Opikoodia project goals

# What is required?

* Application
* a public GitHub repository
* a simple manual
* a simple software design document that includes:
  + architecture design
  + class design
  + database design
  + UI mocks
  + static and dynamic flow design (UML: use cases, class diagrams, sequence diagrams)

# overview of the application

This application is a simple forum application, where users can post images taken in nature and ask other users to recognize species of fauna, insects or animals shown in the images. Other users will be able to comment on the posts, sharing their knowledge of the species. Comments can be rated by all users, based on how useful or truthful they are. A reaction can be added to posts, indicating the reacting user also didn’t recognize the species. This will boost the post’s visibility on other users’ feeds. Posts will be displayed based on which have most reactions. Once the user that owns the post has received an answer that they are happy with, they will be able to (and prompted to) mark the post as resolved, giving room for other posts. Resolved posts can be view on another feed. Comments are disabled on archived posts.

Posts will have tags, such as #insects, #animals, #trees and they can be used to filter the posts seen on a user’s main feed. For example, a user can filter their feed with #trees to only view posts about trees.

This application requires an account and cannot be viewed without logging in first. A log in page will be prompted as the application starts. Users have access to their personal profile page where they will see their username and a profile picture. Both can be edited through the profile page view.

This application is made for people who enjoy nature and photography. It provides users with less knowledge of nature (or just a certain species) a place to find answers and learn, but also presents users with more knowledge a place to share their wisdom. And to all users a place to enjoy images of nature!

# Technologies used / dependencies

This application is built using MERN stack. The front end of the application is built with React and the back end using Node.js. and Express. This application uses MongoDB as a database. In addition, Mongoose and Cloudinary have been used to build this application.

# Architecture design

The front end will be built with React and create-react-app. Backend is built with Express and Mongoose to set up a server and connect to the database. User information will be stored in MongoDB and http requests will be made to fetch data back to the application.

# USER INTERFACE

## Log in page

* Application logo
* A log in form
  + an email input
  + a password input
  + a submit button
  + sign up link
* A background image or gradient

## Sign up page

* application logo
* a sign-up form
  + an email input
  + a username input
  + a password input
  + a submit button
  + ‘Already have an account? Log in’ -link
* A background image or gradient

## Navigation bar

* Application logo
* notifications button
* user profile button

## Feed

* make new post -button
* filter buttons and options
* option to choose between recent posts and resolved posts
* a list of posts

What transitions occur between these states and views? And how should they be animated?

# FUNCTIONALITY

Key questions to ask in the application design document:

What does the application do, and how quickly does it do it?

What are possible failure conditions and how are they handled?

What one-time operations are done at the first execution (i.e., after installation)?

If the user creates entries of any kind (e.g., bookmarks), what are the limitations?

## Database

This application uses MongoDB as a database. User information (username, email and password) will be stored in the database. Posts and comments will also be stored in the database.

As for images, Cloudinary will be used to store them as cloud storage. Cloudinary will also be used to style the images.