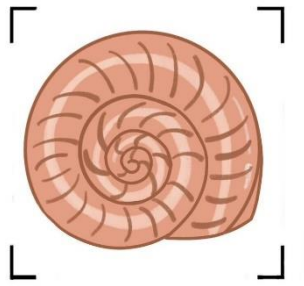
Spring 2023



SeeShell: a mobile app for image recognition of sea shells

UNC CSC 550: DR. VETTER

ALEXANDER JOHNSON, ERON NEILL, LEAH SCHNEIDEREIT, MICHAELA PIERCE, THOMAS SMITH , AHMED ELGAZAR

Contents

1. Introduction

1.1 Project Purpose Statement

During the Spring semester of 2023, Alexander Johnson, Eron Neill, Michaela Pierce, Leah Schneidereit, Ahmed Elgazar and Thomas Smith were members of a team working on a full stack application called SeeShell as part of a Software Engineering (CSC 550) group project. The purpose of the project is to not only develop a working software system, but to fulfill the learning objectives of the class. These objectives can be found in UNCW Course Catalog in SeaNet under CSC 500, and cover concepts such as the Software Development Lifecycle, project planning, requirements gathering, and good software design practices. The application will use machine learning to match and identify see shells based on images that users capture. The deadline for the application is May 4th, 2023, with deliverables due throughout the duration of the project.

1.2 Project Statement

Shell collecting is a popular hobby for many people living near and visiting the coast. Anyone who has visited Wrightsville Beach, or the surrounding beaches of Southeast NC has surely either partaken in the activity themselves or noticed someone searching for shells near the water's edge. The ocean tends to surprise beachgoers with unique and mysterious treasures as they walk along the sand. Seashells are not only a beautiful work of nature, but each one holds a story. Those of us who find these shells on the beach may not only be interested in the natural beauty but the history behind the creature that created the shell that has captivated us.

The SeeShell mobile app has been designed to enhance the shell collecting experience for beachgoers and avid shell collectors alike. The app allows users to take a picture of a shell that they found and upload it, where a machine learning algorithm will utilize image recognition to determine the species of shell that was found. From there, the user can discover the story behind that very shell they hold in their hand. Features have been designed to appeal to the average beachgoer as well as avid collectors. Not only does the app provide an educational experience for children and adults, but it provides the opportunity to manage a digital collection of the shells that collectors find. The user will be able to learn more about the species, its evolutionary history, and the area that it originated from. This app presents an interesting and educational experience for those who enjoy learning about our mysterious ocean as they walk along the beach.

1.3 General Considerations

Various forms of deployment have been considered for the SeeShell application. A mobile application has been chosen because the team believes it will provide the most value to the user. If the user has an application downloaded on their smartphone, they are more likely to utilize the app as a resource when walking on the beach, rather than having to navigate to a web browser or wait until they have access to a desktop machine. The ease and accessibility of a mobile app is part of the reason why the SeeShell application will add value to a user's beachgoing and shell collecting experience.

IBM Watson has been chosen for hosting the machine learning algorithm. The team chose IBM Watson based off recommendations from the primary stakeholder as well as taking other project constraints into consideration.

The dataset to be utilized for this application contains over 10,000 shell species. For performance and time considerations for this project, the number of species may be reduced, or the classification may be reorganized.

Users are expected to use the application at the beach, where internet connection is known to be unreliable at times. The application will allow users to upload an image even without internet connection. Once internet connection becomes available, the image will be transmitted to the cloud ML platform for recognition and matching.

1.4 Scope and Limitations

The purpose of this section is to outline and define the scope of features expected to be implemented into the first release of the SeeShell App. This section will not only define what will be included in the system but will effectively define the boundaries of the scope of this project. This ensures that time is being spent on the most critical features, and the project team stays on track to deliver a working and robust mobile application. The scope of this project will be outlined with specific limitations in mind. Features are prioritized to ensure that a valuable application is delivered. If a limitation is realized throughout the development period is realized, then lower priority features will be considered future work for future releases.

A major limitation the project team faces is the timeline of the project. The team has 16 weeks to develop a working mobile application. The accelerated timeline places constraints on the amount of features that can be included in the application. Because of this, the team has decided to refer to the final product for the CSC 550 course as the initial release. This means that the final product at the end of the project will include basic functionality with the opportunity for future work to implement more complex functionality.

1.4 Glossary of Terms

The glossary defines terms that may not be familiar to all users. This is especially important if the application is expected to reach a wide range of users[1].

Mollusks	An invertebrate of a large phylum that includes snails, slugs, mussels, and squids. They have a soft unsegmented body and live in aquatic or damp habitats, and most have an external calcareous shell. Mollusca are the second largest animal phylum on Earth after arthropods.
Taxonomy/Taxa	A taxonomic group of any rank, such as a species, family, or class.
Species	A group of living organisms consisting of similar individuals able to exchange genes or hybrids. The species is the main natural taxonomic unit, ranking below a genus and referred to by a Latin binomial, e.g. *Homo sapiens*.
Biota	The animal and plant life of a particular region, habitat, or geological period.
Classification	The action or process of classifying something according to shared qualities or characteristics.
Genus	A principal taxonomic category that ranks above species and below family, and is denoted by a capitalized Latin name, e.g. *Leo*.

[1] The glossary information was obtained from the following source: Molluscabase. (2014). Molluscabase.org. https://www.molluscabase.org/

Blurb	A brief description of the species intended to be displayed to the user after an image match
Digital collection	The digital collection refers to the location where users can save and manage the images they take and upload

2. Software Requirements Specification

2.1 Major Features

1. **Image capture and recognition:** the app will allow the user to take a photo or upload an image of a shell using their phone camera, and use image recognition technology to match the shell to a species from a database
2. **Image saving:** the app will automatically save matched images to a digital collection. The user will be able to view the images and species information at any time
3. **Account management:** the app will prompt users to create an account or log into an account where their digital collection information is accessible and up to date
4. **Digital collection management:** the app will allow users to manage the photos in their digital collection. This includes features such as adding a title to the image and deleting images from the collection

2.2 User Requirements

The main role of the users of this app will be the general user. The features of the system are designed for the everyday person, so it is not necessary to define various roles for the system. Anyone using the app will experience the same functionality/needs as one another. User requirements have been defined for the "user" role.

2.2.1 Roles

User	Individual who uses the app to take/upload images of shells, view information on the species match, and manage a digital collection of the shell

2.2.2 Functional User Requirements

Feature	**Requirement**	**User Story**
Image capture and recognition	Capture photo of a shell	As a user, I want to take/upload an image of a sea shell so I can learn more about the species
	Allow user upload from phone gallery	
	Display best match of shell from ML and provide information about shell	
Account management	Allow user to log in to account with credentials	As a user, I want to have an account so that my information is saved and I can access it on another device
	Allow user to create account	

	Allow user to change password if forgotten*	As a user, I want to be able to keep my account even if I forget my password
Digital collection management	Allow users to delete image from digital collection	As a user, I want to be able to manage the images in my digital collection to keep it organized
	Allow users to view information about shells in collection at a later date	
	Allow users to add title/location to image*	

2.3 System Requirements

2.3.2 Functional System Requirements

Feature	**Requirement**	**User Story**
Image capture and recognition	System will be able to recognize and match shell species based off an image	As a user, I want the application to recognize the correct species based on shell images
	System will upload captured image from client to server	
	System will upload user's gallery image from client to server	
	System will crop uploaded image from user's phone	As a user, I want to be able to upload an image I previously took to be matched on the app
	System will perform matching analysis when if/when internet is available	As a user, I want to be able to capture/upload an image for processing even when I don't have service so that my image is not lost and can still be matched when service is available
	System will include a database of shell species	As a user, I want to be able to view useful information about the matched species so that I can learn more about the shell
	System will be capable of producing proper queries to return species information when match is successful	
	System will connect to server to send and receive information about species	
Account Management	System will authenticate user credentials and allow users to use the app upon successful authentication	As a user, I want to have an account so that my information is saved and I can access it on another device
	System will query the database to add new user information when a new account is made	

* Given the time and experience constraints, these requirements had a lower priority compared to other functionality of the app, and were therefore deemed future work when it was realized that there would not be enough time to implement them

	User Sessions will be generated for each account. **	
Digital Collection Management	System will automatically save captured images	As a user, I want to have access to my library when offline so that I can view my digital collection without internet service
	System will save species information and shell photos locally	

2.4 Non-Functional Requirements

Performance	The ML algorithm will perform matching and return information in a reasonable amount of time
	App will be able to analyze photos that have varying backgrounds
Security	App will maintain security of user information
Usability	Structure of the app will be easy for the user to navigate
	User will be made aware of any errors such as: Server connection errors Authentication errors Species matching errors
Reliability	The application will operate failure-free
Availability	The app and client-server communication will have a reasonable uptime
	Users will be able to use the app even without internet connection

2.5 Data Requirements

Image Data	Image will be saved in a local file
	Image will be transferred to cloud provider for processing
	Image will be saved locally for digital collection
User Data	Authentication data will be maintained in database
Species Data	Species data will be maintained in central database
	Data will be obtained from the MolluscaBase [4]
	Species blurb information will be saved locally for digital collection

** User sessions were not implemented for the prototype and were deemed future work should the app be developed to go to market on the Play Store

[4] Species data was obtained from the following source: Molluscabase. (2014). Molluscabase.org. https://www.molluscabase.org/

Machine learning data set	Dataset obtained from a study done at the University of Macau[5]

2.6 Supplemental Diagrams

2.6.1 State Machine Diagram

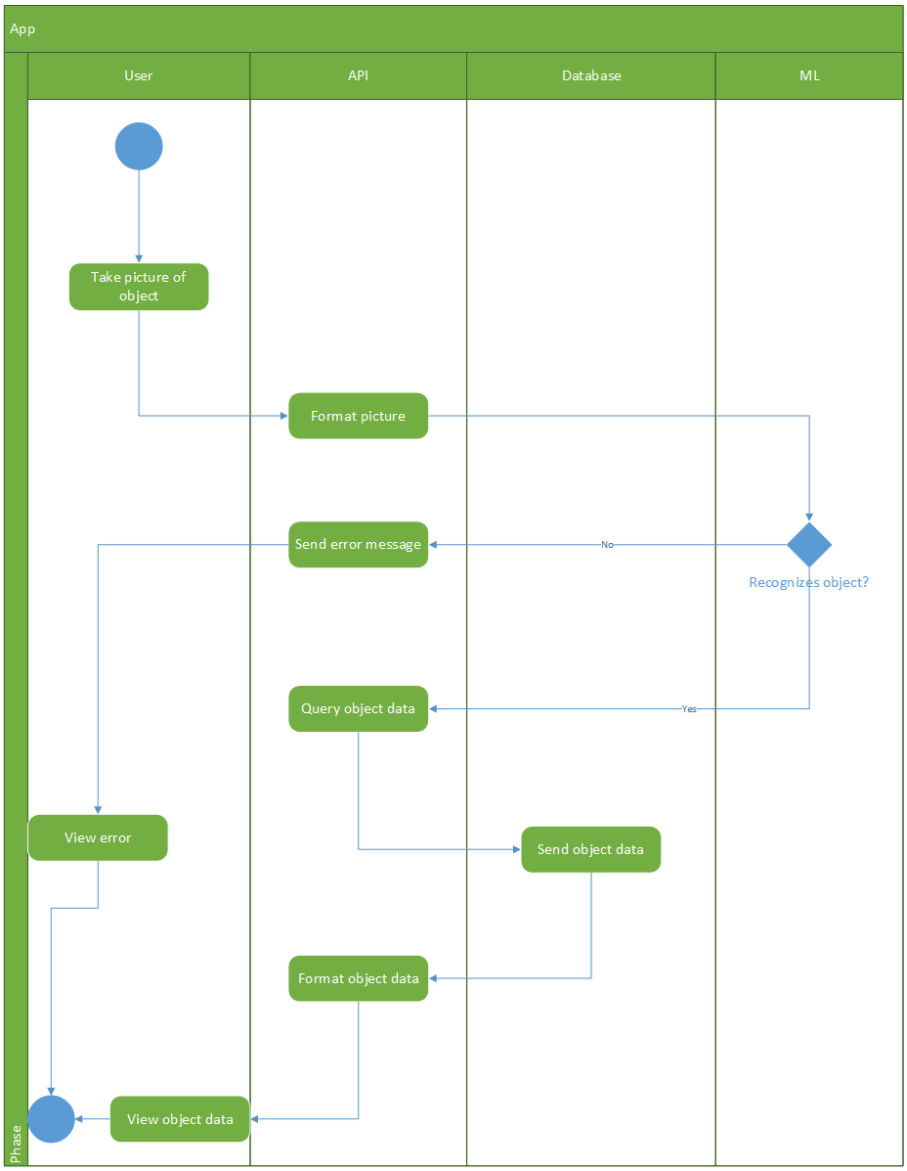
The state machine diagram describes the various states of the application upon different actions that are taken by the user. When the application is opened, the login screen is opened first. From there, the application can enter the upload image state or capture image state. When internet connection is available, the system can perform image analysis and recognition. If a match is successful, it can display the results. Otherwise, the system must return to the image capture page to try again.

2.6.2 Activity Diagram

The activity diagram provides a high-level overview of the major use cases of the system – taking an image that can be uploaded and matched to a species using machine learning, then viewing the information about the species. The diagram displays the flow of actions that the user takes while using the application, and how the system responds.

(see diagram on page 7)

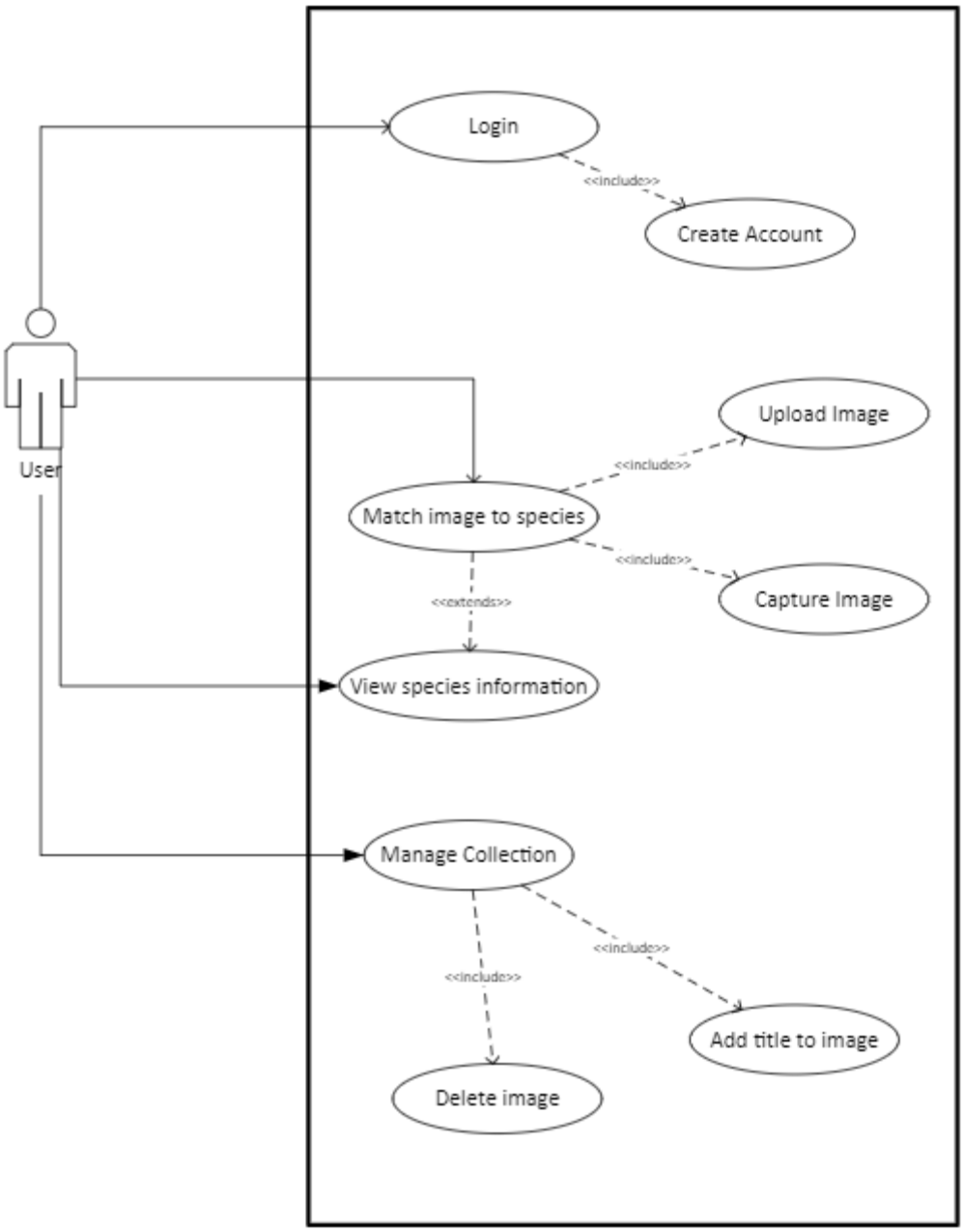
[5] Machine learning image dataset was retrieved from the following source: A shell dataset, for shell features extraction and recognition (2019). https://springernature.figshare.com/collections/A_shell_dataset_for_shell_features_extraction_and_recognition/4428335



2.6.3 Use Case Diagram

The use case diagram provides a high level and comprehensive overview of the use cases that may be presented by the user. The main use cases are login, match images, view species information, and managing the digital collection. From there, some use cases can be extended with further functionality. The login feature requires that a user create an account. The image matching requires that an image is uploaded, and the view species information is an extension of the image matching. Finally, the manage collection use case is extended by adding a title and deleting images from the collection.

(see diagram on page 8)


Login
<<include>>
Create Account
User
Upload Image
<<include>>
Match image to species
<<include>>
Capture Image
<<extends>>
View species information
Manage Collection
<<include>>
<<include>>
Add title to image
Delete image

3. Design Specification

3.1 Design Objectives

The system objectives are below. The functionality of the application will be prioritized over the appearance of the application. The interface will be configured based on time constraints. A priority scoring was assigned to each goal with 1 being the lowest and 5 being the highest. The objective is to consider time and scope of the project and choose higher scored goals before implementing lower scored goals.

Goals	**Objectives**	**Priority**
The mobile application will be user friendly and easy maintenance.	The application will use a simple GUI for the user to provide easy navigation.	3
The mobile application will support the Android platform.	The application will be functional on android devices.	5
The mobile application will support account creation and management.	Users will be able to create and delete an account using their email. Users will be able to change their password for their account.	4
The mobile application will have a camera function used by users.	Users will use the camera function in the application to take pictures of the shells.	5
The mobile application will use image recognition and machine learning to identify shells.	Image recognition will recognize the shells and provide a result of shell type.	5
There will be a journal feature that will allow users to store entries.	Users can store entries in the journal such as images, location, date and time, and user input.	3
There will be an educational feature that provides users with descriptions.	The application will return with a description of the shell identified to educate the user.	3
The mobile application will have a sleek design for users.	The GUI will be configured to provide an aesthetically pleasing interface.	2

3.2 Software Design Constraints

Software Design Constraints are limitations that should be carefully weighed to guarantee the product is delivered on time and meets all requirements.

- The interface design is limited to that of what can be done with Kivy as this is the main library that will be used for android mobile app development
- Time – the accelerated timeline of the project proposes a constraint on the amount of time that can be spent on lower priority features such as UI/UX design and individuality
- Mobility – there is an expected constraint concerning the mobility of the application because there is not yet a dedicated server for hosting the back-end services of the app.
    - Internet connectivity is required for running matching on the images

- ML – IBM Watson imposes a design constraint

3.3 Software Design Assumptions

Below is a list of factors that are assumed to be true when developing the application for users and developers.

- The user understands camera functions and basic Android functionality
- The user has access to cellular service (either at the time of taking the photo or at a later, reasonable time)
- The user has an email to create an account
- Project team members have access to necessary data
- The user will upload images of items that resemble shells
- The user does not expect matches for low quality/poorly defined images

3.4 System Architectural Design

The system consists of an all-python client-server architecture built with modularity and code reuse in mind.

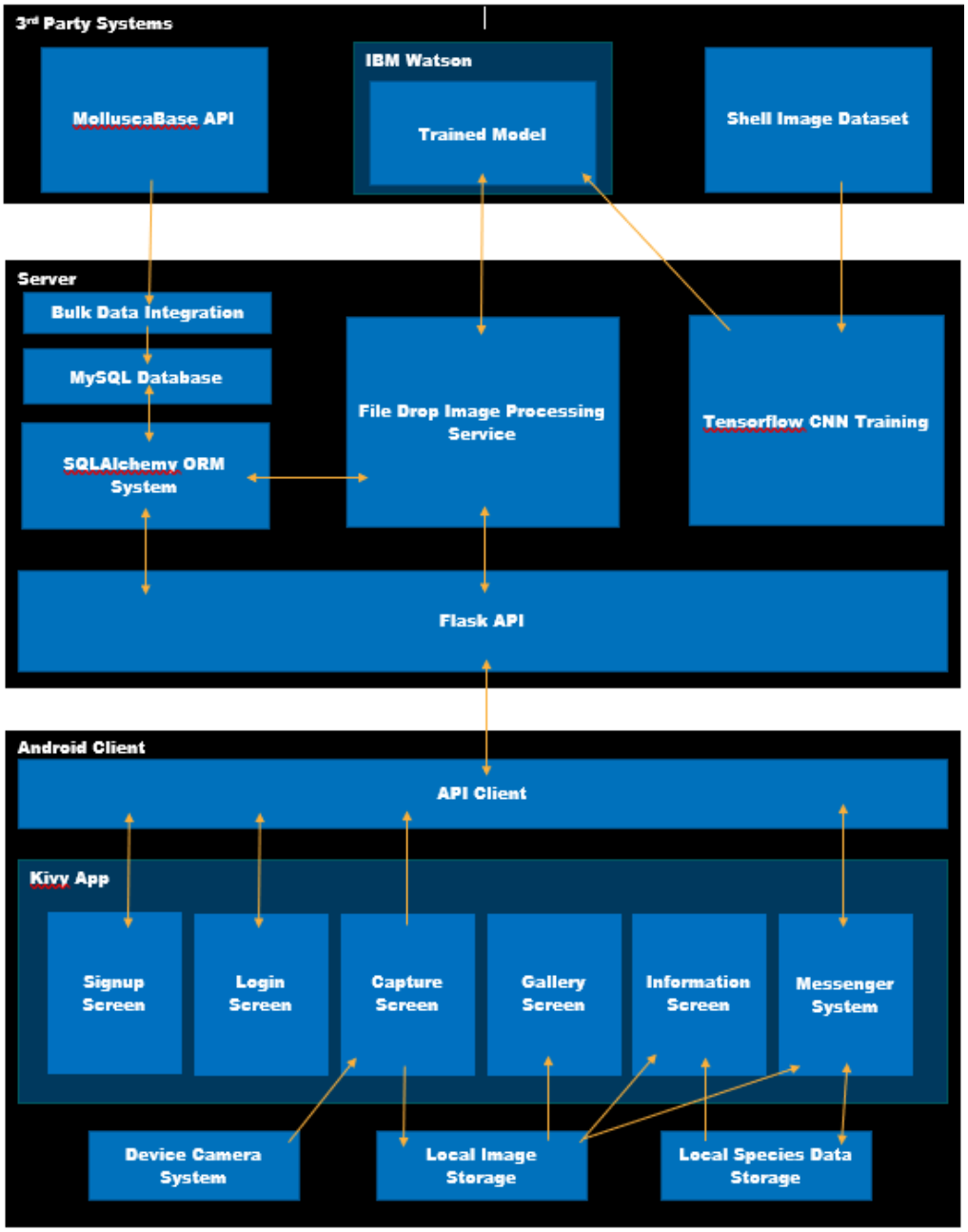
The client is designed to run on Android devices, although Kivy is a cross platform library and it has been run successfully on Android, IOS, Mac, Windows, and Linux with most being fully or near-fully compatible. The client interacts with the server through an API client library which provides python bindings for the SeeShell API.

The server is composed of four main parts, the API which exchanges data with the clients, the FileDrop Service which processes images and sends them to IBM Watson for classification, the MySQL database and SQLAlchemy ORM system, and the machine learning component, which trains models and sends them to IBM Watson for production deployment.

The system integrates with 3 external parties. An API based bulk data download from MolluscaBase,

uploading the trained classifier model to IBM Watson and inferencing it in production, and the shell image dataset from the University of Macau.

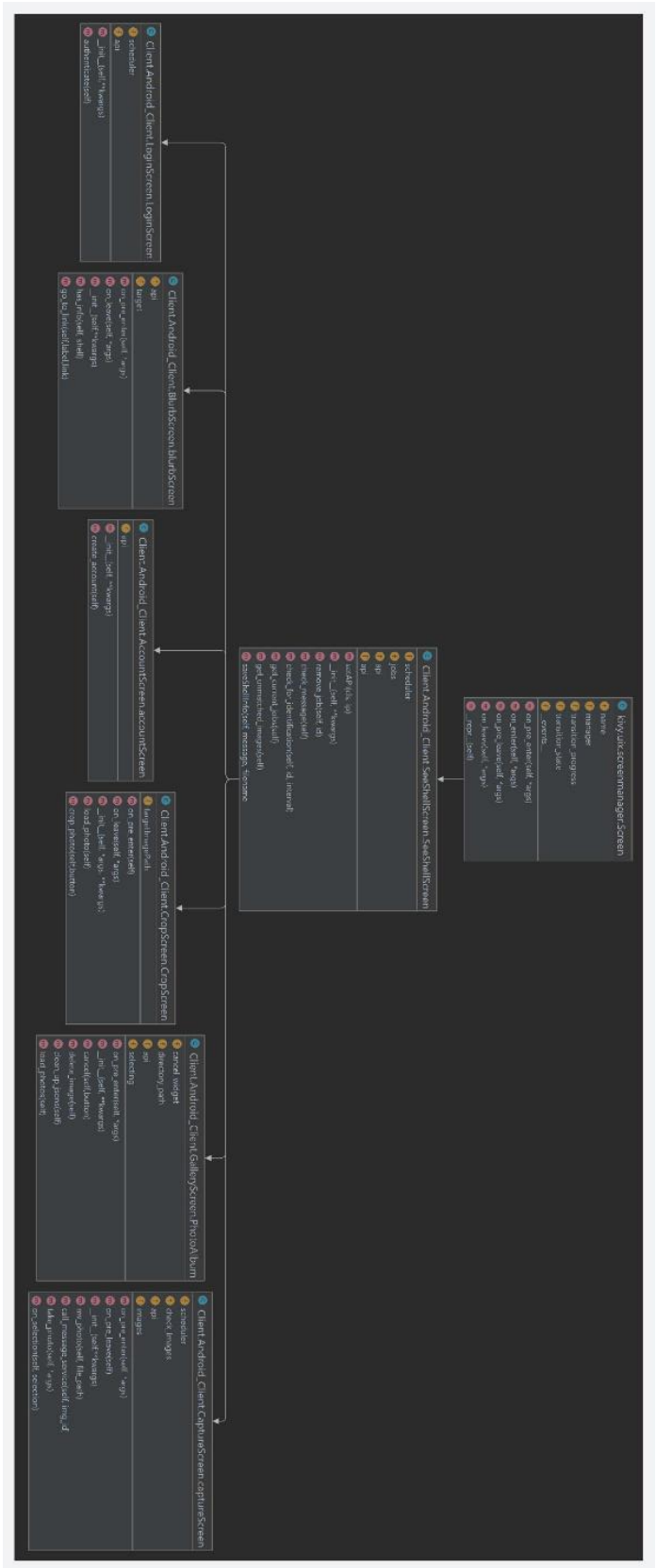
(see architecture diagram on page 10)



3.5 Software Structural Design

The class diagram below displays a high-level view of the classes that will be necessary for the mobile app. The diagram displays the relationships between the various screens, as each screen is implemented as its own class.

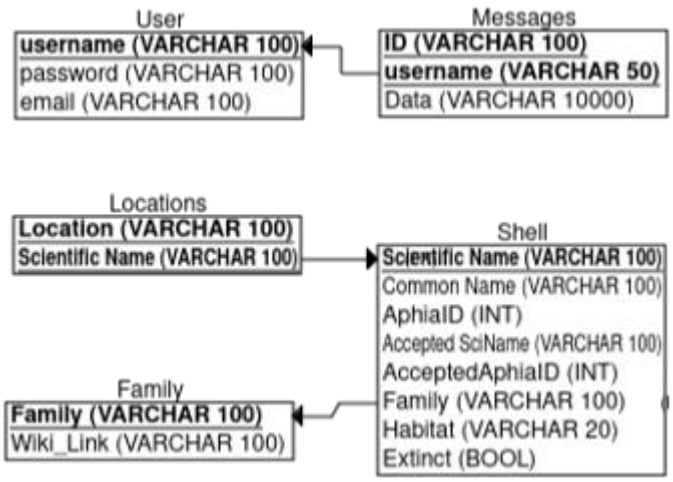
(see class diagram on page 10)



3.6 Data Design

3.6.1 Database Design

Below is an Entity Relationship Diagram (ERD) which details the structure of data storage in the server database. Shell information will be stored in the "Shell", "Family", and "Location" tables and will be used to generate information about the identified shell for displaying to the user. Shell images will be stored in a file structure on the server, but not on the database, and user images will be stored on the device in a similar file structure as well. Messages for communication of the client and server with the API will be stored in the "Messages" table.

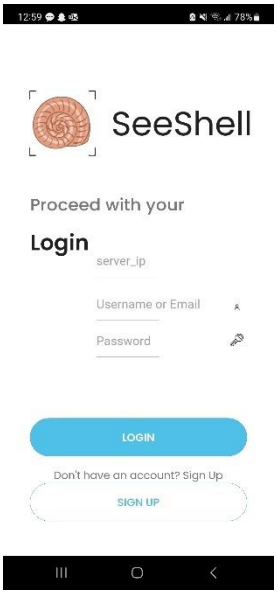


3.7 User Interface Design

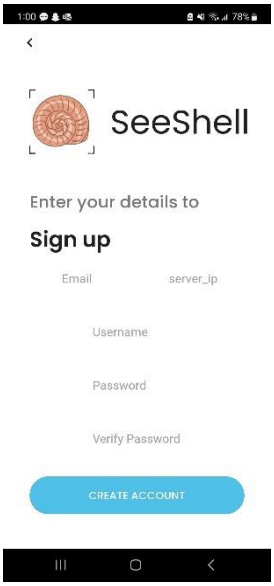
3.7.1 Overview of Screens

There were 5 functional screens developed for the user interface. The screens can be separated into four general categories. Each of the screens is numbered and will be referred to in section 3.8.2, Interface Description. The screens were developed based off wireframes previously developed in the design specification phase of the project. It is important to note that while some changes were made in the final product, the wireframes provided a baseline for understanding requirements and functionality. The 5 screens shown below represent the main functional screens. While there is other functionality built into these screens, the details of those were left out for the sake of simplicity in designing the interface (such as the upload from phone gallery screen feature which is not a screen that we specifically developed).

Functional Screen	Wireframe Screen
Login	1, 2
Image Gathering	3
View Species Information	5
Digital Collection	4



Screen 1 - Login Screen



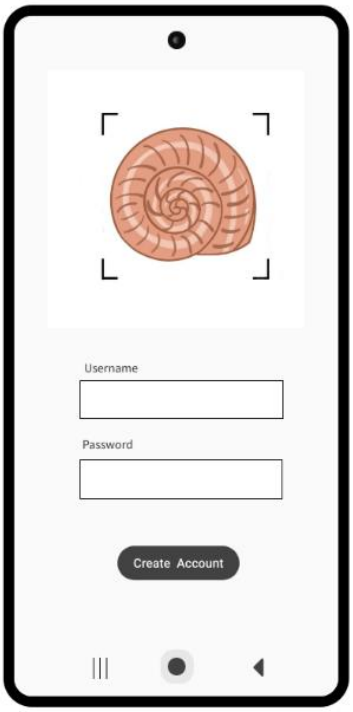
Screen 2 - Create Account Screen

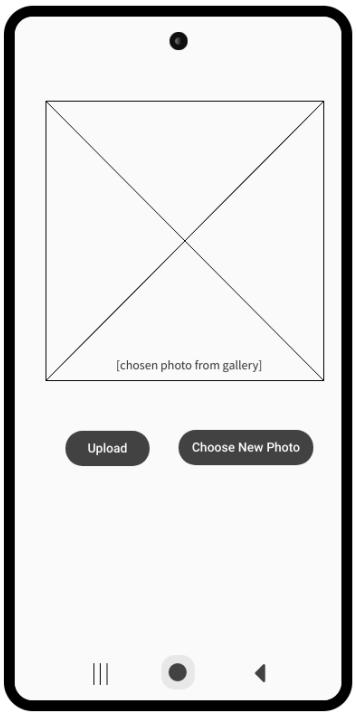
Screen 3 - Capture Screen

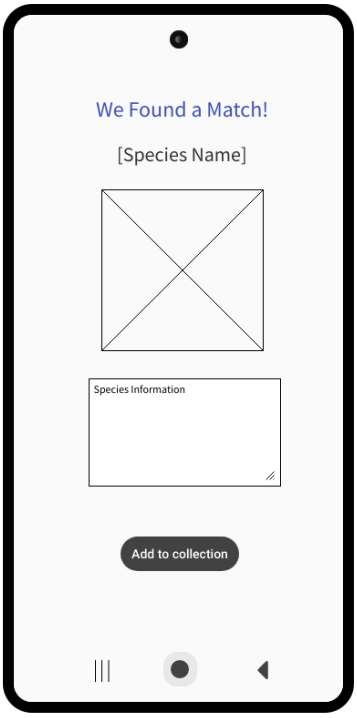
Screen 4 - Gallery Screen

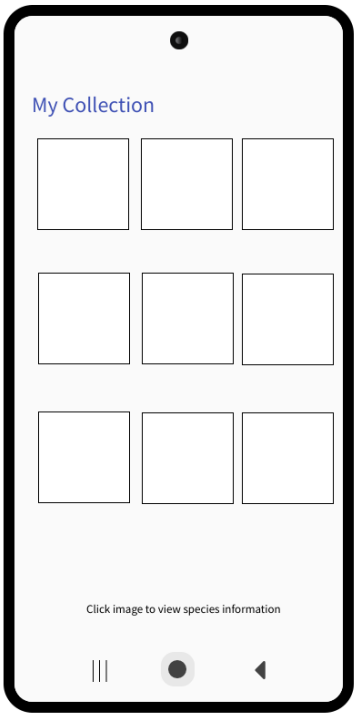
Screen 5 - Blurb Screen

3.7.1 Wireframes









3.7.2 Human Machine Interface Design Rules

Design Rule	Description
Simple Interface	Users should be able to easily navigate and understand the interface
Feedback	The interface should give feedback to the user in response to actions
Error prevention	The interface should be designed in a way that minimizes the chance of user errors. Clear instructions and confirmation signals such as visual or legible feedback can be used to achieve this
Visibility	All important elements and functions of the interface should be visible to the user. Vital information should not need to be searched for

3.7.3 Human Machine Interface Specification

The specification below takes the design rules from section 3.7.2 above into consideration to describe a comprehensive list of expectations for the user interface design of the app.

- The app should have a simple and intuitive user interface with clear and concise instructions.
- The home screen should display options for capturing a new picture to identify a new species and viewing identified species.
- The identify species option should prompt the user to take a photo of the shell using their smartphone camera.
- The app should provide feedback to the user while the photo is being analyzed, such as a progress bar.
- If the app is unable to identify the species, it should provide the user with feedback, such as an identification error.
- The view identified species option should display a list of previously identified species and allow the user to view details and pictures of each captured species.

3.8 Procedural Design

3.8.1 Processing Narrative

The key steps in the process of the app are highlighted below. The steps in the process outline the minimum viable product, in other words, the basic and essential functionality of the app.

Process: Identifying shell species and viewing information about the identified species

Inputs

- User's smartphone camera
- Images of shells captured/uploaded by user
- Database containing information about the shell species

Outputs

- Identified species name and details
- Link to source of species information
- Image saved to local file

Key Steps

1. User opens app on smartphone
2. User logs in to account using
3. User captures image or uploads image of shell
4. Client sends image to the server when there is internet connection
5. Machine learning algorithm searches for species match to picture
6. If a match is found, the information is sent back to the user
7. User navigates to digital collection
8. User clicks on image to view species information
9. Blurb screen shows information about the shell including link to further information
10. If no match is found, the screen displays feedback for the user

Roles and Responsibilities

User	Responsible for taking picture and using app to identify sea shells
App	Responsible for providing understandable and usable interface in which the user can successfully upload the image that can be sent to the IBM Watson server and matched

Key Performance Indicators

- Accuracy of shell matching: the shell matching ML algorithm should be able to match shells correctly at a higher rate than the typical human can
- User acceptance and satisfaction of the user interface

Potential Risks and Challenges

- App may have difficulty identifying low quality pictures
- App may have difficulty identifying shells with obscure or rare species that are not present in the machine learning data set
- Users may provide incorrect or ambiguous photos leading to identification problems

3.8.2 Interface Description

Below is a step-by-step description of the user interface. The description highlights the sequence in which the user may use the app and refers to the wireframe screens in section 3.7.1.

1. User opens app
2. App opens to login page (Screen 1)
    a. User may login with credentials
    b. User may choose to create an account if they do not yet have one

3. Upon clicking “Create Account”, the app navigates to the “Create Account” screen where the user can enter email, username, password, and password verification.
4. Upon (1) successfully logging in or (2) successfully creating an account, the app will navigate to the “Capture Screen” (Screen 2)
    a. User can take an image using capture mode
    b. User can navigate to digital collection
    c. User can choose to upload an image from their local phone gallery
5. When a user takes an image, a popup will notify them to view match results in the gallery (Screen 4)
6. When a user chooses to upload an image, their local android gallery will open and they can choose a photo upload
7. When a user chooses to view their gallery, they app will navigate to the gallery screen (Screen 4)
8. Once in the gallery, the user will be able to view the images that have been saved
    a. User can click on image to view species information
    b. User can choose to delete photos from the gallery
9. If the user chooses to view species information, they app will navigate to the blurb screen(Screen 5) upon clicking the corresponding image
10. If the user wants to delete images from the collection, they can click “Delete Photos”, select the photos to delete, and then select the red “Delete Button”. All of this is carried out in the Gallery screen (Screen 4)
11. Back buttons are available in the top left corner to navigate to previously visited pages in the app in order to make all parts of the app accessible.

3.8.3 Design Language Description

- **Color Palette** - the color scheme of the app was chosen to be blues and yellows to resemble the colors of the beach; the purpose of the standard color palette is to make the interface aesthetically pleasing and cohesive for the user
- **Typography** – typography will be legible and easy to read for users
- **Iconography** – Icons and symbols will be distinct and easy to understand; the affordance of the page controls will be obvious
- **Layout** – layout will be clear and free of confusion to direct the user to the desired location
- **Transitions** – transitions among screens were implemented to keep the user informed of responses; call to action and other interactive elements were implemented

3.8.4 Overview of Libraries Used

To enable the functions of both the client and server sides of our application, we utilized several libraries. On the client side, we used kivy and kivymd to create the user interface, while Camera4kivy, plyer, and pyjnius helped us to access the device hardware and built-in functionalities, such as file selection. We employed the requests module to send data to the server, and apscheduler was used to ensure periodic message checking in the background of the app.

On the server side, flask was the main library utilized to create the web server, with sqlalchemy and pymysql providing secure and efficient connectivity to our database. We leveraged Tensorflow, matplotlib, and ibm_watson_machine_learning to train and host the machine learning model. Password

encryption was achieved through the use of the bcrypt and cryptography libraries. Finally, we used the json and simplejson packages to neatly bundle the retrieved data about each species.

4. Project Management

The Project group decided on general guidelines for the approach to project management of the See Shell Mobile App. Several factors were taken into consideration such as time, scope, technical experience, project management experience, and experience working on a development team. Based on the deliverables due to the primary stakeholder, Dr. Vetter, throughout the semester, it is expected that much of the earlier phases of the project will be spent on requirements gathering and documentation. Because of the nature of this project and the expectations set forth by the primary stakeholder, a project management approach that blends Agile methodologies and plan-driven development was created. It was understood that the project management approach may change throughout the project as the team members become more familiar working with each other and more familiar with the tasks at hand. The following sections highlight the approach to managing the project.

Because the team does not have experience working together, there was ambiguity in determining the project velocity and estimating time/effort for user stories. To approach this problem, the team decided to define general functions to include in the application. The most important functions and user stories were chosen first for implementation. If there was more time, more features were implemented. The goal was to ensure that time was spent developing the main functionality of the app so there was a minimum viable product (MVP) to present at the end of the project. This approach ensured that a working product was developed, with the opportunity to incrementally add valuable functionality as the team better understood its capabilities throughout the project.

4.1 Roles

The project group successfully self-organized itself into a front-end and back-end team. The groups were created based off of experience and comfortability with the tasks presented in the project. Because one team member had the initial vision for the project, she took on a project manager role. Throughout the project, a third team was created to serve as the testers. The roles for the project are as follows.

Project Manager

Responsible for communication between stakeholders and team members as well as making sure project stays within scope and timeline

- Leah Schneidereit

Front-End Team

Responsible for developing the functionality of the screens involved in the user interface of the app

- Leah Schneidereit
- Michaela Pierce

UI Designer

Responsible for designing the user interface in coherence with the design principles set forth in the project

- Alexander Johnson

Back-End Team

Responsible for integrating client-server communication, machine learning activities, and data acquisition

Responsible for integrating client and server communication, machine learning activities, and data acquisition

- Thomas Smith
- Eron Neill

Testing

Responsible for ensuring the app runs properly on a mobile phone with testing of builds and APK files

- Thomas Smith
- Leah Schneidereit

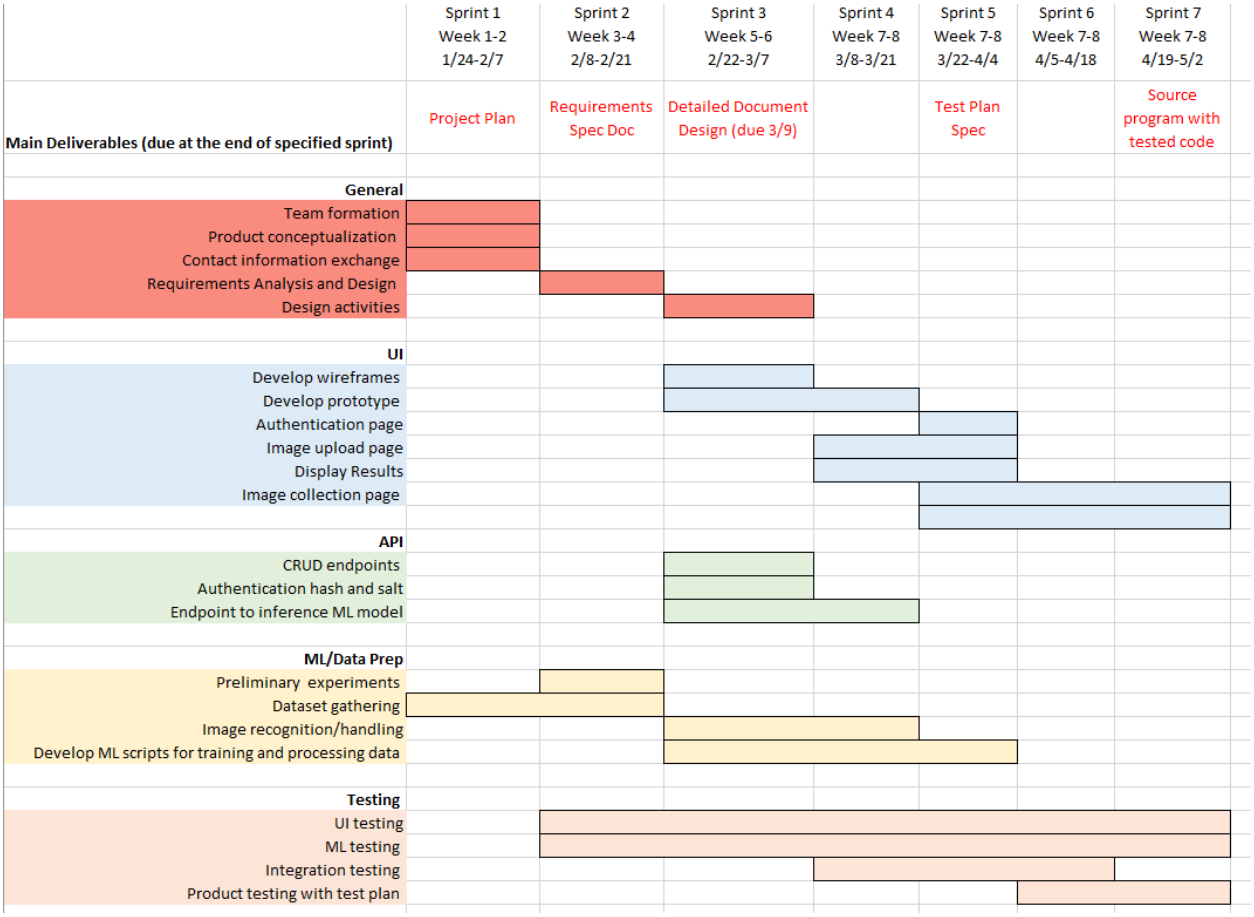
Supporting Researcher

Responsible for conducting research on IBM Watson, secure coding, and test cases.

- Ahmad Elgazar

4.2 Project Timeline

The team developed a high-level plan of the sprints over the project period with general activities expected to be performed in each sprint. The purpose of this is to eliminate the rigid nature of a Gantt chart but still provide the necessary structure that agile methodologies permit to keep a software project within scope and timeline.



4.3 Project Estimates

The most important estimate for this project was time estimations. It was difficult for the team to estimate the number of man-hours expected to be needed for the project. This challenge arises from the lack of experience working with one another as well as the varying levels of technical experience among team members. To estimate the total number of man-hours for the project, each team member decided on a best guess estimate for the number of hours. The methods of which the team members used for estimating this number valued. Some team members-based estimates off the expected hours to be worked each week while others based it off of the complexity of the project and prior experience working on similar projects. The average of each of the best guesses was calculated and used as the

overall time estimate. It is important to note that the time estimation includes meeting time per person, time spent on documentation, coding, debugging, and testing.

High estimate: 954 hours

Low estimate: 420 hours

Average project time estimate: 574 man-hours

4.4 Projects Risks

Below is a table displaying the risks associated with this project. The table assigns a severity score(out of 5 with 5 being the worst) and likelihood score to each of the risks and outlines a mitigation plan should any event occur throughout the project. The risk table helps the team consider potential roadblocks that can occur and provides guidance for overall decision-making as well as risk mitigation.

Risk	Severity	Likelihood	Mitigation Plan
Individual Sickness	2	40%	Healthy team members will cover work assigned to sick group member until they are recovered and able to contribute.
Team member dropping class	3	50%	Meetings will be held via Zoom. Tasks and deliverables assigned to the team member will be clarified.
Hardware Issues	1	10%	All documents will be backed up to OneDrive and GitHub to ensure protection and availability. Meetings and work will be conducted in CIS to ensure access to computers.
Inability to find a data set that supports the scope of the project	4	10%	Research on a data set will be performed early in the project stages to allow for any possible changes to the project scope or functionality to be made – the change in scope will allow for the use of a more available data set
Project management methodology does not work	2	30%	Adjust project methodology to better suit project/team at the first sign that the current one is not effective.
Relying on 3rd party services	4	50%	Develop secondary plan to implement features should 3rd part services become unavailable

4.5 Project Resources

Each team member leveraged his/her own technical and personal experience for the project. The team kept in close contact with the primary stakeholder, Dr. Vetter for any questions or guidance. The team also utilized any documentation for open-source libraries utilized in the development of the app. IBM

Watson is another resource that was utilized for the training of and hosting of the machine learning model.

4.5.1 Data Resources

- The images for machine learning model were obtained from the University of Macau and can be found by following this link: https://springernature.figshare.com/collections/A_shell_dataset_for_shell_features_extraction_and_recognition/4428335
- The species information was obtained from MolluscaBase and can be found following this link: https://molluscabase.org/

4.5.2 Team Member Contributions

Alexander Johnson

- Front-end experience
    - .NET and JS

Eron Neill

- Back-end experience
- Machine learning background
- API development and Data Integration
- Collaboration experience

Michaela Pierce

- Back-end experience
    - Intermediate in Python and VB
- Collaboration experience
- Leadership experience

Leah Schneidereit

- Front-end experience
- Leadership experience
- Collaboration experience
- Python programming

Thomas Smith

- Back-end experience
- Machine learning experience
- Collaboration experience

Ahmad Elgazar

- Research

- Security Testing & Code Review

4.5.3 Hardware

Each team member used their own personal device for the completion of project tasks. A dedicated Android cell phone was used for testing of the application. For machine learning tasks, Eron's desktop computer was used for training and IBM Watson Machin Learning was used to host the model.

4.5.4 Software

- GitHub: version control
- Slack: communication
- Zoom: communication/meetings
- Microsoft Office: documentation
- When2meet: team member availability
- OneDrive: filesharing
- IBM Watson: machine learning activities
- Android Studio: Android app testing and emulator

4.6 Product Backlog

The product backlog contains a list of user stories that were defined and used throughout the duration of the project. At the beginning of the project, user stories were prioritized to ensure that a minimum viable product was developed by the deadline. Tasks were assigned based on the experience of the team members so the various skills of the team could be leveraged for success. The product backlog contains split user stories into "To Do", "In Progress", and "Done" statuses. The backlog depicted in this table is the current state of our backlog. The user stories that are "Done" are those that were necessary to develop an MVP. The "To Do" and "In Progress" are those of lower priority and what the team deemed to be Future Work for later releases.

To Do	In Progress	Done
As a user, I want to upload an image for processing even when I don't have service so that my image is not lost and can be matched when I have service	As a user, I want the application to recognize the correct species based on shell images	As a user, I want to create an account so that my information is saved and I can access it on another device
As a user, I want the image recognition capabilities to improve over time so that I have a better experience		As a user, I want to upload an image of a sea shell so I can learn about the species
As a user, I want to be able to change my password if I forget it so I can maintain access to my account		As a user, I want to be provided useful information about species after they are identified so that I can learn about animals
As a user, I want to have access to my library when offline so that I can view them without internet service		As a user, I want to know when I am using the app wrong so that I don't waste time or get confused

As a user, I want to add a title/location to my images so that I can remember where I found the shell		As a user, I want to be able to delete an image from my digital collection so that I can personalize my collection

4.7 Future Work

There are several features that we plan to add to our application in the future. These features are ranked in order of their importance and are listed below:

- **Account Management:** The first feature that we plan to implement is account management. This will include password management and account deletion capabilities, which will allow users to maintain the security of their accounts and protect their data.
- **Offline Access:** The next important feature we plan to add is offline access to the application. This will allow users to access the application and use its functionalities without being connected to the internet.
- **Location Visualization:** We plan to incorporate more information into our application, specifically location visualization in the form of maps, to give users a better understanding of the locations where different species of plants and animals can be found. This feature has been implemented on the client side, but as of yet has not been implemented on the server side.
- **Image Cropping on Upload:** The machine learning model is trained on images with a 4:3 ratio, which means that images need to be this ratio for the classifier to work. As of right now, the application automatically crops any uploaded image to the largest 4:3 area in the center of the image, but a user cropping tool would likely be preferable.
- **Image Gallery Management:** We also plan to add image gallery management capabilities, which will enable users to title and favorite their images. This feature will help users to organize their image collections and more easily find their favorite pictures.
- **User Images in Machine Learning Model:** Finally, we plan to incorporate user images into our machine learning model to improve its capabilities. This feature will allow us to enhance our model by using real-world data and improve its accuracy when identifying and classifying different species.

These features will be implemented in future releases of our application, as we continue to improve and enhance its capabilities.

5. Testing

During the development process, we conducted extensive testing to ensure that our application was functioning as expected. Our testing approach consisted of two main methods: continuous integration testing during development and testing on physical devices.

Continuous Integration Testing: We conducted continuous integration testing during the development process to ensure that each feature of the application was working as expected. This helped us to catch and fix issues early in the development process, before they could cause more significant problems, and ensure that all newly generated code did not interfere with the existing codebase.

Testing on Physical Devices: After we had completed the initial development phase, we built the application into an executable and tested it on physical devices. This helped us to identify any issues that might arise when the application is run on different devices and operating systems. We also tested the application on different versions of the operating system using the emulator in Android Studio to ensure that it was compatible with all major versions. Below is a list of the major bugs we encountered when testing on the Android Device:

- Camera compatibility
- File access permissions
- UI layout inconsistencies
- Camera permissions
- Connectivity issues

While the team is aware that more testing is necessary, especially as additional features are integrated, the time constraints of the project required prioritization of testing methods. Thus, testing the application on a physical device was deemed the most critical form of testing. Although it is not a comprehensive testing solution, testing on a physical device provides an opportunity to evaluate the application's functionality in a real-world environment, where a range of variables can impact the application's performance. Additionally, conducting testing on a physical device allows for the identification of any hardware-specific issues that might not be captured in other forms of testing.

APPENDIX

Activity Log

The team met once weekly throughout the duration of the project. The purpose of this meeting was to touch base on individual progress, work on documentation, and assign tasks for the following week. These meetings were an excellent time for team members to discuss any roadblocks and work together during the development phase
The team scheduled other meetings as needed throughout the project, either with entire team or subgroups

Meeting Time: Tuesdays 6:15pm, ~1hr

Weekly Meeting Notes

Meeting 3 – 2/14/2023 6:06 PM – 7:20 PM

- Divided tasks among members for Deliverable #2 (Software plan not relevant to our actual group)
- Assign task to Ahmed per Dr.Vetter's recommendation – Learn everything about IBM Watson for the next 2 weeks and when he comes back, he can tell us everything he knows about IBM Watson
- Discussed User requirements, System requirements, and Diagrams
- Trello Workspace created for assigning tasks
- User Requirements – Front End / System Requirements – Back End
- Discussed priority of user stories
- Users can upload something from their phone gallery and the application will add file name to the image and send it to API after they have uploaded the image.
- Already have a shell image on phone that was taken outside of our app should be considered
- Possible option: Give user option to save image // Provide a delete option for user
- Should the image be deleted from our database or added to our training data?
- Discussed performance recommendation for non-functional requirement for user, to use the image
- System non-functional requirement, users can take image without connectivity and upload later due to bad cellular service on beach

Meeting 4 – 2/21/2023 6:03 – 7:26

- Worked on software requirements document
    - Scope is intended to produce a minimum viable product
    - Error handling of app briefly discussed
    - Worked on requirements and user stories
        - Imported user stories to Trello

Meeting 5 - 2/28/23 6pm - 640pm

- Created design doc
- Each person should choose some tasks from the doc and inform teammates in slack
- Front end people: begin researching Android mobile app programming
- Deliverable for next meeting: mobile app that allows users to log in and goes to next page that says "hello world"

- Begin thinking about class/methods that will be implemented

Meeting 6– 3/14/23

- Talked about assigning someone to create the basic test plan for deliverable 4(maybe Ahmad)
- Talked about working on getting access to IBM Watson
- Worked on git connections
- Worked on app interface and utility

Meeting 7- 3/16 - 5pm-6:15

- Adobe XD for visualizing interface/prototype
- Alexander will work on UI design stuff
- Michaela will work on digital collection page
- Leah will work on create account page
- Worked on some technical aspects

Meeting 8 - 3/21 - 6pm –6:45

- Discussed progress with dr vetter
- Discussed individual progress on tasks
- Discussed tasks for next week
- Leah and michaela – front end programming
- Alexander – front end UI for login page
- Thomas – API validation
- Eron – API

Meeting 9 - 3/31 - 5pm – 6:15pm

- Discussed IBM Watson
  - Currently in the process of learning what it is capable of
  - Some work may need to be done on our end to format the picture for Watson to process
- Discussed the test plan
  - Work on polishing it up
  - We may need a few test cases regarding connection failure, camera failure, etc. to cover incidents so that the app continues to function without crashing despite certain failures.
  - Keep track of the errors you encounter and the solutions you find and document them, they are going to be a part of the final document
- Discussed new pushes to the repo
  - The capture screen has been pushed and requires minor error handling.
- Ahmad is here!

Meeting 10 - 4/4

- No meeting this week due to Easter Holiday
- Team members worked individually on development tasks

Meeting 11 – 4/11

- Team worked together on technical roadblocks
- UI needs to be integrating with current version of code base
- Thomas working with Buildozer to begin building APKs

Meeting 12 - 4/18

- Working together on technical bugs
- Testing
- Restructured code base

Meeting 13 – 4/25

- Final touches
- Final touches on presentation
- Prioritize final changes for the demo
- Leah and Thomas need to work on testing
- List of final changes developed:
    - Polish UI
    - Allow file permissions
    - Proper information sent from server for Blurb screen

Individual Time Contributions

Below is a table displaying the time contributions that team members made to the project. Team members logged the hours spent on the project including meetings, programming, documentation, etc.

The original time estimation for the project was 574 man-hours. The actual number of hours reveals that the team overestimated the time necessary to complete the project by about 22%.

Team Member	**# hours**
Leah	110
Alexander	48
Thomas Smith	120
Eron Neill	75
Michaela Pierce	100
Ahmad Elgazar	15
Total Hours	468

Project Repository

The SeeShell repository resides on Github. The repository can be found at the following link: https://github.com/eyneill777/SeeShell.git

The code documentation can be found within the Github repository in the Documentation folder, at the following link: https://github.com/eyneill777/SeeShell/tree/main/Documentation