ELEC5619 - A2 - Project Report

Dex - Gamifying Ecology

15/10/2024

Daniel Chorev - 510502137

Zeeshan Ansari - 510370813

William Walker - 520659025

Nakul Reddy - 500066919

Syed Hamza Kaliyadan - 500585454

Group: M_9_GRP3

Table of Contents

1. Introduction

1.1 Background

Plants play an important role in maintaining the land health, preventing erosion, providing a habitat to animals and many other benefits [1]. Of the 21000 species that occur in Australia, 1385 are listed as threatened [2]. Additionally, about 27% of plant species in Europe are classified to be at risk of extinction [3]. This has led to a great increase in conservation efforts, especially from governmental organisations, with over A$ 7 Billion being spent by the Australian government in just the last 40 years [4].

Many of these threatened species are also classified as ‘data-deficient’, indicating a crucial lack of knowledge about their distribution [5]. Half of these data-deficient species are further predicted to be threatened by extinction. [6]

As such, the idea proposed focuses on tackling these issues directly by letting regular people contribute data to fill in the existing gaps in knowledge. Existing solutions to this problem, such as iNaturalist, have a similar model of letting people record and share their observations, however these solutions are primarily used by professionals [7].

The solution proposed here is geared towards the general populace, aiming for a broader solution that also addresses education systems neglecting plant education at both the school and university levels [8].

Furthermore, in order to make this process more engaging for the users, a ‘PokeDex’ style of application is proposed, capitalising on the growing number of studies showing the positive effects of ‘gamification’ in education [9].

1.2 Overview

Globally, our natural environment is increasingly being damaged. Our ecosystems are being harmed, and more species of flora are becoming endangered and difficult to track. There is a pressing need to involve the community in protecting our environment through localised data collection of flora. This will help us better understand endangered species, their locations, and how we can protect them.

1.3 Aims

This project aims to develop a gamified, community-based flora sighting application to collect geo-location-based data for environmental protection initiatives. By leveraging gamification inspired by Pokemon, Pokemon Go, and the Pokedex, the application seeks to:

- Entice and engage users to contribute to environmental conservation efforts.

- Increase the volume of data collected on endangered flora species.
- Provide educational information about various plant species to users.
- Foster a community dedicated to protecting the natural environment.

1.4 Project Scope

The scope of this project includes the development of a mobile application with the following features:

- **Flora Sighting Capture:** Users can use their mobile phone cameras to capture and record sightings of different flora in their environments.
- **AI Integration:** The application will use AI to identify the captured images, provide information about the species, and generate a make-believe 'Pokémon' resembling the species.
- **User Profiles:** Sightings are recorded against user profiles, allowing users to track their contributions.
- **Leaderboard:** A leaderboard feature will display user activity compared to others in the community, promoting friendly competition.
- **'Pokedex' View:** Users will have access to a 'Pokedex'-like interface showcasing all their recorded sightings.
- **Gamification Elements:** Incorporation of game mechanics to increase user engagement and motivation to contribute regularly.

1.5 Project Scenarios

Real-world scenarios where this application will have an impact include:

- **Environmental Education**: Students and educators can use the app as a learning tool to study local flora.
- **Citizen Science Initiatives**: Environmental organisations can leverage the collected data for research and conservation efforts.
- **Community Engagement**: Local communities become more aware and involved in protecting their natural surroundings.
- **Data Collection Enhancement:** The gamification elements encourage more frequent and widespread data collection, filling gaps in existing environmental data.

1.6 Audiences and Primary Users

The primary users and target audience for this application are:

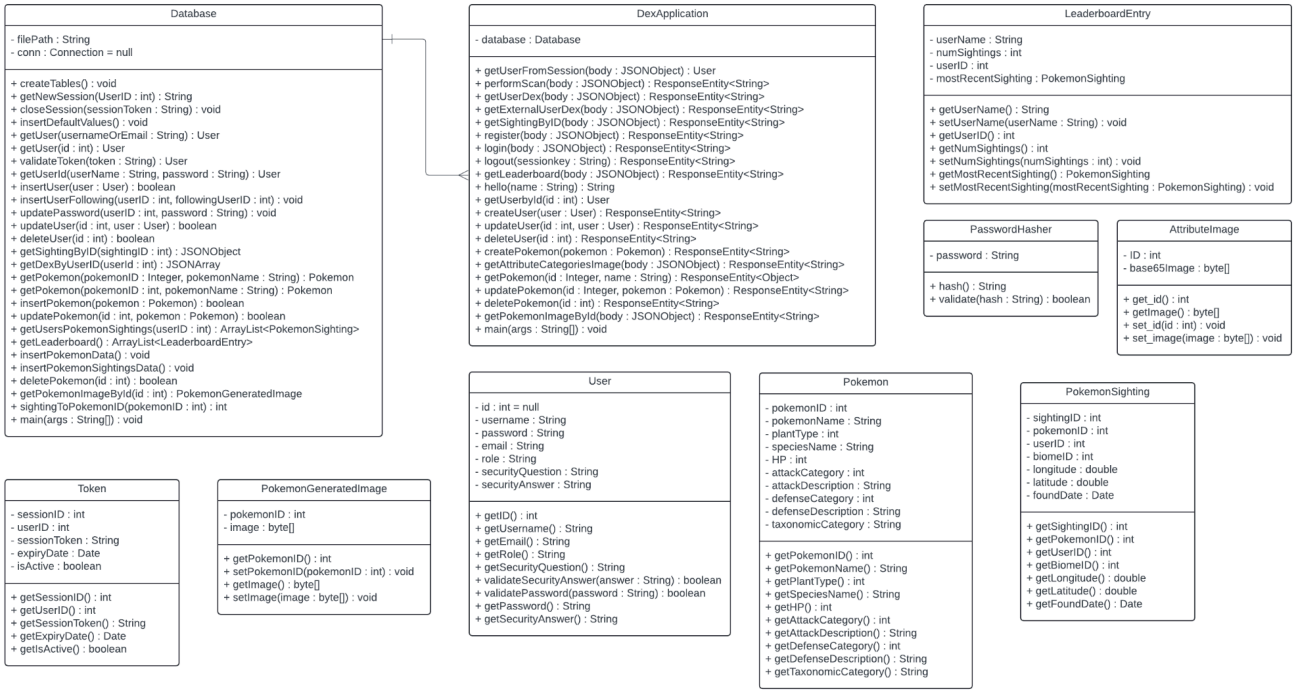
- **Nature Enthusiasts:** Individuals interested in exploring and learning about flora.
- **Students and Educators:** Schools and universities focusing on environmental science and biology.
- **Gamers:** People who enjoy augmented reality and gamified experiences similar to Pokemon Go.

- **Environmental Organisations**: Groups that can use the data for conservation planning and action.
- **General Public:** Community members who may not have prior interest but are drawn in by the game's engaging aspects.

By appealing to a wide range of users, the application aims to build a strong, engaged community that contributes valuable data for environmental conservation.

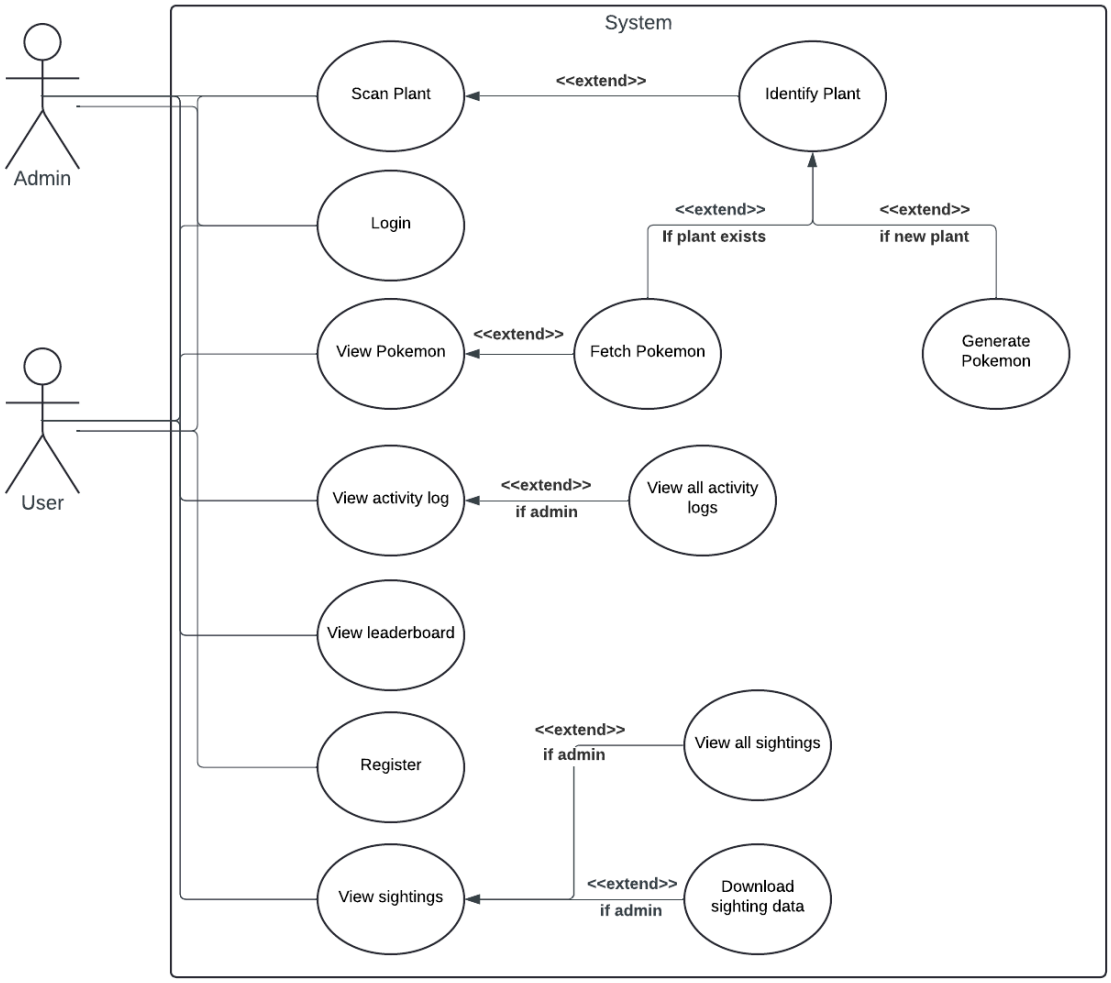
2. Project Architecture

2.1 Class Diagram



The class UML diagram above describes all of the classes used in the backend of the system and describes the attributes and methods for each class. The core DexApplication class acts as the spring boot controller for all of the API endpoints and handles the functionality for the backend application.

2.2 Use case Diagram



The use case diagram above describes the core functionality of the application with two user roles specified for admin and regular user. An admin has a management view of the application and is able to see the activity log of all users and access a full list of pokemon within the system dex rather than just a single user's dex.

3. Functionalities

3.1 Project functionalities

3.1.1 Sign Up System

Users can create a new account within the system, which employs procedures to ensure that each account has a unique username and password. Additionally, password requirements are enforced on both the backend and frontend to protect user data integrity and anonymity.

3.1.2 Login System

Users can log in with their account credentials. Upon successful login, a token is issued to the user, enabling ongoing authentication. This approach balances usability and security, as the token allows users to disengage and seamlessly resume activity with minimal disruption.

3.1.3 Upload New Entry and Generating Pokemon Data

The primary feature of the application allows users to upload an image containing a plant, where its species is identified via a third-party API. This information is passed to ChatGPT, which generates descriptive details in a style similar to a Pokémon card, paying homage to the plant's species and characteristics. After uploading, the newly generated Pokémon is added to the user's personal Dex. This 'gamifies' ecological reporting and engages the users to contribute to the data collection of plant species in their area, thus assisting the data collection and environmental reporting goals of the application.

3.1.5 Generate Real-world Data

Additionally, when a new plant entry is submitted, the AI API is used to identify the real-world qualities of the plant to enable valuable, non-fictional data collection and education for users.

3.1.6 View Dex

Users have the ability to see a view of their entire Dex, showing all of the plants they have personally identified and sightings that they have logged in the application. This contributes to the gamification experience by allowing the collection and display of a user's achievements, thus enhancing user engagement.

3.1.7 View Card Entry

Users also have the ability to view a specific entry from their Dex as a 'Pokemon card'. The screen will display the pokemon's fictional generated qualities and stats in a stylised format similar to a Pokemon Card. Again, this contributes to the gamification and engagement experience.

3.1.8 Generate Pokemon Image

After submitting a new plant image to the system, a pokemon-style character image will be generated using generative AI APIs. This becomes the icon and image for the generated fictional plant pokemon character. This image is visible with the pokemon's profile.

3.1.9 Leaderboard of users

Users are able to see a ranked list of other users on the platform. Their ranking is determined by the number of submitted plant sightings and size of their dex. Users can click on and view other users' Dexs' from the leaderboard links.

3.1.10 Export location based sightings data

Users are able to download a JSON formatted list of sightings data based on their map view. This differs slightly from the map selection functionality described in the proposal but offers the same data export ability.

3.1.11 Generate card printout

Users can generate a pokemon 'card' printout for any pokemon in their or another user's Dex and save it as an image to their device. The card will display the stats and abilities of a pokemon as well as their image.

3.1.12 View other users Dex

Users can access other user's Dex and view all pokemon that they have added via sightings. They can compare and analyse pokemon sightings amongst each other. Other user's can be found through the leaderboard screen.

3.1.13 Map view of entries

Users can see all of their plant sightings on a map view through the Google Maps API. All of their sightings will be displayed as red pins marked on the map. Admin users can view all sightings in the same map view.

3.2 Non-functional requirements achieved

3.2.1 Strong Login Security

The application utilises strong login security by implementing a few robust security measures. The application does not store plaintext passwords and instead uses strong Bcrypt hashing function for password storage including salting of passwords. Additionally, user accounts are locked after numerous failed login attempts to prevent brute-force attacks on login security.

3.2.2 Fast Screen Performance

The application provides a seamless user experience and ensures that all screens load in under 2 seconds under typical usage conditions. For longer loading scenarios such as AI API calls, a user-friendly loading bar is responsive and informative, allowing for seamless screen performance.

3.2.3 High Availability and Robust Uptime

The application would be highly available in a deployment scenario, deploying across various regions and numerous server instances. The database would be deployed with read replicas to ensure uptime and failsafes in case of hardware failures.

3.2.4 Fast AI Data Generation.

AI API calls are optimised and efficient and the application performs all AI generation tasks in under 10 seconds with typical usage.

3.2.5 Aesthetic and Intuitive User Interface

The user interface has a thoughtful, simple and consistent design with a retro feel which is easy to understand and navigate for first time users. The design incorporates aesthetic colour combinations and iconography to seamlessly direct users to their desired functionality.

3.2.6 Responsive Design

The application is responsive to many different screen sizes and device types. Optimising for tablet and small laptop use.

4. APIs and Spring

4.1 External APIs and AI API

In order to achieve the goals of this project—which included creating visual material, integrating AI-driven identification, and turning plant species data into a gamified experience—external APIs were essential. Together, the various APIs we used added to the application's functionality, resulting in a seamless, interesting, and instructive user experience. The goal, capabilities, and significance of each external and AI API utilised are described here.

4.1.1 PlantNet API

The early process of identifying plant species from user-uploaded pictures relied heavily on the PlantNet API [10]. This API's main objective was to correctly identify the plant species from the given photo. After analysing the plant's features, the PlantNet API returned a SpeciesName, which we used as the basis for the rest of our application's actions. This feature was essential because it transformed plant photos into recognisable data points, giving the gamified experience a grounded, real-world basis. We added an educational element to the experience by incorporating the PlantNet API, which allowed visitors to interact with nature through technology while learning about different plant species. Because of this API's ability to produce precise plant identifications, we were able to combine real-world information with made-up characteristics to turn these plants into characters akin to Pokémon.

4.1.2 Google Cloud AI API (Gemini 1.5)

Following the identification of a plant species, Pokémon-inspired features were generated using the Google Cloud AI API (Gemini 1.5) and the plant's SpeciesName, which was obtained from PlantNet [11]. Gemini 1.5 was designed to creatively build on the plant by generating Pokémon-like data properties, rather than to recognise the plant itself. This includes attributes that provided the plant with a fictional identity, like PokemonName, PlantType, HP, AttackCategory, and others. We could easily incorporate creative data that enhanced the practical knowledge offered by PlantNet by utilising Gemini1.5's sophisticated AI capabilities. Because it turned a straightforward plant sighting into a distinctive, gamified experience, this API significantly increased user engagement. In addition to learning about plants, users may come across Pokémon-like characters with a variety of abilities and traits.

4.1.3 OpenAI DALL-E API

By producing distinct pixel art representations based on the plant photos and Pokémon traits, the OpenAI DALL-E API was the last component needed to create a fully realised Pokémon-inspired experience [12]. The idea was brought to life by using DALL-E to convert the plant species and produce Pokémon traits into a visual Pokémon character. DALL-E was given instructions to build a pixel art Pokémon that reflected the colours, forms, and general essence of the plant while retaining a whimsical, magical nature appropriate for the Pokémon universe. The prompt was created using the plant's SpeciesName and data properties. The DALL-E API's capabilities included producing pixel art that focused on distinctive qualities like floral features and vivid colours rather than exact similarity to actual creatures. By giving users a tangible outcome that blended fictitious and real-world components, DALL-E dramatically increased the project's engagement factor and demonstrated the innovative potential of AI in educational applications.

4.1.4 Google Maps API

The application's ability to visualise the locations of plant sightings was greatly aided by the Google Maps API [13]. This API was designed to give users the ability to monitor their

Pokémon-inspired discoveries worldwide by providing an interactive map that shows the geographic coordinates (latitude and longitude) of where a plant was discovered. It worked by using markers to plot position data on a map, signifying the sighting of each detected plant. By producing an immersive, real-time map interface that displayed biodiversity in various locations, this visualisation greatly improved the user experience. By connecting plant species to their natural habitats, the API strengthened the project's educational component by enabling users to experience the thrill of exploration as they tracked their Pokémon encounters across multiple locales.

4.2 Spring

4.2.1 The performScan controller processes images uploaded by users to identify the plant species. It first checks if the user is logged in by validating their session. After verifying the session, it uses the ImageProcessing class and external AI and web APIs to determine the species in the image. Next, the controller extracts GPS data from the image's metadata to record the location of the Pokémon sighting. Once the sighting information is processed and/or generated, the sighting is saved in the database under the logged-in user's account. If the scan or metadata extraction fails, an error message is returned.

4.2.2 The getUserDex controller allows a logged-in user to retrieve their personal Pokedex, which contains all Pokemon sightings associated with their account. It validates the user's session to ensure the request is valid, then queries the database to get the list of Pokemon the user has found. The data is returned in JSON format, to be displayed on the front end. If the session is invalid, an error message is sent back.

4.2.3 The getExternalUserDex controller lets users view another user's Pokddex by providing the other user's ID. Like getUserDex, this method first verifies that the session is valid, then gets the Pokdmon sightings associated with the specified user ID from the database. The data is returned in JSON format for display. If there is an issue, such as a missing or invalid session, or if the query ID is not provided, the controller sends an error message.

4.2.4 The getSightingByID controller allows users to get detailed information about a specific Pokemon sighting by ID. It validates the user's session, checks the provided sighting ID, and retrieves the sighting details from the database. The retrieved information is then returned in JSON format. If there's no session or if the sighting ID is missing or invalid, the controller returns an error message.

4.2.5 The register controller handles the user registration process. It accepts the username, two password entries (to confirm they match), an email address, and security question details. After ensuring that all required fields are provided and that the passwords match, it creates a new user in the database. If successful, a session token is generated and returned. If there are any issues - such as missing information, mismatched passwords, or a duplicate username - the controller sends back an error message.

4.2.6 The login controller manages the user login process. It verifies the submitted username and password against the database entries. If the credentials are correct, it generates a new session token and sets a cookie containing the session key. This cookie is then used for future requests. If the username is not found or the password is incorrect, the controller returns an error message indicating the issue.

4.2.7 The logout controller logs out the user by closing their current session. It retrieves the session key from the user's cookies and invalidates it in the database. Once the session is closed, it confirms the logout with a success message. If the session key is missing or invalid, an error message is returned.

4.2.8 The leaderboard controller retrieves the top users based on the number of sightings they have recorded. It queries the database to get the leaderboard data and returns it in JSON format. This allows the front end to display rankings of the most active users. If any issues occur during data retrieval, an error is returned.

4.2.9 The getUser controller gets user details based on the user ID. It queries the database for the user's information and returns it if found. If the user does not exist, the controller returns a null response.

4.2.10 The getPokemon controller retrieves information about a Pokemon based on its ID or name, returning the relevant Pokemon details if found. If the Pokemon is not found, it returns an error message.

4.2.16 The getPokemonImageById controller gets a Pokemon's image based on the sighting ID provided in the request. After validating the session and checking the sighting ID, the controller queries the database for the image. The image is returned in a Base64-encoded format, allowing it to be easily displayed on the front end. If the image cannot be found, an error message is returned.

5. Testing

Testing for our project involved validating key functionality and features across all the classes of our Spring Boot project utilising a combination of unit and smoke testing. The JaCoCo dependency was imported into our project to generate test coverage reports.

The classes that were tested include:

- **DexApplication**: This class contained methods for creating, updating, and deleting Pokémon and user records, alongside utility methods such as `login`, `logout`, and `performScan`.
- **User**: The `User` class was tested for functionality related to user management, including password validation and security answer checks.
- **Pokemon**: This class handled Pokémon-specific data and operations, including methods to set attributes like HP and retrieve details such as Pokémon ID.

- **Token**: Token management and session validation methods were tested to ensure security tokens were handled correctly.
- **LeaderboardEntry**: This class managed leaderboard data and interactions, which included updating and retrieving the most recent sightings for users.
- **PokemonSighting**: This class contained methods to manage Pokémon sighting data, such as geographic location and sighting date.

Methodology:

Our methodology involved a combination of unit testing and smoke testing. Unit testing mainly targeted methods with specific inputs and verified them against a particular set of expected outputs, while smoke testing was used to ensure that methods were executed quickly and ran without errors. However this meant that detailed validations could not be involved in the testing process.

Challenges:

Despite our best efforts, certain code paths and methods could not be fully tested due to the following reasons:

- **Database Interactions**: Several methods in our classes are tightly coupled with interactions with a live database (e.g., `createPokemon`, `deleteUser`, `getPokemon`). To test this functionality effectively, would require the setup of a comprehensive test environment with mock data. Although smoke tests were able to execute the methods, verification of the results from database queries was deemed infeasible given our testing strategy and the scope of the project.
- **External Dependencies**: Some methods, such as `register(JSONObject)` and `performScan(JSONObject)` involve integration with external, generative AI and other services. This made it challenging to test these methods through conventional unit testing and test multiple branches. They also depend on external input. While integration testing would be more appropriate for the methods, it was beyond the scope of our project.
- **Branches Missed**: Exceptional circumstances such as failed database connections, invalid input formats, other JDBC exceptions were not completely tested. These branches require mocking exceptions and the generating erroneous data. This was beyond the scope of our test setup.
- **Complexity of Conditionals**: Certain methods contain complex conditionals that could not be fully tested due to time constraints and the need for multiple data permutations. For example, methods like `getUserDex(JSONObject)` and `login(JSONObject)` have several branching paths that depend on the specific state of the database, making comprehensive testing difficult.

Testing Results:

Our testing efforts achieved the following coverage:

- **Instruction Coverage**: 73%
- **Branch Coverage**: 40%
- **Line Coverage**: 73%
- **Method Coverage**: 102 methods tested across various classes.

While some edge cases and error scenarios remain untested, the core functionality of the application has been verified to work as expected. The test reports from Jacoco can be found in the latest commit in the `develop-nakul` branch for reference.

6. References

[1] Author, "About native vegetation," *NSW Environment and Heritage*, Jun. 14, 2022. https://www.environment.nsw.gov.au/topics/animals-and-plants/native-vegetation/about-native-vegetation

[2]"Flora and fauna | Australia state of the environment 2021." https://soe.dcceew.gov.au/biodiversity/environment/flora-and-fauna#plants

[3] P. Weston, "Number of species at risk of extinction doubles to 2 million, says study," *The Guardian*, Nov. 10, 2023. [Online]. Available: https://www.theguardian.com/environment/2023/nov/08/species-at-risk-extinction-doubles-to-2-million-aoe

[4] L. Broadhurst and D. Coates, "Plant conservation in Australia: Current directions and future challenges," *Plant Diversity*, vol. 39, no. 6, pp. 348–356, Dec. 2017, doi: 10.1016/j.pld.2017.09.005. https://www.sciencedirect.com/science/article/pii/S2468265917300781

[5] "Data-deficient species," *NSW Environment and Heritage*, Jan. 22, 2024. https://www.environment.nsw.gov.au/topics/animals-and-plants/threatened-species/saving-our-species-program/threatened-species-conservation/data-deficient-species

[6] J. Borgelt, M. Dorber, M. A. Høiberg, and F. Verones, "More than half of data deficient species predicted to be threatened by extinction," *Communications Biology*, vol. 5, no. 1, Aug. 2022, doi: 10.1038/s42003-022-03638-9. https://www.nature.com/articles/s42003-022-03638-9

[7] "INaturalist," *iNaturalist*. https://www.inaturalist.org/

[8] D. Lewis, “Education system ‘neglecting the importance of plants,’” *University of Leeds*, Jul. 11, 2022. https://www.leeds.ac.uk/news-science/news/article/5120/education-system-neglecting-the-importance-of-plants

[9] Department of Education, “Gamification,” *Department of Education*, May 02, 2022. https://www.education.gov.au/australian-curriculum/national-stem-education-resources-toolkit/i-want-know-about-stem-education/different-kinds-stem-education-initiatives/gamification

[10] “Identify, explore and share your observations of wild plants.” Available: https://identify.plantnet.org/en-au

[11] “Generate content with the Gemini API,” *Google Cloud*. Available: https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference

[12] “OpenAI Platform.” Available: https://platform.openai.com/docs/guides/images

[13] “Add a Google map to a React app | Maps JavaScript API | Google for Developers,” *Google for Developers*. Available: https://developers.google.com/codelabs/maps-platform/maps-platform-101-react-js#0