanding. We will use a base and contingency approach to estimate the project.

2.2.3 Estimates

The project has not been decomposed into individual tasks, due to unresolved questions that need to be presented to the IBM Watson advisor.

The current estimation will arrive as follows:

- Decompose project into tasks
- Estimate each task individually, add task contingency.
- Add the estimates together
- Add project contingency factor.

Task	Labor Hours
Training IBM Watson models (including wait time)	30
Front-end development	6
Documentation	10
Total	46

Expected labor hours by task

Task	Financial Cost
Training IBM Watson models (including wait time)	0
Training IBM Watson models (not including wait time)	0
Front-end development	0
Documentation	0
Total	0

Expected financial cost in US dollars

2.2.4 Reconciled Estimate

The final cost of this project including effort, time (duration) estimate for the project (at this point in time):

Task	Labor Hours
Training IBM Watson models (including wait time)	40
Training IBM Watson models (not including wait time)	10
Front-end development	10
Documentation	15
Total	75

Labor hours by task

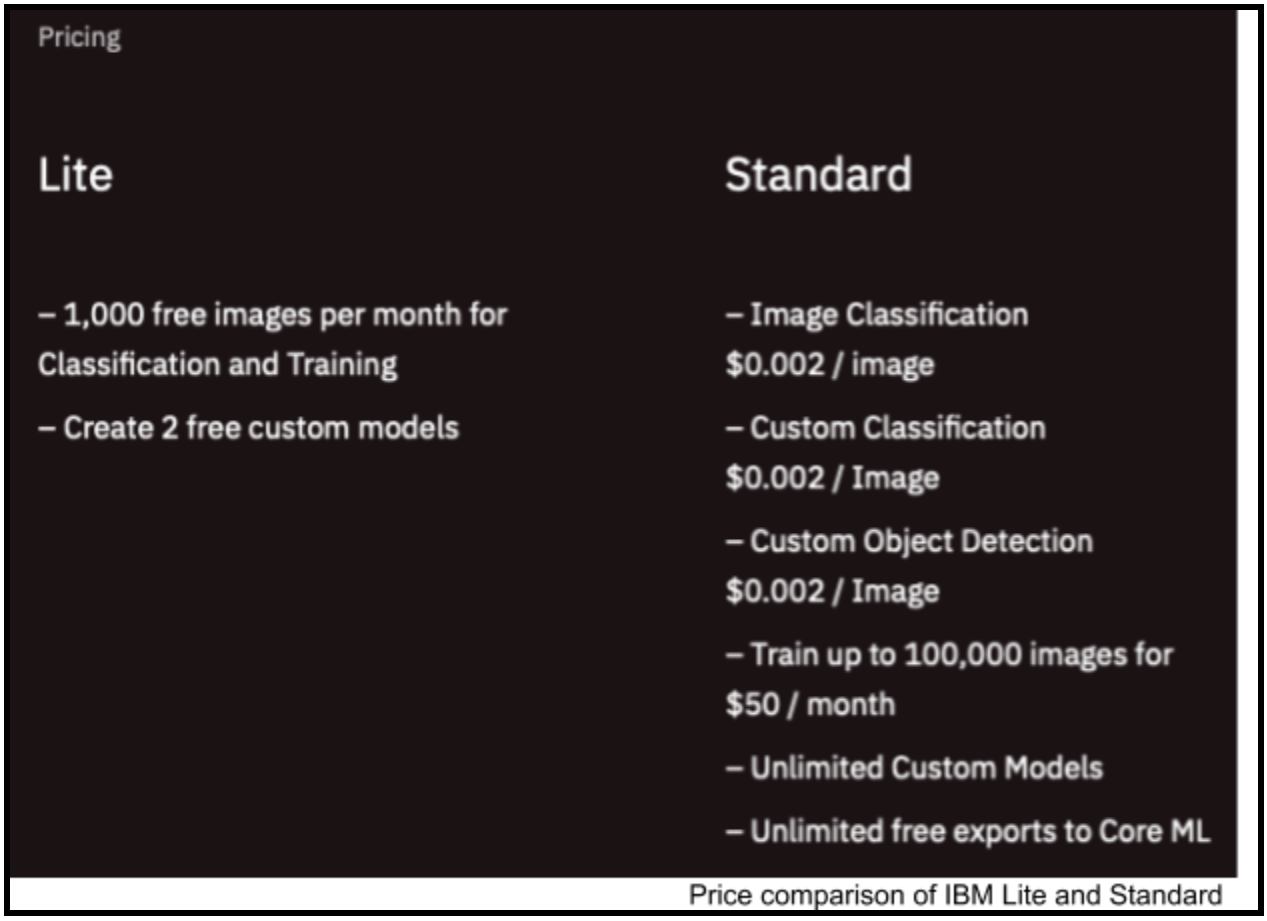
Task	Financial Cost
Training IBM Watson models (including wait time)	0
Training IBM Watson models (not including wait time)	0
Front-end development	0
Documentation	0
Total	0

Financial cost in US dollars

2.2.5 Future Estimates

The development of this project as outlined was conducted with no financial cost. However, given the lack of technical support provided to IBM Lite Plan users and limited API calls, as this project scales then the necessity of upgrading to at least a ‘Standard’ paid plan would be necessary. Given the limited scope of identifying ten bird species and the ability to train IBM Watson over a series of months allowed for this project to not cost any money. For this product to be viable beyond ten bird species however, it is expected to require at least the ‘Standard IBM Plan’.

Based on experience from the initial prototype of this project it is expected that adding at least four bird species to the identification capabilities of this application would require at least one individual training. This would require at least $50.00 per four bird species. Additionally, this would likely be the bulk of the cost of future project of larger scope. The costs of the IBM standard plan is included below.



Price comparison of IBM Lite and Standard

A Detailed Overview of Standard IBM Plan Costs:

Access to everything from the Lite Plan including:

Train up to 100,000 images per month (only charged if training occurs during the month)

Unlimited image classifications per month (charged per image)
Unlimited custom models
Unlimited free exports to Core ML
$0.002 USD /GeneralTagging
$0.002 USD /CustomTagging
$0.002 USD /FoodTagging
$0.002 USD /ExplicitTagging
$0.002 USD /ObjectDetection

$50.00 USD /Training

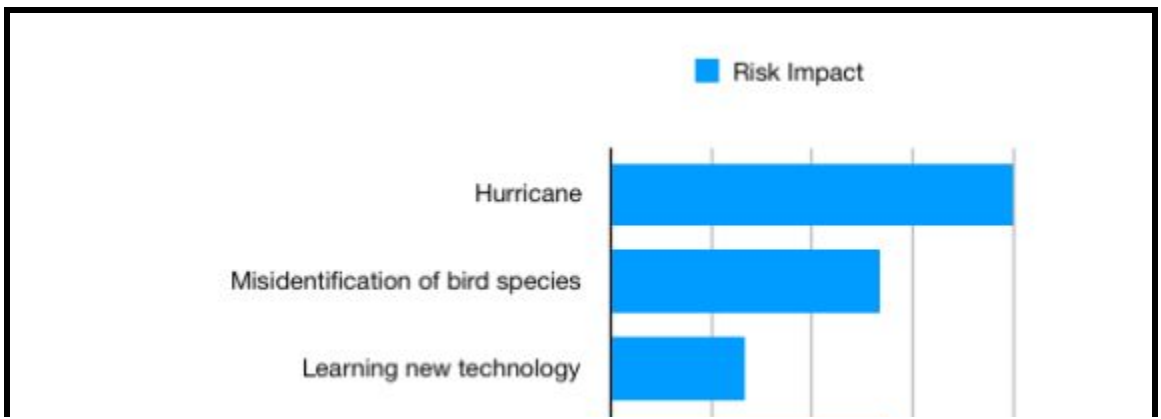
2.3 Project Risks

2.3.1 Risk Identification

- We've identified several risks posing a threat to this project.
- An inclement weather situation that impacts the school schedule, and therefore the project schedule.
- Misidentification of species, which will happen as our application will not have 100% accuracy.
- Students learning new technology.
- Incompatibility of the final product in our GUI.
- Scheduling issues of team members.
- Scheduling errors due to erroneous time estimates of task completion.
- Project scope too large.

2.3.2 Risk Estimation

- Estimating the risks we know about may require iteration, and new risks may be added further along in development. The rating system used in the chart below is as follows:
- Rating system from 1 to 5
- 1 has the least impact on the project, minimal risk
- 5 has the most impact on the project, maximum risk
- Hurricane is rated at 3, low chance but very high impact if it does occur.
- As for rating the misidentification of species - with enough risk control even if the accuracy is below where we'd like it to be, it will have somewhat low impact and as such is rated at 2.
- The new technology will take time to learn, however the risk seems minimal and therefore rated at 1.
- As this is a new technology for us, incompatibility of final product in GUI is of noticeable concern, rated at 2.
- Scheduling issues of team members may occur, but there are mechanisms for handling this, and so rated at 1.
- Scheduling errors due to erroneous time estimates of task completion, will have significant impact and therefore rated at 3.
- Risk of large scope, may not be able to implement everything planned, may have to cut features or project may lose focus, rated at 2.



2.3.3 Risk Resolution

Regarding the risks identified earlier, with the deadline in December, there is not much room for resolutions beyond accepting and controlling these risks. For the chance of a hurricane, the main way to resolve this risk is to accept the chance of it happening, and move forward with the project.

As for the chance for misidentification of species with this project, all measures must be taken to control the risk and minimize its impact. Next, as we are all learning this new technology there is a risk for using it wrong and unintended behavior occurring, as such to minimize the risk we have to understand how to use it.

Regarding incompatibility of the final product in our GUI, we need to thoroughly test the product. For the scheduling issues of team members, the use of technology to ease the impact it can cause, such as using slack to post important links like meeting minutes, or using skype if remote calls are an option. As for scheduling errors that crop up due to erroneous time estimates of task completion, as the project moves along we need to gauge, control, and update our scheduling, and make up for any time lost.

Finally, the last of our projected risks - risk of too large a scope. In order to lessen the impact this risk can cause, there needs to be a focus on what is doable, and we must remind ourselves that not every feature gets implemented.

Potential Hazard	Who is at risk?	Risk Rating	Action to be taken
Inclimate Weather	Students/Project	3	Work ahead when possible.
Misidentification of species/Insufficient data	Wildlife/End user	2	Make sure data is accurate with enough samples, otherwise provide bias towards endangered species.
New Technology	Project	1	Understand it as well as possible before moving forward.
Incompatibility of final product in GUI	Project	2	Find any flaws that could cause this, control the outcome.
Scheduling issues of team members	Project	1	Use technology to ease the impact, use slack to post important links like minutes, use

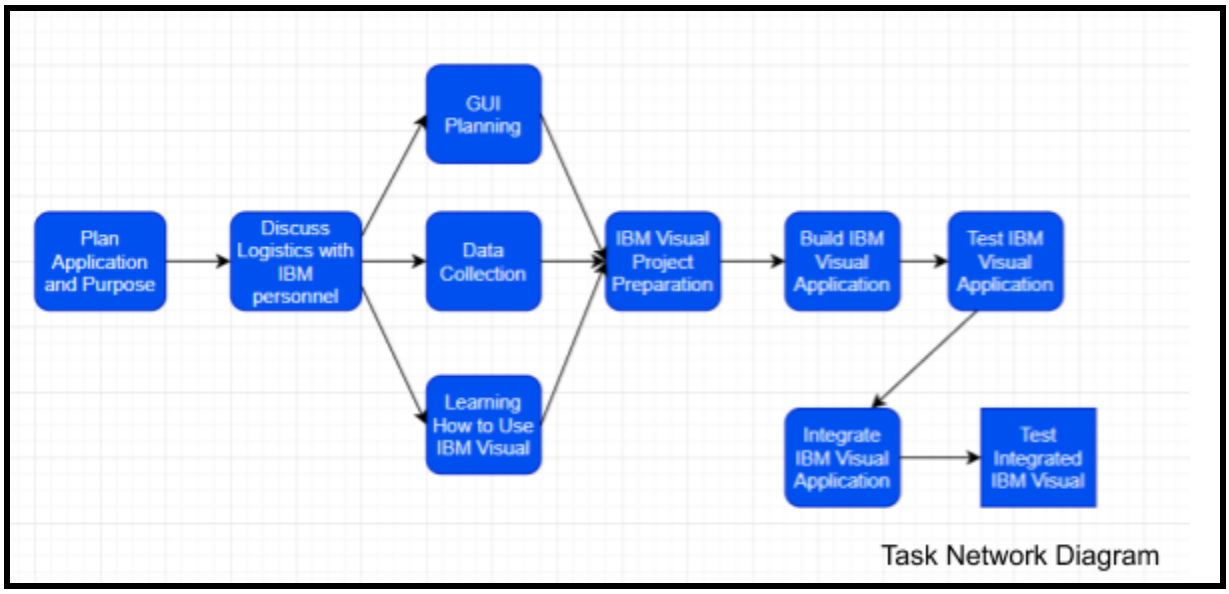
			skype if remote calls are available.
Scheduling errors due to erroneous time estimates of task completion	Project	3	Continue to gauge scheduling as the project goes along, make up for lost time.
Risk of large scope	Project	2	Focus on what is doable, not every feature makes it to implementation.

Risk assessment and Identification (Ratings: 1,2,3,4,5 - 5 being the most dangerous risk)

2.4 Schedule

Once all of the members have reviewed the material and we have had a meeting to discuss how the project will be divided into several components. Each member will receive an equitable share of the overall work-load and work will largely be divided by task dependency, with strict deadlines for tasks and members, meaning that some members may have extended periods without deliverables due.

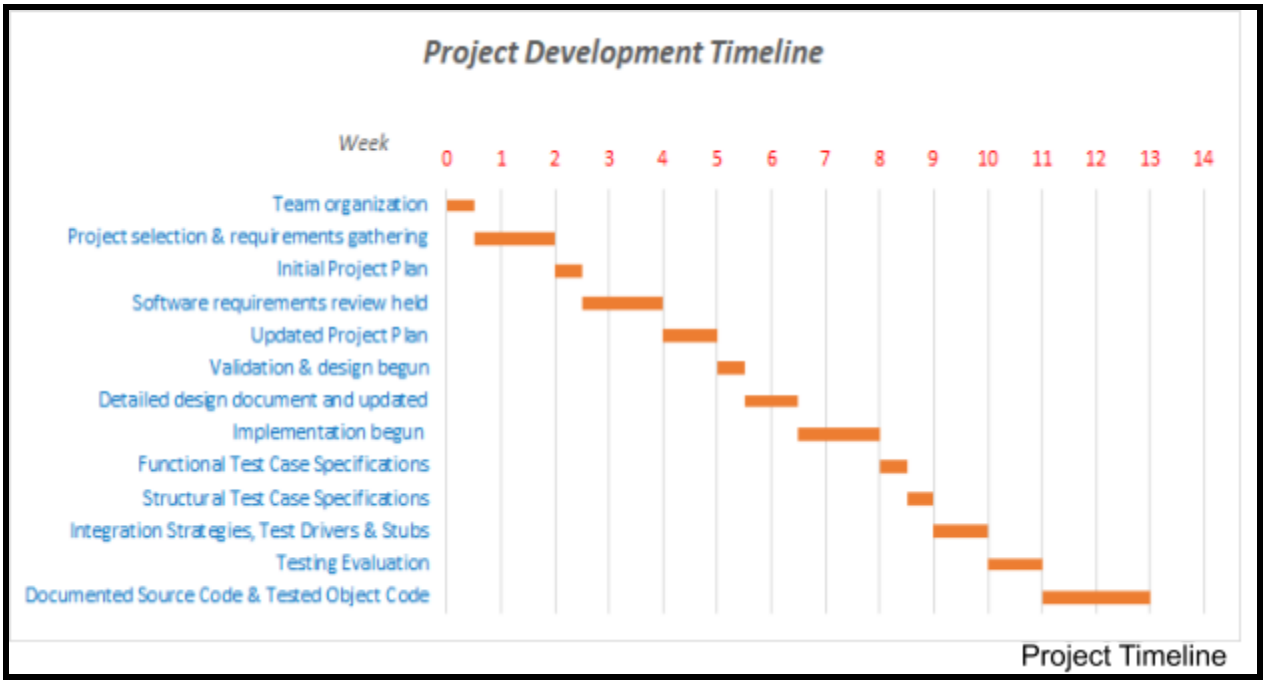
2.4.1 Task network (CPM)



Detailed Description:

Top down view on our project, describing how each step connects to the next, and which steps in order we will need to take to finalize the project. Currently our task network looks as above, with planning our application and purpose to start us off. Next, we will discuss the logistics of this project with IBM personnel. After we check-in with those at IBM, we move on to GUI planning, collecting our data, and beginning to read documentation and learn how to use the IBM Visual tool. Once we complete all of these, we will move on to preparing our IBM Visual project. After initial preparation is completed, we will begin to build and test our IBM Visual application. Once it has been fully tested, the plan is to integrate our application, and finally test it in completeness.

2.4.2 Timeline chart (Gantt chart)



Detailed Description:

Timeline describing the estimated time that each section of our project will take to complete, with the project concluding by the end of the Fall semester. Week zero contains team organization, which we meet and share contact. After that, Project selection and requirements gathering takes place. Starting week two our initial project plan will take form. Next, we will review the software requirements for this project. After reviewing our requirements, we update our project plan accordingly. Week five begins with the validation and design stages starting up, and also will also produce a detailed design document for the project. Next, implementation begins, and extends somewhat far into our planned timeline. When implementation ends, we begin creating test cases,

starting with functional and moving onto structural cases. As we start into the twilight stages of our project, integration strategies are discussed, and test drivers are produced. All of week ten is dedicated to testing and evaluation of our project and finally, the timeline concludes with documenting our source code in more detail, and final testing of our product.

2.5 Project Resources

2.5.1 People

- This project will utilize help from a scientist in the UNCW Marine Biology program, in order to assist with the bird data. If possible, they will be able to point us in the right direction on where we will be able to find the data needed, or already have a set compiled.
- This project will also need assistance from an IBM advisor, to assist us with the technical side of Watson Vision and any problem we may come across.

2.5.2 Team Members

- Adam Gray
- Dat Tran
- Dylan Mennen
- Ryan Bouffard
- Tyler Killgore
- Tyler Kramlich

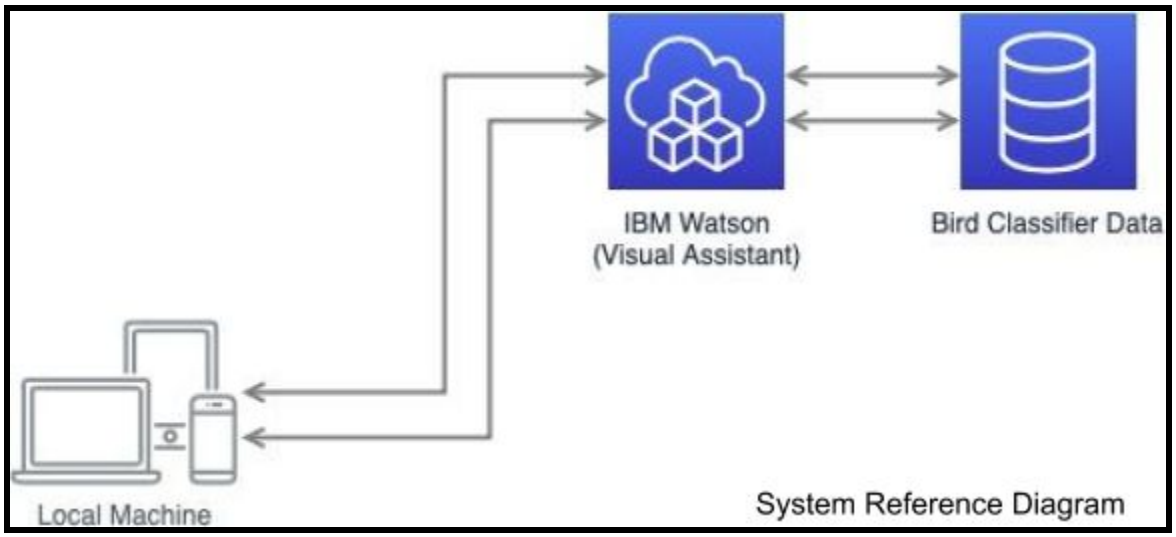
2.5.3 Hardware

- User phone or computer to send photos looking to be identified.
- Watson servers to do the data crunching and return an identified photo.
- Each team members computer for contribution to the project.

2.5.4 Software

- IBM Watson Visual for machine learned photo identification.
- We will need to gather data from a database already set up by a marine biologist, or gather data on our own from a variety of websites.
- Slack for team collaboration.
- Discord for virtual meetings.
- Google docs for documentation.
- Github for version control.
- Draw.io to accommodate chart creation.

3. Software Requirements Specification



Detailed Description:

System reference diagram. This is the connection and transferability from local machine to IBM Watson as a Visual Assistant, and then classify the data of birds to be the standard results to compare with uploading images.

3.1 System Requirements

- Equipment: Desktop, Laptop, and mobile-device.
- Software: A web application, IBM Watson Visual application, and operating system such as versions of Microsoft Windows, Apple's macOS, Chrome OS, BlackBerry Tablet OS etc.
- Tools: searching and uploading for images, images of marine birds, text of marine birds' lifestyle.

3.2 Software Project Constraints

Cost:

While we have no foreseen monetary expenses in this project, our budget will be $50 just in case something comes to our attention.

Scope:

We have to collect information pertaining to a number of marine birds. We have limited our data collection to North Carolina (NC) birds because the original proposal for marine animals would result in a data set too large for us to efficiently manage. Our project will have a

user application that will access our trained IBM Watson tool, and then pull information from our dataset containing pertinent information. The scope is limited in this sense because communication with IBM advisors is complicated and time sensitive.

Risk:

Due to delays caused by inclimate weather and an ongoing experience gap with the IBM visual software, the project may not be complete or perfect by the deadline.

Resource:

For each NC marine bird, we have to collect many images in order for the IBM Watson Visual application to analyze and recognize it correctly. In general, if we want to expand the project scope to more birds or marine animals, data collection may prove to be a challenge.

Time:

This will likely pose the largest challenge for our software. While we have begun studying how to work with the IBM Visual tool, our complete understanding of the tool may pose a threat to our limited time available. Additionally, collecting individuals, materials, and data together to process our software development is time consuming. A thirteen week semester may be long enough for building a software, but team members have to work on the other classes' projects and also learn how to create our visual project for the first time.

3.3 Information Description

Data Object

Bird	
P	Name
	Description
	Location
	Color
	Biological_Class

Database table of bird

Detailed Description:

Our program only uses one entity, the bird entity. The name primary key will be used to identify the JSON file given by Watson Visual, to connect it to the data in a small python dictionary.

3.4 Functional Description

3.4.1 Functional Partitioning

- When web app opens, prompt for user's permission to use camera
- After permission is granted, allows user to take a picture

- When the picture is taken, has a confirmation window/button
- When picture is confirmed, checks it using IBM Visual Recognition
- IBM Visual Recognition returns species name along with confidence value
- If the confidence is above a certain threshold, then the webapp returns the species name and information about it. Otherwise gives an error message saying it requires a different picture as it cannot tell the species with enough confidence.
- Will request the user to take another picture.

3.4.2 Processing Narrative

If a user wants to know what bird they're looking at, it can open our webapp. After the user opens the webapp, it'll need permission to use their camera, and then when the user take a picture, it'll check with the IBM Visual Recognition tool. If it has enough confidence it'll send back the species and other information, otherwise it will have to send back an error message, asking the user to take another picture.

Restrictions / Limitations

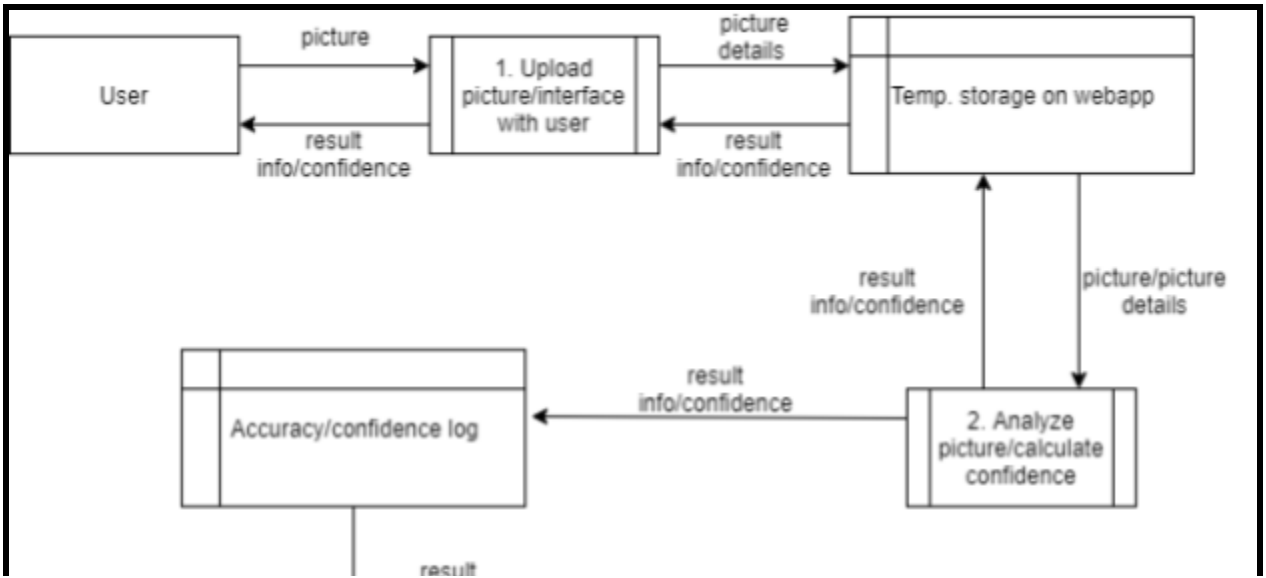
- Don't allow users to affect our dataset, only researchers could have that possibility or peer reviewed pictures.

Performance / Requirements

- Webapp must be able to run on a phone, as that is the expected use case from our users
- Minimize wait-time
- Allow users to check from both camera and gallery applications.

Design Constraints

- Less screen real estate on phones, possibly less information can fit on screen if it correctly identifies species.
- Make controls rare but easy to use.
- Requires an internet connection



Detailed description:

Supporting diagrams. In details of the system reference diagram, a user is going to upload a picture, it stores on the webapp and is analyzed or calculated confidence. Then, the result is sent into two different ways; firstly, it is sent the confidence log of the developer to update the accuracy; secondly, it is sent to the webapp's storage to let the user know the result info.

Control Description

Provide more data or data sets to train this visual recognition tool, otherwise allow researchers to add data that is relevant.

Control Specification

- Do not allow users to modify data, only use the provided application
- Possibly allow researchers to modify the data, to allow for greater confidence in visual recognition

Design Constraints

- Relies on an abundance of accurate and documented data.
- Relies on oversight from development team.

3.5 Behavioral Description

3.5.1 System States

Start / Display App Information

On access of the Web App, an information screen explains the application in detail as well as includes instructions on how to use the system.

Upload Photo Prompt

User is prompted to upload a photo of a bird for system determination.

IBM Processing

IBM Watson processes the image and returns the result and confidence rating.

Display Results / Confidence

This state gives the user information such as which bird the system determines, along with the confidence rating of that determination.

3.5.2 Events and Actions

Go to Upload Photo Prompt

Allow user to navigate through controls to the upload photo prompt.

Go to Information Screen

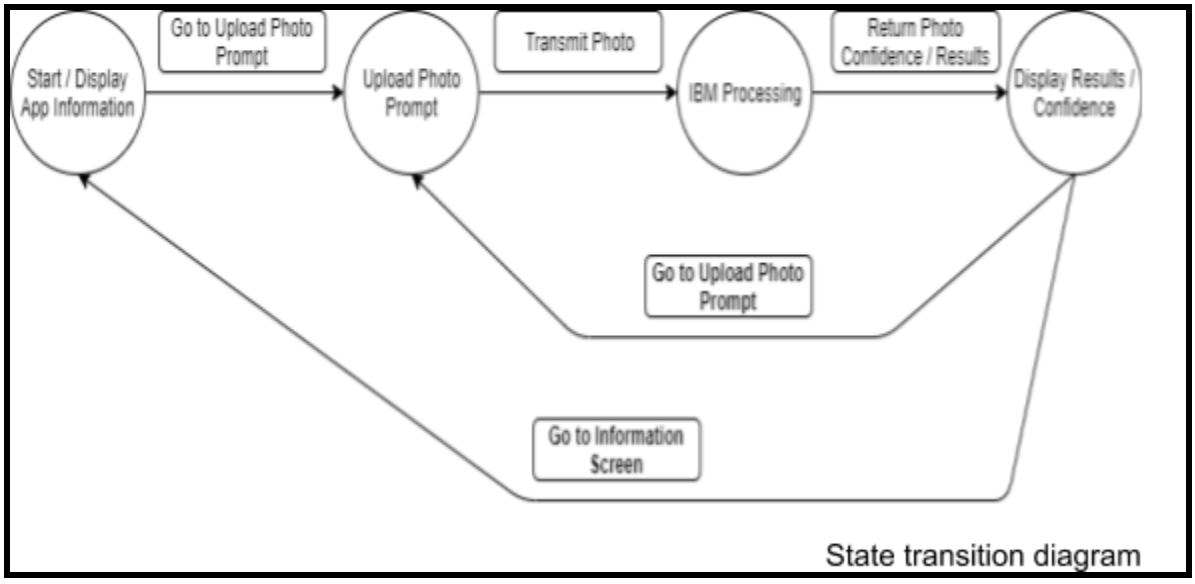
Allow user to navigate through controls to the information screen.

Transmit Photo

After uploading a photo using the prompt, the information is then fed into IBM Watson for processing.

Return Photo Confidence / Results

After the IBM Processing state, IBM Watson will transmit it's determination of the photo along with its confidence score to the web app. This event will then trigger the Display Results / Confidence state.



State transition diagram

Detailed Description:

Project State Transition Diagram. Ovals indicate the states within the system. Rectangles indicate events and actions that trigger transitions between these states.

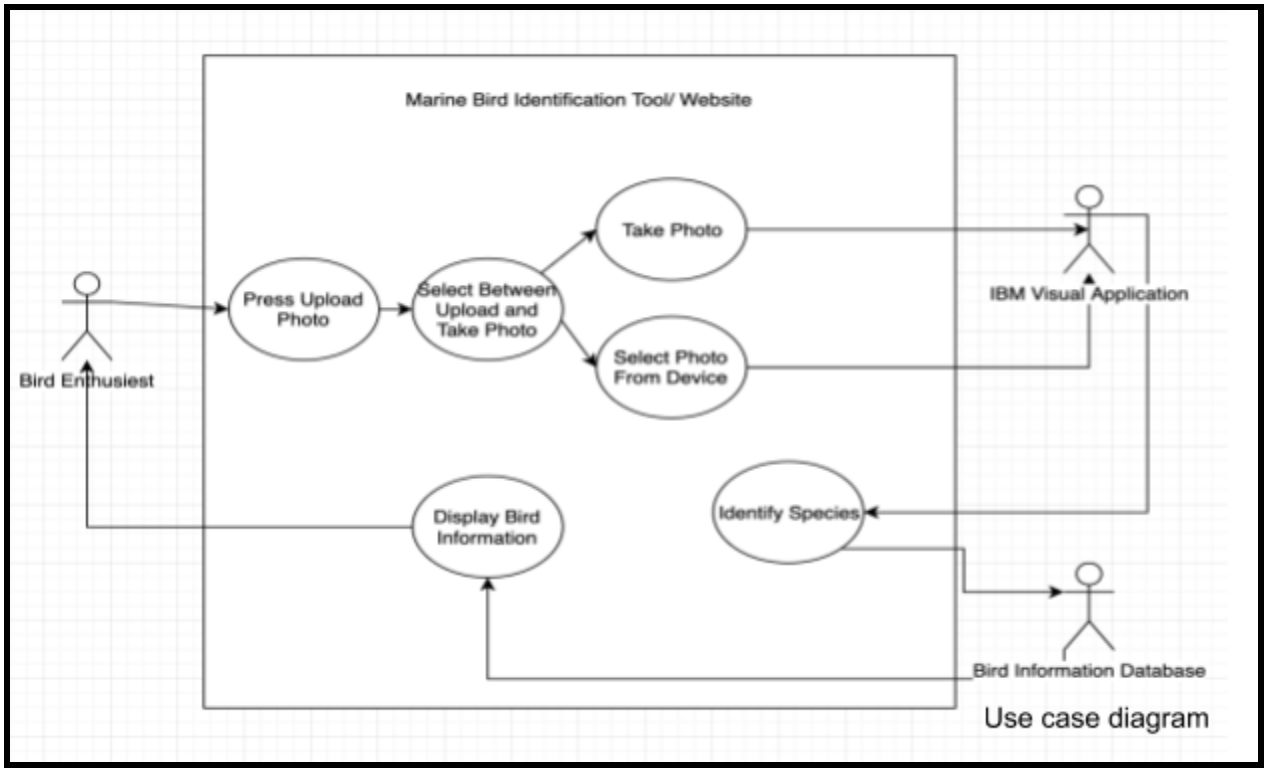
3.6 Use Cases

3.6.1 Stakeholders

We have a very limited list of stakeholders due to the nature of the project. Our stakeholders include marine scientists, bird enthusiasts, team members, Dr. Vetter and IBM. The marine scientists and bird enthusiasts are our expected users, and their stake is limited to the ability to utilize our functional application. The team members stake is learning about the software development process along with receiving a grade based on the documentation and functioning application along with learning IBM Watson Visual and the time it will take to complete the project. Dr. Vetter, the professor of the course must grade the documentation, application and provide support and assistance when required throughout the project. IBM has supplied an advisor as a resource to assist throughout the project. While IBM has no financial stake or direct benefit from providing an advisor, their education program encourages students to use and learn their software for an increased involvement and understanding in the artificial intelligence field, which will result in a more knowledgeable workforce and community of software engineers.

3.6.2 Actors

Actors fall under two categories for this project: end-users and those involved in the program. There are two potential users of our application. The first is a user that is involved in the scientific or research community who would use the application to more efficiently identify marine birds and pull information on the bird for purposes related to their work/study. The second group of users are simply those who may enjoy learning about marine birds and wish to be able to identify birds or gain some basic understanding of them. The other “actors” include the IBM Visual tool which our application will access and a database of information pertaining to the birds that our application will identify.



Detailed Description:

The use case diagram above shows the use case of a "Bird Enthusiast". The user will access the website/ application and select "Upload Photo", followed by a prompt to select between "Select Photo From Device" or "Take Photo". If the user selects the "Select Photo From Device" option the photograph will be passed to the IBM Visual Application. If the user selects the "Take Photo" option the photograph will be passed to the IBM Visual Application. In either case the IBM Visual Application will, by its own means, analyze the photo against learned attributes and conclude the species of bird in the uploaded photo. This information will be passed back to the application. After the application receives the species information, a call to the Bird Information Database will occur that returns relevant information to be displayed on the users device.

4. Software Design Specification

4.1 Data Design

4.1.1 Bird Information Database

Our data design includes a database table that will store each bird species' basic information. As birds are identified by IBM Watson, this table information is then queried.

Database table of bird (species)

4.1.2 File Structures

JPEG / PNG

The user takes a photo of a marine bird and submits to our application a JPEG or PNG photo. This photo is then discarded after relevant processing and information is displayed to the user.

JavaScript Object Notation Results File

IBM Watson returns a JSON file after processing the aforementioned JPEG / PNG submitted by the user. This file contains the confidence rating of each bird the picture may contain. This file is then discarded after relevant information is displayed to the user.

Django Framework

Using Django models, the records of birds identified by the JSON Results File can be extracted from our SQLite Table (see above). This is then gathered into variables and sent back to the user through Django's powerful templating engine.

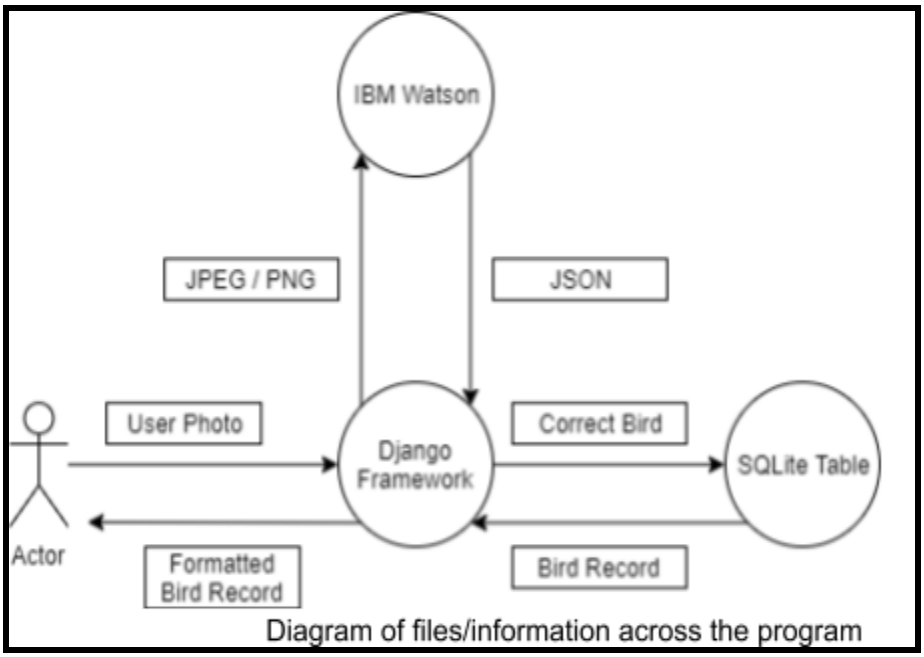
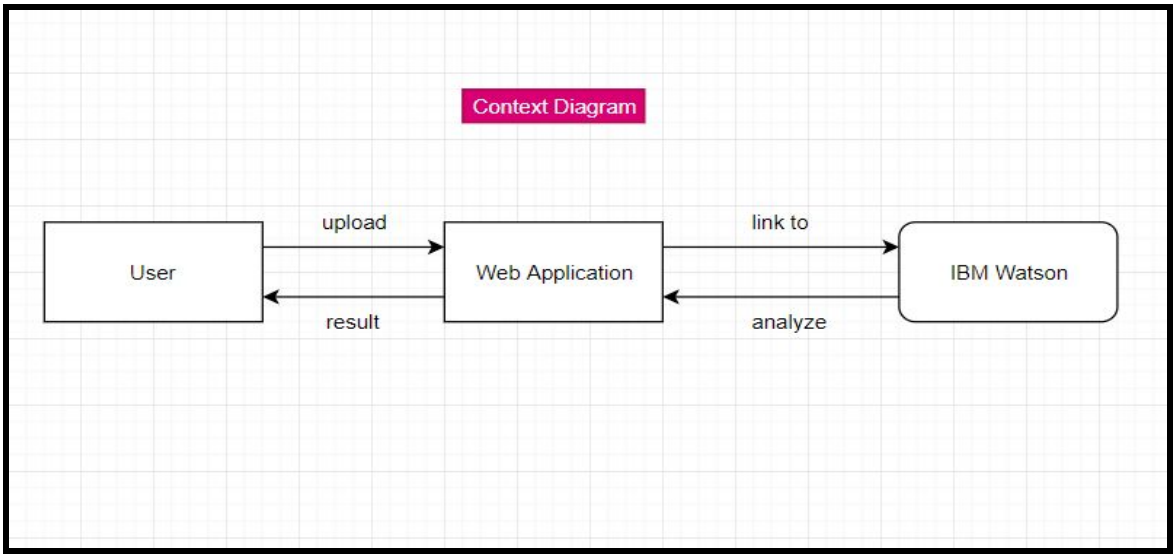


Diagram of files/information across the program

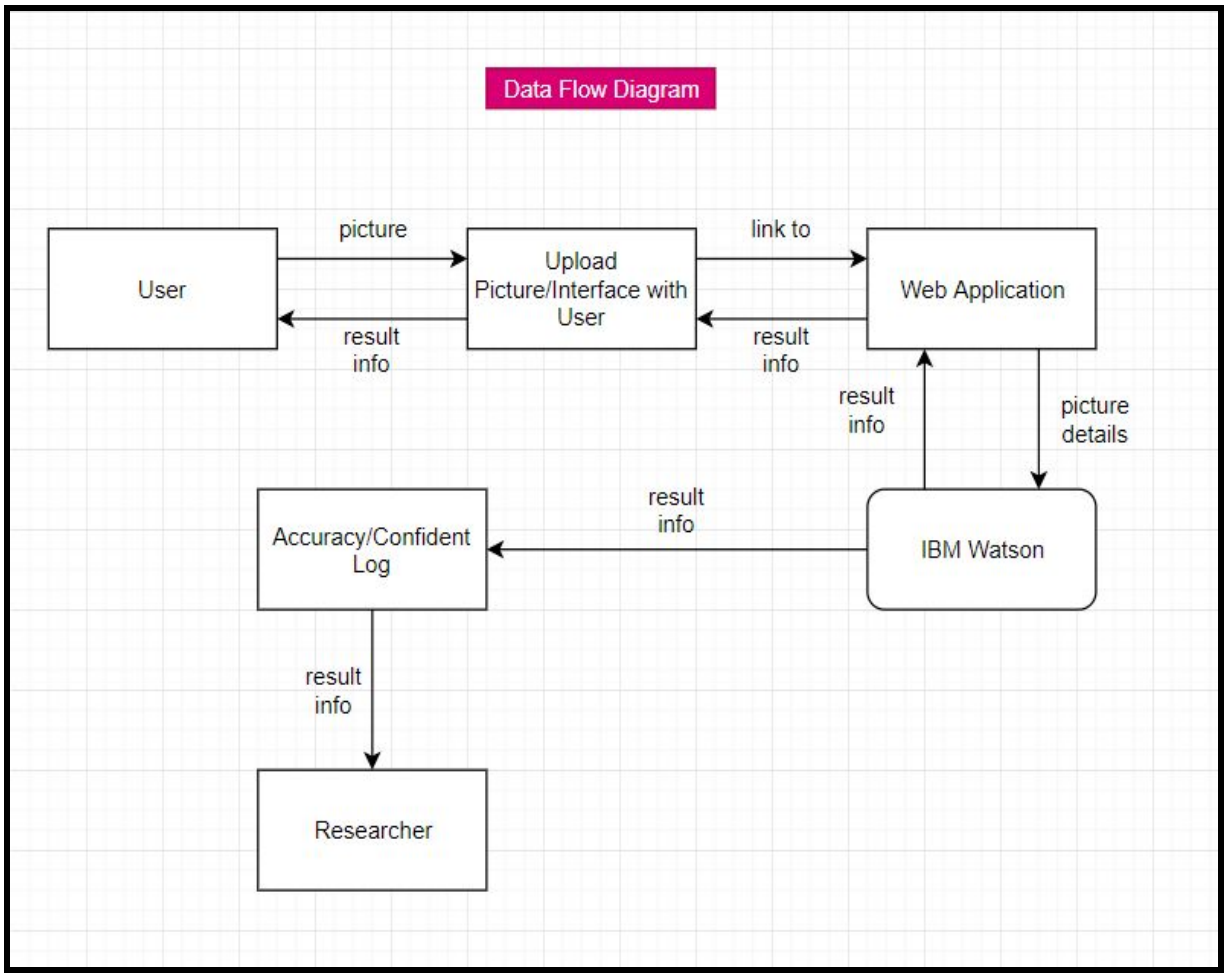
4.2 Architectural Design

- Review of Data and Control Flow
- Review of control flow: all data structures within this Marine Bird Identification web application will be provided and managed by the IBM Watson Visual Recognition.
- Review of data flow:



Detailed Description:

A context diagram, sometimes called a level 0 data-flow diagram, is a short cut of the data flow. A user uploads an image to the webapp which is linked to the IBM Watson to analyze the uploaded image . Then, the result is sent back to the Web Application to let the user see the result info.



Detailed Description:

Data flow diagram is a way of representing a flow of a data of a process or a system. As details of the system reference diagram, it shows that a user is going to upload a picture (an interface with user), it stores on the Web Application and is analyzed by the IBM Watson. Then, the result info is sent into two separate ways; one result is sent the confidence log of the researcher; secondly, it is also sent to the Web Application to show the result info on the user screen.

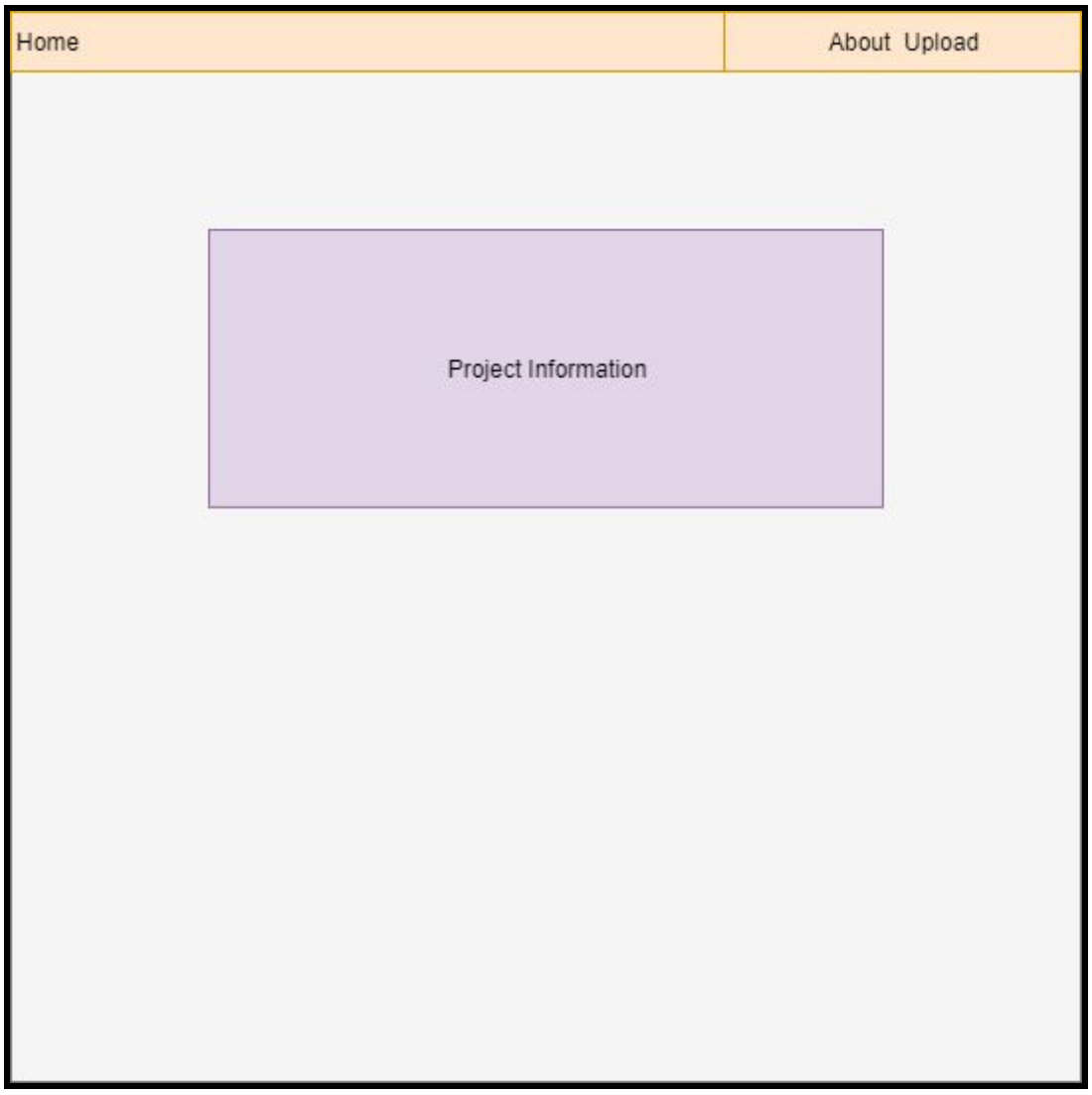
Derived Program Structure

- Marine Bird Identification uses the web application as its interface. It is similar to a lively library which gathers everyone who feels interested in the information of marine lives.
- We are going to design this web application allowing everyone to be able to search through their images. Once using it, each user will have access to the information in

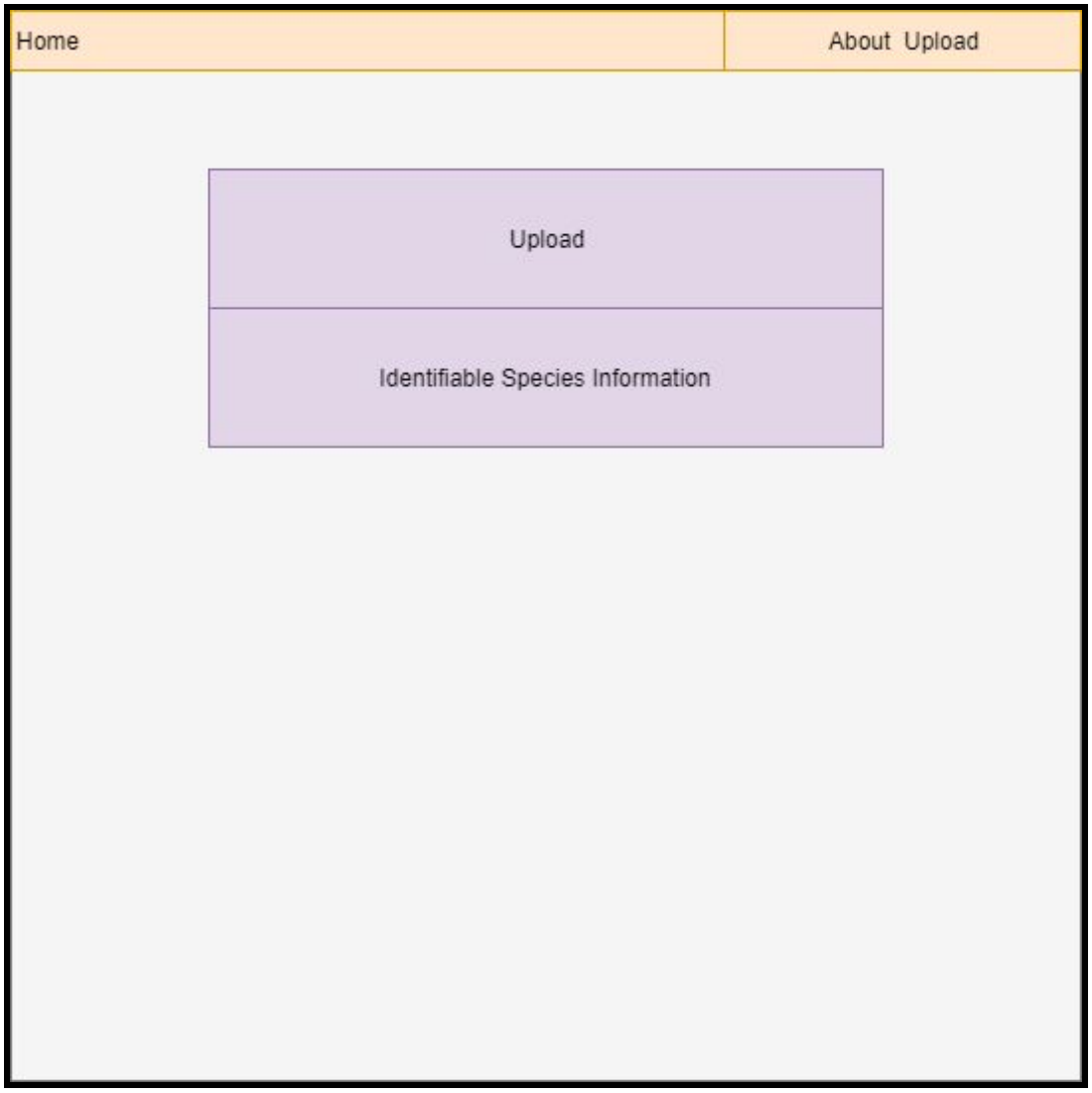
the storage. However, the number of marine birds is limited in NC, users should awake about that before they use it.

4.3 Interface Design

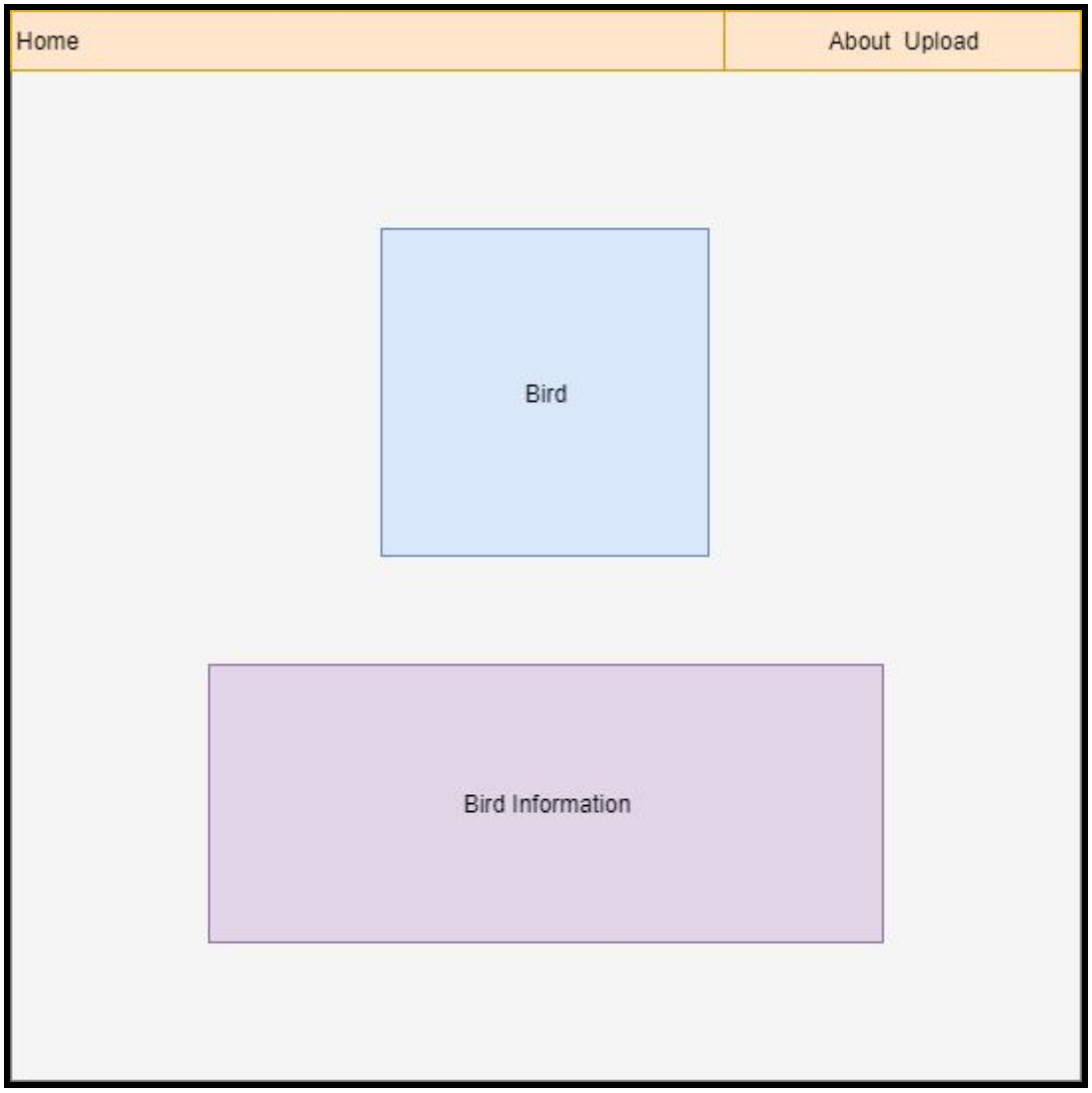
4.3.1 Project Information Screen



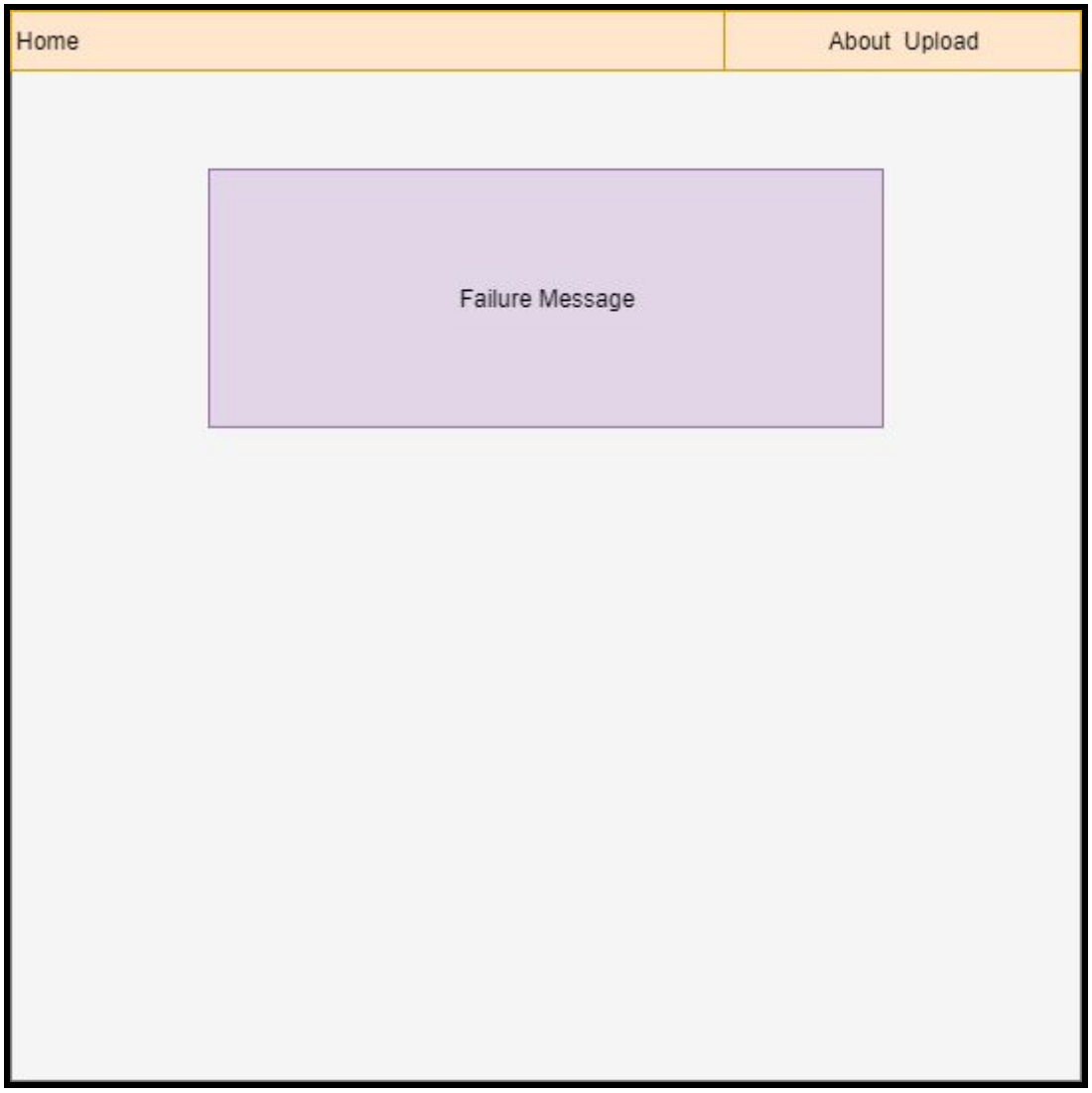
4.3.2 Project Upload Screen



4.3.3 Project Results Screen



4.3.4 Project Failure Screen



4.4 Procedural Design

For the processing narrative, if a user wants to know what bird they're looking at they can open our webapp. After the user opens the webapp, it'll need permission to use their camera, and then when the user take a picture, it'll check with the IBM Visual Recognition tool. If it has enough confidence it'll send back the species and other information, otherwise it will have to send back an error message, asking the user to take another picture.

The interface designed for this project will include three main guidelines: consistent colors across its entirety, a confirmation button - for taking and keeping a picture before submitting it, and a consistent layout for both mobile and desktop.

Multiple design languages are followed, to allow for greater flexibility in design, but narrow enough to follow consistently. One language we are using is UML or Unified Modeling Language, which helps create models and diagrams to base our design and planning on. The

second language being used is HIG or Human Interface Guidelines. Following HIG guidelines allows for a more consistent and user-friendly interface.

Multiples modules are required for our project, the first being an upload function that allows our users to provide a picture to be classified. Second, a confirmation function to make sure the user wants to upload the photo they have chosen or if they would like to retake or choose another picture. Next there is a procedure to request IBM Watson resources in order to classify the correct species in the provided photo. The last anticipated module is to parse through IBMs response and either display the species classified in the photo, or to display an error and possible species with a confidence rating. For this project only a single internal data structure is planned, a database of bird species, including information about them.

As for any restrictions planned for implementation, once a model has been trained with a dataset, strict analysis needs to be done to make sure the model is correct and can be used for judging bird species from photographs. Once the model is trained, and complete for simplicity's sake it is best not to allow users to affect our dataset. One possibility for a changing dataset may include researchers adding pictures or having peer reviewed photographs added into our dataset.

5. Test Plan With Test Specifications

Our test plan includes in-depth testing top-down testing of all components the user will interact with. Current testing plans are for content, UI, security, and components of the web application. Content testing will involve confirming if the information in the sqlite database is accurate and complete, the aesthetic is consistent and correct, and that all written information is formatted correctly with no typos. UI testing includes checking all links to ensure they work correctly as well as ensuring the site is designed to be user friendly, and follows the guidelines provided by HIG. Due to needing the user to input an image, security testing will focus around only allowing a PNG or JPG to be uploaded. Finally, component testing will have most of our focus, as it includes testing the integration of the web application and visual recognition model.

Some of the possible tests will include: testing all pictures of birds in dataset, testing dimly lit pictures of birds, testing pictures of non-birds, testing pictures with multiple birds, testing pictures of birds not in dataset. As for when we plan to test our work, the testing will occur after basic implementation is completed, but early enough in the development to make any necessary changes. In order to test this web application, we will use an independent tester from members of the team not involved with the making of the web application itself. Any of the errors that a user could experience will be found and fixed beforehand by following how a user would navigate the application.

5.1 Functional Testing

In order to test our data, and the functionality of the system, we will employ two tactics. To test the identification feature, on a local copy of the software we will run a number of photos through the system. These tests will include photos that are blurry, dark, bright, or otherwise unusable to see how the system responds. Also we will test the system with birds that are in the data set, that are not in the set, and also we will run through images containing nothing similar to a bird. On the front end side, similar tests will be run, along with confirming that the system will work with different file types and sizes. Once a near production level product is ready, we will also test how the system performs on mobile, and on a web platform.

To test the functionality of our web application, we look to make sure we get the correct output with our input. We first send watson an image file, either a jpg or png and expect a JSON file to be returned. No other file type is allowed to be uploaded to our website. After the user uploads their picture and we receive the JSON file, we can parse it. This file once it has arrived is parsed and after we send Django the class ID and we expect to get the correct bird back as an output.

5.1.1 Acceptance Test Cases:

ATC1: When testing all pictures of birds in dataset, the system must be able to recognize the species and differentiate between birds in the dataset.

ATC2: When testing dimly lit pictures of birds, if the species is in dataset it must be able to recognize the bird unless the image is extremely underexposed then it will return a low confidence score or a confidence score of 0.

ATC3: When testing pictures of non-birds, the system will either respond with a very low confidence rate or 0.

ATC4: When testing pictures with multiple birds, if multiple are in focus it will most likely fail to recognize the individual bird as the data set we have trained on is mainly single bird pictures. Otherwise it will return a confidence score on the bird most prominent in the photo.

ATC5: When testing pictures of birds not in dataset, the system will return a low or zero confidence score.

5.2 Structural Testing

Due to the required functionality for our project being reliant on IBM's Watson, much of the coding complexity is not present as would be in a more traditional software focus. With less complexity comes less of a need for testing, and our attention can be focused on the functionality of the application as a whole.

5.3 Integration Testing

Due to our application being fairly simple, little testing is needed to ensure it works as a whole. As mentioned with our functionality testing earlier, each test will be laid out with an expected result to ensure we have integrated our webapp with IBM Watson correctly. To test integration with IBM watson and our web application, we first check the homepage page. Next we click the upload button, and when we arrive at the upload page we upload a photo before getting taken to the results page.

5.4 Test Evaluation

5.4.1 Test Cases And Actual Test Results

- Front end tests
- All links work correctly, manual verification.
- All pages work correctly, manual verification.
- Upload function works correctly, manual verification.
- Submit function works correctly, manual verification.
- Security function works correctly, manual verification.
- When inputting a .doc, .pdf, .exe, .txt, the system does not accept the upload as expected.

5.4.2 Identification Tests

All well exposed image files of birds included in the data set work correctly and give confidences 93% or greater. When inputting a picture of a laughing gull upside down, the system incorrectly brings up a picture of a snowy egret with a confidence of 73%. When inputting a picture of multiple canadian geese the system correctly identifies the species with a confidence of 93%.

When inputting a picture of a bright picture of a snowy egret the system correctly identifies the species with a confidence of 93%. When inputting a dimly lit picture of a snowy egret the system correctly identifies the species with a confidence of 93%. When inputting a picture of an underexposed picture of a killdeer the system incorrectly identifies the species as a canadian goose with a confidence of 87%.

When inputting a picture of a toucan the system incorrectly identifies the species as canadian goose with a confidence rating of 69.5%. When inputting a picture of a potato the system incorrectly identifies the species as ring billed gull with a confidence rating of 93%. When inputting a picture of a blurry killdeer the system correctly identifies the species with a confidence rating of 93%. When inputting a picture of a blurry greater yellowlegs the system correctly identifies the species with a confidence rating of 93%.

5.4.3 Summary Of Testing Experience

Our errors were mainly the result of Watson Visual, not the front end the team designed and built. If we had more time to fine tune the Watson side of our system, we would likely be able to increase the confidence scores and lower the amount of false positive identifications. While running through the final project tests as a whole, we did not find any errors with the front end system. The errors that could be removed would be those of misidentified species, which would likely be removed or severely lessened after much more training, that which we do not have access to with an IBM student login, at which point we could confidently rely on the system as a whole.

6. Documented Source Program And The Tested Object Code

6.1 Overview

- Overall description of implemented system
- Summary of programming experience

6.2 Source Code Listing

Our project used the Django Framework to implement the web interface. Django provides many plug-and-play aspects that made programming easy. Below are the various files Django uses to orchestrate the program and their purpose.

A Django project is a complex endeavour typically split into many parts.

Django Framework [visualbirds]

Title: settings.py
Purpose

The settings.py module contains all of the critical settings that drive the visualbirds project.

Modifications:

Modification: In order to store files on the 'server' in order to send to IBM Watson for analysis, a "MEDIA_ROOT" and "MEDIA_URL" were added. These simply give the project a physical place to store files such as photos.

Title: urls.py
Purpose

The urls.py module determines the links that are internal to the project.

For example, mapping out 'home', 'login', 'logout', would all need urls stated within this module.
Modifications:

Modification: Added the path that links the main project to the bird_identifier app.

Modification**:** Addied the static MEDIA_ROOT option to establish a place for photos to be temporarily stored.

Title: wsgi.py
Purpose

The wsgi.py module runs internal Django operations.
Modifications: N / A

Django Framework [bird_indentifier]

Title: admin.py
Purpose

When 'models' within Django are created they can be registered within the admin.py module. This allows a user to access an admin interface and input each individual object.

In our case we registered the 'Bird' model and were able to add each bird individually as well as information associated with said bird.
Modifications:

Modification: Registered the 'Bird' model with the admin.py module.

Title: apps.py
Purpose:

The apps.py module allows the bird_identifier app to be recognized by the parent Django Framework.

Modifications: N / A

Title: forms.py
Purpose

The forms.py module allows custom forms to be registered with Django for use through the app. This allows for a more pythonic way of creating forms instead of relying on pure HTML.
Modifications:

Modification: Created the UploadFileForm form. This allows Django to capture whatever image is uploaded by the user for use.

Title: models.py
Purpose

The models.py module allows a user to create new models within the Django framework. Models are synonymous to a database table.
Modifications:

Modification: Created the 'Bird' model. This is equivalent to a database table that would hold information about our birds. In fact a Django model is simply a high level interface instead of writing SQL tables by hand.

Title: tests.py
Purpose

The tests.py module allows a user to write automated tests for a Django project.

Modifications: N / A

Title: urls.py
Purpose

The urls.py module allows users to create links within a web app.
Modifications:

Modification: Registered the various paths needed by the Django app to navigate the website. For example, registering the 'results/' path, allows the user to type in the search bar "www.EXAMPLEWEBSITE.com/results/", and return a valid link.

Title: views.py
Purpose:

The views.py module is the most critical part of the Django framework and contains the logical behind the web application.

Each ‘page’ of the web app is represented as a view within Django. The upload, information, and results views each contain logic that drive their functionality.
Modifications:

Modification: Created the *index*, *upload*, *success*, *failure*, and *results* view as well as the helper functions that drive the program logic.

Django Framework [bird_identifier/views.py]

The core logic behind our project is contained within the *bird_identifier/views.py* module.

Function: getResponseFromWatson
Purpose:

This function sends the file (JPG / PNG) supplied by the user and sends it to IBM Watson for analysis.

Parameters: file - The JPG / PNG supplied by the user.
Returns: classes - A JSON Object that contains IBM Watson’s results.

Function: getConfidenceLevels
Purpose

This function extracts the confidence levels from IBM Watson’s returned JSON Object.

Parameters: classes - A JSON Object that contains IBM Watson’s results.
Returns: confidence_levels - The array that contains the confidence levels from the JSON result.

Function: separateConfidenceLevelsIntoWorkable
Purpose

Due to the IBM Watson JSON Object containing nested JSON Objects and Arrays, an intermediary function was required to dig to the level needed.

Parameters: confidence_levels - The array that contains the confidence levels from the JSON result.
Returns: list_of_confidence_ratings - An array that contains tuples of ‘bird_id’ and the corresponding ‘confidence_rating”

Function: determineHighestConfidenceRating
Purpose

Parses the given array and finds the highest probable bird.

Parameters: confidence_rating - An array that contains tuples of ‘bird_id’ and the corresponding ‘confidence_rating’
Returns: (bird_class, maxRating) - A tuple that contains the highest probability bird and that probability.

Function: index
Purpose

The index view renders the homepage of the application. It uses the base.html template and information.html template.

Parameters: request - The web request
Returns: The rendered homepage.

Function: results
Purpose

The results view renders the results of the application. It uses the base.html template and results.html template.

Parameters: *request* - The web request | *watson_id* - The id of the highest probable bird
Returns: The rendered results page.

Function: failure
Purpose

If the confidence rating of the highest probable bird is not high enough, the failure.html template will be displayed, indicating to the user that the image is not any of the identifiable birds.

Parameters: *request* - A web request
Returns: The rendered failure page.

Function: upload

Purpose

The upload view allows the user to upload an image.

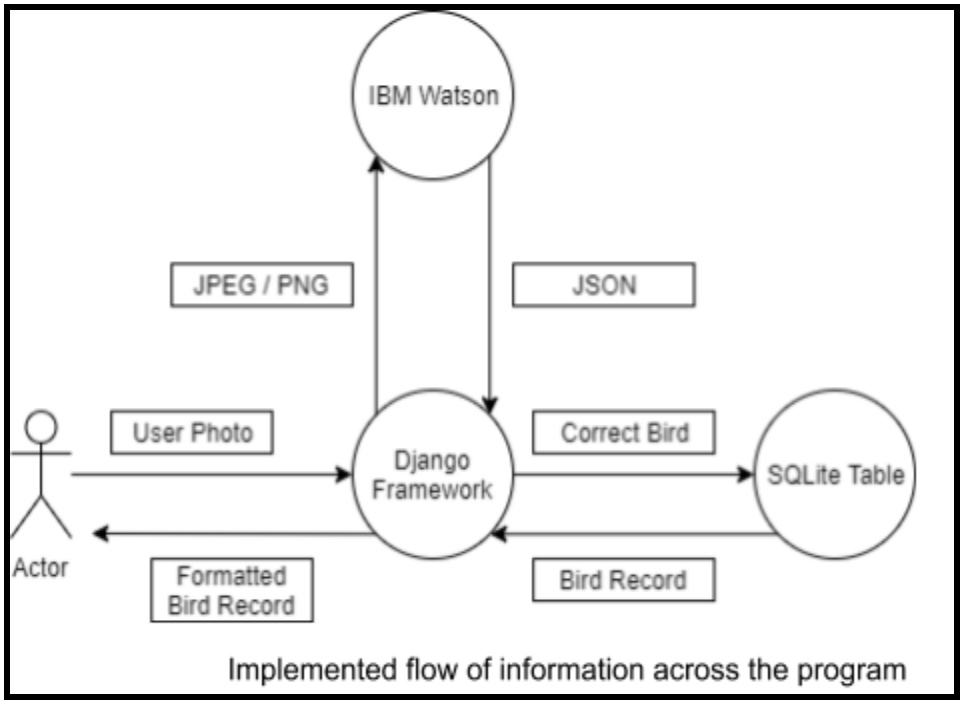
Parameters: *request* - A web request
Returns: The rendered upload page.

---

7. Software Design Implementation

7.1 Data Design Implementation

Data design was implemented as planned with no significant changes. Below are diagrams outlining data design of this project.



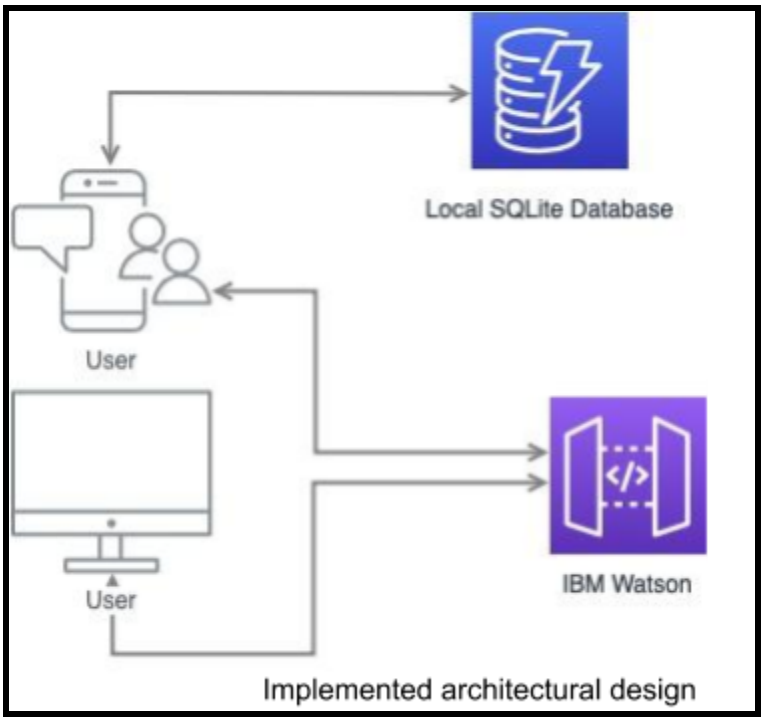
Implemented flow of information across the program

Implemented database design

7.2 Architectural Design Implementation

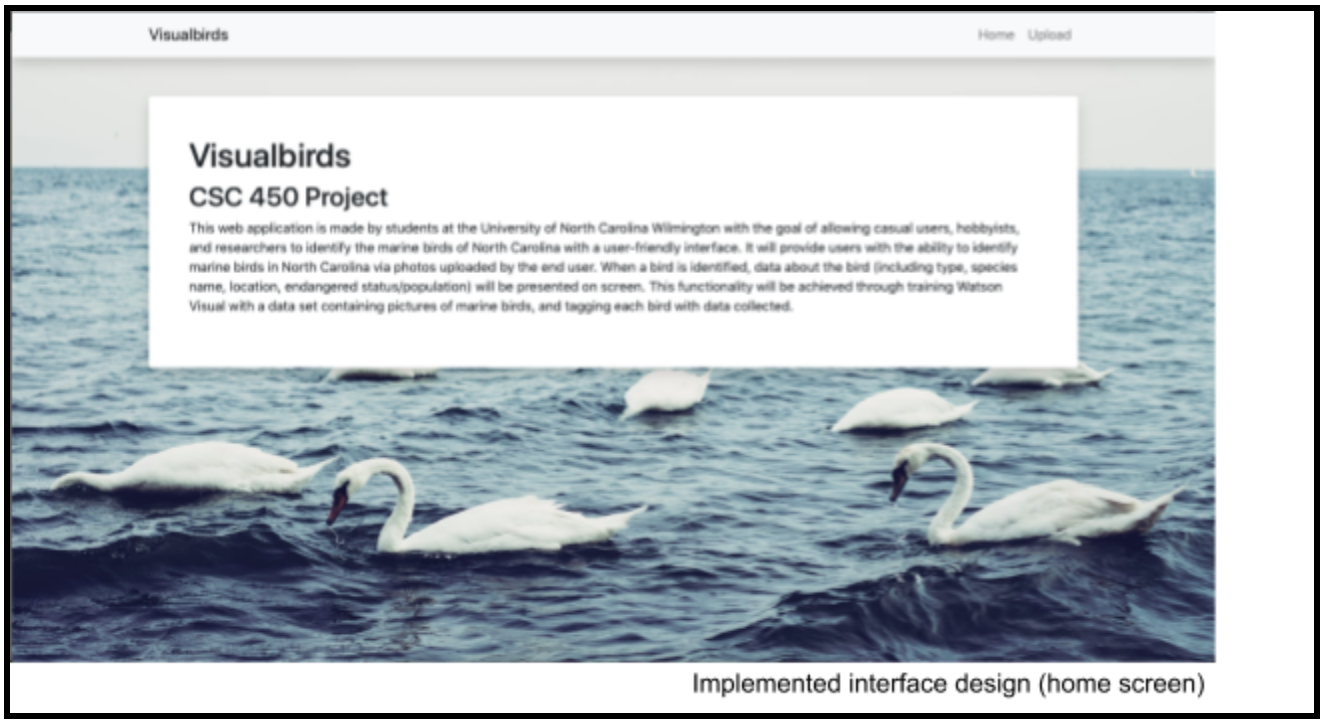
The design architecture implemented varies slightly from the planned architecture. Originally, it was determined we would use IBM Watson database. However, to expedite software development process, minimize costs, and maintain full control of the database we utilized a local SQLite database.

To make this project highly available will require future versions to host a server with a persistent database either on a cloud service or physical server, possibly UNCW Satoshi server.



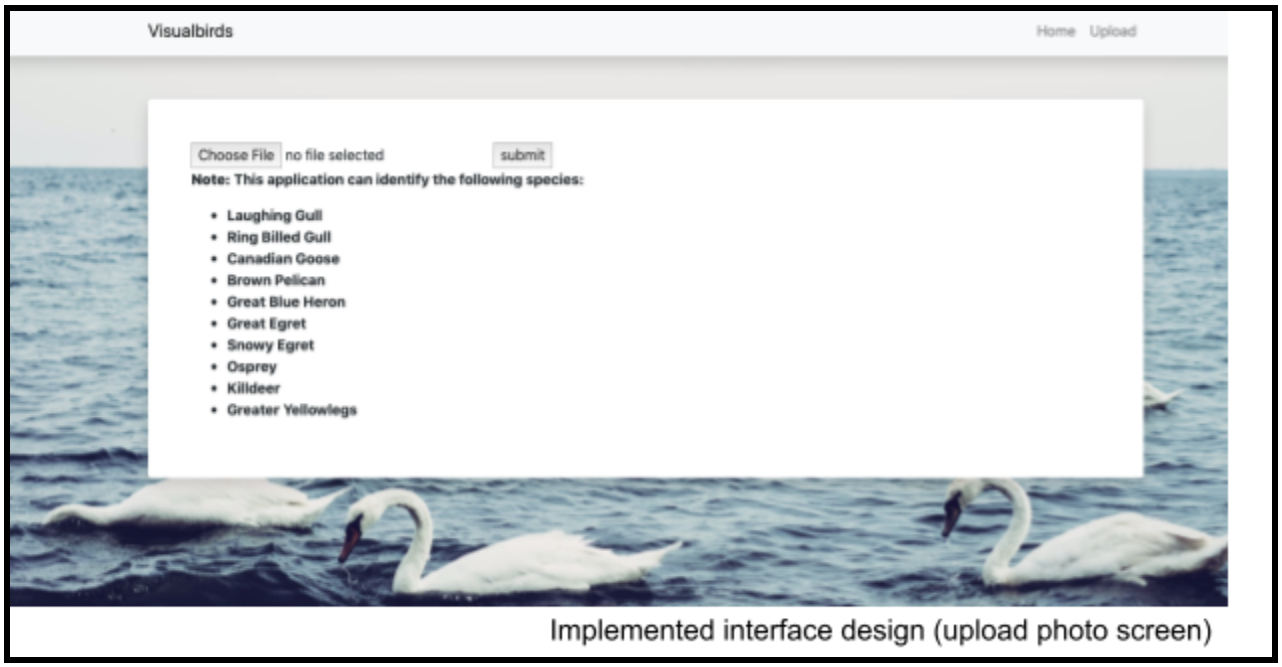
7.3 Interface Design Implementation

7.3.1 Project Information Screen



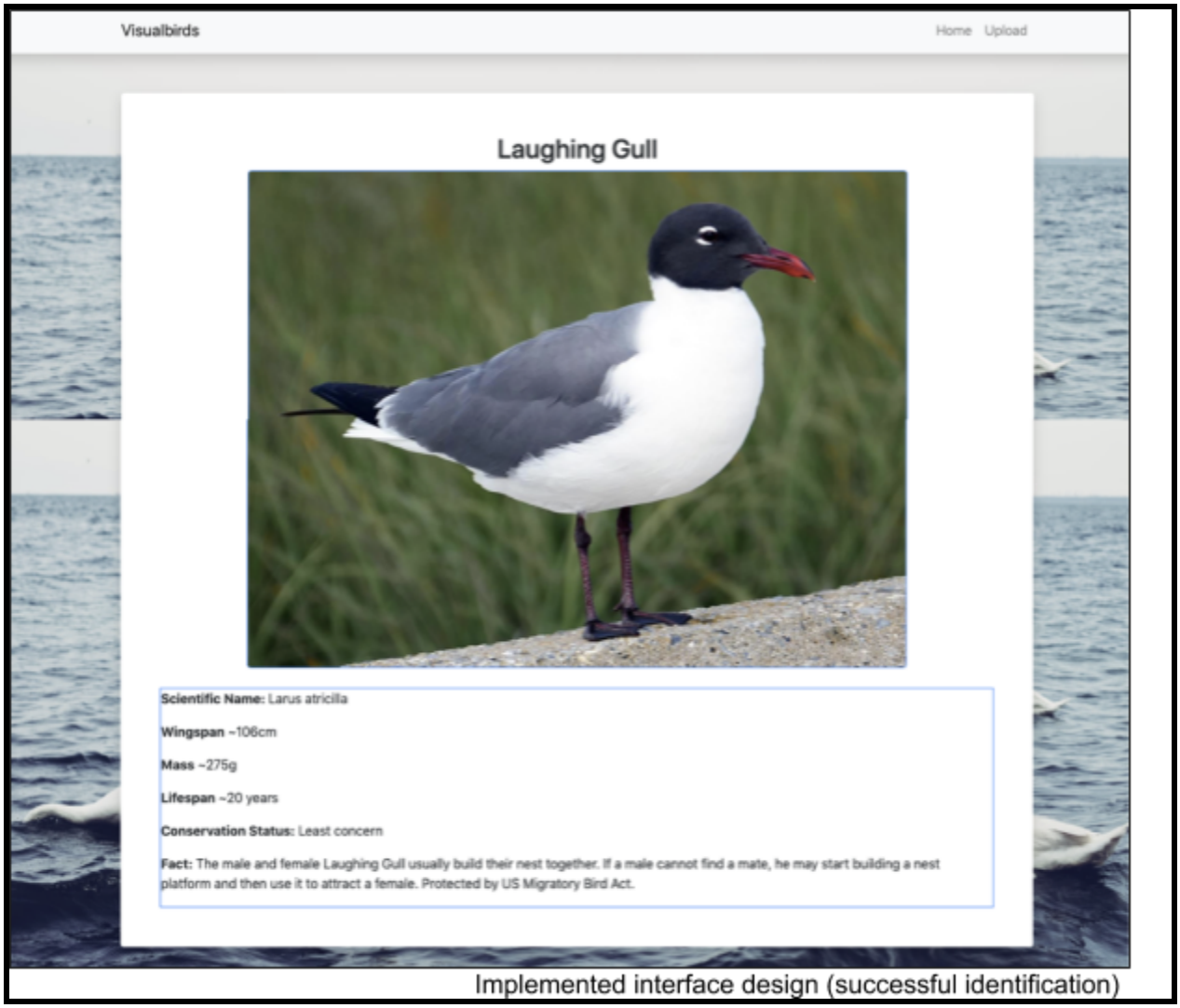
Implemented interface design (home screen)

7.3.2 Project Upload Screen



Implemented interface design (upload photo screen)

7.3.3 Project Results Screen



Implemented interface design (successful identification)

7.3.4 Project Failure Screen

Implemented interface design (unsuccessful identification)