Technological Feasibility Report
Team Bird's iView

October 23th, 2020

Project Sponsor: David Plemmons

Project Mentor: Sambashiva Reddy Kethireddy

Team Members:

Jonas Dunham Jordahl

Jordan Colebank

Chenhao Li

Tyler Riese

Table of Contents

1. Introduction

More than 50 million people spend more than $40 billion on equipment and travel related to bird watching every year. From a scientific perspective, being able to identify individual birds would give bird researchers a whole new dimension of data that would allow them to better study trends in different bird species. From an educational perspective, vivid close-up images of birds will only further our understanding, respect, and curiosity for them. From a human perspective, we desire, perhaps now more than ever, more ways to connect with each other, and a platform where fellow bird enthusiasts can share, learn, and grow could provide those rich human-human connections that are so vital for our wellbeing.

Our client, David Plemmons, is an avid bird watcher and enthusiast, he has been working hard over the last three years to bring his vision to fruition. His product, the pEEp Smart Feeder, aims to get people more involved with the birds around them by providing an educational platform and vibrant community for people to connect and grow with other like-minded individuals. Even though he has made significant progress towards his goal, there are still technological deficiencies which must be addressed. Currently, his platform only allows users to view and save images, but provides no information about the contents of the stream or ways to share and interact with saved images and videos. Because of this, we are confronted with numerous problems to solve. Just to name a few:

- How can we classify the species of birds in an image?
- How can we identify individual birds from each other?
- How can we organize saved images so users can share them?
- How can we allow experts to review user classified images?

To solve these problems, we will create an Artificial Intelligence (AI) module which will be integrated into a web application. Because AI has proven to be incredibly powerful, we will leverage AI technologies to deliver on the project's goal of connecting people to the birds they coexist with. We will first and foremost use AI so that we can classify the species of a bird, but our web application will also employ a database for storing and sharing images, an elegant interface for users to interact with images, and a community forum where users can share their experiences with others.

To establish the feasibility of our solution, we first address the major technological challenges of the project. Next, we analyze potential solutions for each challenge and objectively compare them with each other. After this is complete, we briefly discuss the integration of all technologies we plan to use and, lastly, provide a roadmap for the next phase of development.

2. Technological Challenges

To deliver a product capable of handling the deficiencies in the current version of the pEEp Smart Feeder, there are numerous technological challenges that must be addressed before development. Our project requires a trained AI model to interact with a web application, so we must first address the challenges of building a website to fit the project needs, and then address the challenges of integrating an AI model into the website. Given these considerations, our technological challenges may be grouped into the following categories:

2.1. AI Challenges

- We will need a Machine Learning (hereafter referred to as ML) framework that provides good support for the ubiquitous algorithms used in today's ML applications.
- We will need an ML framework that has a fast learning curve so that we may focus on the high-level demands of the model rather than its low-level implementation.
- We will need an ML framework that has an abundance of educational material to further expedite learning.
- We will need an ML framework that is open source (and hence free to use), and has a strong base of developers devoted to maintaining it.

2.2. Backend Challenges

- We will need a back-end framework that has sufficient compatibility with our front-end framework, AI training and our database.
- We will need a back-end framework that is fast enough to download and upload files.
- We need a backend framework that is secure enough that our customers' information will not be compromised.
- We will need an additional server to append the saved images to our training set, and periodically retrain the model based on the expanded training data.
- We will need this server to be able to redeploy and update the model being used in production.

2.3. Frontend Challenges

- We will need a frontend framework that has additional components and libraries to ensure efficiency in development.
- We will need a frontend framework that is quick and easy to use.
- We will need a framework that allows for browser compatibility so any user can access our web application.
- We will need a secure interface that can handle XSS attacks to guarantee user data is secure.

2.4. Database Challenges

- We will need a reliable database store where users can save settings, images, etc. to their profile, which also allows the sharing of user content with other users.
- We will need a reliable database store where forum threads are stored for others to view and comment on.
- We will need a secure database that will keep our user's data private.
- We will need a database that can scale on a local level while still maintaining the integrity of our data.

3. Technological Analysis

3.1. Section Overview

The major technological design decisions that we have identified as being critical to the success of our project, and which stem directly from our research of the technological challenges have derive, can be grouped into the following four categories: AI Training and Deployment, Backend Infrastructure, Frontend Interface, and Database Management.

We begin our investigation of technological feasibility with the AI module. The AI challenges we have identified in the previous section can be further classified into two groups: one pertaining to the actual developing experience when using the framework, and the other pertaining to the features we must provide by leveraging the framework. Regarding the developing experience, we should be able to quickly learn the basics, implement rapid prototypes, and be able to rely on the vitality of the framework. Regarding the features we must provide, our framework should have excellent support for developing CNNs, as well as excellent support for all other required algorithms that are integral in contemporary learning applications.

We then move our focus to the backend framework that must interact with all other aspects of the system. For the backend, the challenges we have identified are closely related to compatibility with both the AI module and the database system. Because the backend is involved in nearly all interactions between system components, it will be highly advantageous for significant aspects of the system to retain some level of homogeneity, which could be achieved by using the same programming language for the AI module and the backend. In addition, we must also ensure clean database integration so that users can be served backend information (forums, profile settings, etc.) in a reasonable time and in a consistent manner.

To tie the backend together, we must also ensure an intuitive user interface that delivers our core functionalities to our users. In order to do this, we need to be able to rapidly cycle through prototypes in order to derive a simple interface that is pleasing and suitable for users of all ages. In order to do this, we must utilize a frontend development framework that is efficient to learn, provides solid security measures, and has total support for most browsers.

Lastly, we need a database so that the web application can deliver on our goal to provide a vibrant community for bird enthusiasts. When considering different database frameworks, we will utilize the CAP theorem (described below), and also consider the scalability, speed, and security measures that each framework offers.

3.2. AI Training and Deployment

Intro the Issue

The main functionality around which our project is centered is a robust AI model capable of accurately predicting a minimum of three (3) bird species. Given that the fields of object detection and image classification have been revolutionized by the CNN, we will need a library that provides a framework to train our own CNN, analyze its performance, tune its parameters, and then deploy the trained model for use by our web application.

Desired Characteristics

Training a highly customizable and flexible CNN requires an understanding of the maths involved, but the extent of this understanding varies from framework to framework. There are so-called “no code” frameworks on one extreme (designed for use by non-technical users), and total end-to-end implementations on the other extreme (i.e., writing all functionalities yourself instead of using an existing framework). For our project, we are looking for a framework that falls in the middle of these two extremes so that we may have sufficient flexibility in tailoring the training pipeline to our own needs while also leveraging the power and convenience of pre existing libraries.

Since the algorithms used for training and evaluating a model are relatively ubiquitous, there is little use in writing our own support libraries from scratch, as doing so would require a large amount of time and energy that would detract from the overall aim of the project. This brings us to the first, and most important, characteristic we require in an AI framework: the ability to transform (cleaned) data into a deployable model with the use of core framework functionalities in each major step of the training pipeline. This means that setup, initialization, training, and deployment (of a CNN, in our case) are each executed with just a handful of function calls, while still remaining flexible enough to fit our specific needs.

Now that we know the core functionality we require, the next characteristic we desire in a framework is the ability for our team to develop rapid prototypes and quickly modify the existing training pipeline. This is critical because, in these early stages of development, we need the ability to iteratively update our training pipeline with minimal cost, as we will also be developing the frontend, backend, and database system at the same time. If we were to use a framework designed for novel research, for example, then we would instantly require a deeper understanding of ML fundamentals to realize the benefits of that specific framework, which would, again, detract from the overall aim of the project.

Another important characteristic concerns the amount of educational material present for any given framework. As is with most sub-fields of Computer Science (but in ML especially), the math involved is serious and constantly evolving. Because of the high potential for mathematical mixups during implementation, it would be highly beneficial to have access to quality tutorials and example programs that showcase the correct use of the algorithms we will be utilizing. If we consider this in our analysis, then hopefully we will minimize the amount of time we spend fixing bugs related to our misunderstanding of the core mathematics.

The last important characteristic that we will consider here is the openness of the community for a specific framework, as well as the size of the community using it. Clearly, just because X% of people use Y framework doesn't mean that Y is perfect for our application, but it can give us a general idea of the frameworks' ability to be flexible and easy to utilize. On top of this, it is absolutely necessary that the developers of a given framework are dedicated to fixing bugs and improving the functionality as they (and the community) see fit. While most popular frameworks get full marks on this aspect, it must explicitly be considered as to avoid future maintenance issues.

Alternatives

Tensorflow

When doing anything related to AI, one will invariably come across Tensorflow. Tensorflow is near the top of every list comparing machine learning frameworks. This framework is one of the most popular frameworks to date, and provides the foundation for incredibly powerful apps such as Google Translate. It was originally created by Google Brain for internal use, but openly released under the Apache license in 2015.

Pytorch

Pytorch is the main competitor of Tensorflow. Developed by FAIR (Facebook AI Research), Pytorch is the Python implementation of Torch, and was initially created by Adam Paszke, and excels in use cases related to computer vision and natural language processing. It has been open-sourced under the modified BSD license, and provides the foundation for the equally powerful and impressive Tesla Autopilot.

Keras

Keras was developed by Francois Chollet around the public release of Tensorflow. Tensorflow and Pytorch can be seen as "lower level" frameworks - they provide the necessary tools to train models, but each are with their own intricacies and learning curve. Keras serves to abstract these details from the developer (while still remaining customizable), with the goal of providing a framework suitable for rapid development. Given that its core backend is Tensorflow, Keras has an equivalent level of functionality, but a less steep learning curve.

Analysis

It would be hard to overstate the impact of CNNs: in their initial appearance to the world, they were used to solve problems like handwritten number classification, but today they are used to solve problems like autonomous automobile driving and navigation. Ever since, CNNs have been an essential tool for all flavors of computer vision problems. Because of this, every framework that we are now considering has total support for the development of CNNs, and thus this aspect will not affect our overall choice of framework.

Given the research we’ve done and the timeline for our project, speed of development is one of the most critical factors we will consider when choosing an ML framework. Because machine learning is essentially a sub-field of mathematics, the specifics of implementation quickly becomes highly technical. This is great for those that want to research algorithms in depth or modify and optimize them, but not for those who simply want to reap the benefits of the technology before diving into the weeds and understanding the core mechanisms that make it possible. Since our team is in the latter group, we are looking for a high-level framework that will allow us to initially ignore low-level intricacies and focus on the more important requirements for our AI module.

Tensorflow was born in a setting where highly scalable, production-ready model development and deployment was the main goal. As such, Tensorflow is highly configurable, allowing users to tune every aspect of their training pipeline, but also very flexible as it allows for efficient deployment in any language and setting. These aspects make Tensorflow a highly adaptable framework suitable for many ML problems, while at the same time causing extra overhead for those who are just becoming familiar with AI. This is in contrast to both Pytorch and Keras, though more so to the latter. Pytorch claims to be very “Pythonic”, meaning that it is extremely idiomatic for people who are experienced with (or wish to learn) Python. This is a major advantage because, by forcing their API to match the foundational structure of the language, they reduce the potential for those who are less experienced with Python to become confused, thus reducing the learning curve. That being said, Pytorch is still considered a low-level framework in the same way Tensorflow is, as the programmer is assumed to be experienced with the mathematical primitives that it employs. This aspect of Pytorch isn’t necessarily a weakness of the framework, as these are the very things that researchers need to push the bleeding edge of ML and AI. This brings us to Keras, and how their approach is orthogonal to Tensorflow and Pytorch. The main difference between Tensorflow/Pytorch and Keras is that Keras prioritizes the developer experience, while Tensorflow and Pytorch focus more so on thriving in research and production applications. This is not to say that Tensorflow and Pytorch do not prioritize the developer experience, but rather that neither of them prioritize the developer experience *of beginners* to the extent that Keras does. Because of this, Keras is becoming the default learning framework for those new to ML and AI. Coupled with Keras’ goal for user friendliness is their goal for users to be able to rapidly build and deploy prototypes.

Because of this, Keras is the go-to framework for use in ML competitions, which proves their success with regard to both of their aforementioned goals.

In addition to speed of development, another important characteristic to consider is the amount of educational material present for any given framework. This includes good documentation, sample projects made to showcase functionalities, forums where developers can troubleshoot and/or bounce ideas off of eachother, etc. This is necessary for our team to consider, as this project will mark our first step into the field of AI and ML, and, because of this, we will surely encounter hurdles where our lack of understanding drove us to an incorrect implementation. Having excellent documentation (where assumptions are made explicit and the intended use of certain functions made clear), abundant examples of other developer's successful projects, and an outlet for us to reach out to the greater community if we encounter a never-before-seen problem are all crucial capabilities that, if lacking in a framework, could seriously hinder progress. To get one measure of the amount of educational material present for a given framework, we can simply use Google to search for tutorials for all frameworks, comparing the amount of search results: Tensorflow: ~10M results, Pytorch: ~1M results, Keras: ~30M results. In addition to this metric, we can also check for code examples (ideally, examples associated with the framework). On Keras' website, they list code examples for all flavors of ML problems. On Pytorch's website, they have warm up exercises that get developers comfortable with the core math libraries, but not extensive code examples for all varieties of ML problems like Keras does. On Tensorflow's website, they have extensive resources for tutorials, courses to build the foundational knowledge required, and even a certificate program that covers the entire API. What they lack, however, is the repository of examples that show solutions to many different problems using their framework, which will be indispensable for us as we begin prototyping.

Lastly, we consider the openness of the framework. Specifically, we want to find out if the framework is proprietary or open-source. If it's open-source, is there a strong community of developers contributing to maintaining it for the long haul? And as part of this question regarding openness, for what reasons is the framework being maintained? Is it for research, production, or learning? All frameworks considered are open-source, so all frameworks are able to be downloaded directly and modified. That being said, the context for which these frameworks were born varies, and so does the community of users. Since Tensorflow was born in a context requiring excellent scalability and configurability, it is maintained and improved mostly based on improving with respect to each of those aspects. Because it was created and is maintained by Google, the reasons for maintenance and expansion will likely be centered around performance, configurability, etc., rather than on the developer experience. Likewise with Pytorch being developed by FAIR, this framework is oriented towards turning research prototyping into production deployment. When asked why FAIR developed their own framework, the creator of Pytorch cited the fact that no existing framework did exactly what they wanted it to -- so they just built their own. This is important to understand, because some of the most brilliant AI researchers in the field built this framework for their own bleeding-edge research, and while there is some focus on developer experience, it is not as central to the framework as it is with Keras. Lastly, we consider Keras. Having already explained the main design philosophy of Keras, it is clear that the maintenance and expansion of the framework will be centered on the aspects that has made the framework so successful in the first place: their focus on providing an excellent developer experience and allowing for rapid prototyping through clever abstraction of the ML training pipeline.

	Tensorflow	Pytorch	Keras
CNN Support	XXX	XXX	XXX
Speed of Development	X	XX	XXX
Educational Material	XX	X	XXX
Openness	XX	XX	XXX

Chosen Approach

Not surprisingly, Keras performs best on the objective measures defining our most important needs in an ML framework. Keras' focus on developer experience is unparalleled in other ML frameworks, and, to a higher degree than all other frameworks we have considered, minimizes the cost required to go from minimal experience to deploying a trained ML model.

In addition to the large community of developers who support and maintain Keras, Keras also provides educational content most applicable to our needs. Since we need to be able to quickly develop test programs, the code examples on the Keras website will serve as our starting point, from which we can iteratively add features and begin to explore the full potential of the framework.

Of course, the real advantage with Keras is the ability for us to initially abstract and ignore some of the finer subtleties involved with designing and developing an end-to-end ML solution. However, we will, eventually, need to address those finer details, which is yet another advantage of Keras -- because Keras is layered on top of Tensorflow, Keras retains the hackability and configurability that makes Tensorflow so powerful, so we will not be limited with what we can change as we get deeper into development.

Coupled together, these combined features of Keras will let us quickly and efficiently design and implement our training pipeline, allowing us to forego extensive configuration at the start so we may focus on the most important details, but also permitting us to return to those details at a later point, tuning them to further improve our model.

Proving Feasibility

To prove that the Keras framework is suitable for our needs, we will put the claims of rapid development to the test by producing a series of demos that solve problems similar to those we're addressing in our project.

The first phase of the demo will merely be a proof of concept, in which we will use the Keras library and a sample data set (MNIST, most likely) to train and analyze a CNN. The demo should be easy and fast to produce, as it is essentially the "hello world" equivalent for deep learning.

The next phase will be to use a training set of images that are specific to birds and apply our learning algorithm to that data set, without making any architectural changes at first. The point of this is to test how easy it is to make adjustments to our model in Keras, and to gain some experience in tuning the model to our own needs.

The last phase of demos will involve integration of our naive bird classifier into a web application, where users can load images and use our model to make an inference about a bird's species. This series of tests and demos, if successful, will provide a complete and total proof of concept, ensuring that our core technologies are really capable of doing what we set out to do.

3.3. Backend Infrastructure

Intro the Issue

This brings up the fact that we need an environment capable of hosting our model and using it to infer bird species within our website, while also handling other requests. To do this, we will need a backend framework with good support for libraries capable of deploying a trained model. In addition to this core functionality, our backend framework should have excellent support for a database architecture capable of efficiently creating, reading, updating, and deleting user-stored content.

Desired Characteristics

For our project, one of the most important features of the back-end framework is compatibility. Our back-end framework needs to be compatible with our AI trained model. Since we will use AI learning, our backend framework should be capable of handling this. Since the back-end framework also needs to connect to the front-end framework and the database, we put compatibility in the most important position.

The second important characteristic is scalability. Customers of our project will upload all kinds of documents, such as text, pictures and videos., so we need to consider if the framework applies to larger or smaller projects. Understanding how the framework will handle larger data sets if the amount of data in the future increases, and how the framework can hold large to small amounts of data.Therefore, the backend framework we choose should be demonstrably able to handle large amounts of operations and activity.

The third characteristic we need to consider is the security. The security of our data will be particularly important as our program will help any birders using pEEp to upload their own information and photos. We want to ensure that our data is shared without causing any leakage to prevent it from being exploited by criminals. Any information stored should be completely secure. Users should not worry about any information they have on this application being compromised. Therefore, the framework we choose should provide some authentication and security plug-ins or built-in functionality.

The final characteristic we decided is speed. We need to make sure that the speed of data downloaded and page displayed is fast so that our users can have a better experience when using our application. Therefore, the framework we choose should provide high speed in download and upload the data.

Alternatives

Laravel

Laravel is a concise and elegant PHP Web development framework. It frees you from spaghetti-like clutter; it can help you build a perfect webapp, and every line of code can be concise and expressive. Laravel is one of the most popular frameworks for PHP users. Laravel is easy to understand, powerful, and provides powerful tools for developing large, robust applications. With validation, routing, Session, cache, database migration tools, unit testing and other common tools and functions.

Flask

Flask was born in 2010 and is a lightweight Web development framework written by Armin Ronacher in the Python language based on the Werkzeug toolbox. It is mainly aimed at small applications with simple requirements. Flask is a micro web development framework for Python. It is well documented, rich in plug-ins, including development servers and debuggers, and secure cookies.

Express

Express.js was founded by TJ Holowaychuk. The first release, according to Express.js's GitHub repository, was on the 22nd of May, 2010. Express is a concise and flexible framework for Node.js web applications that offer a range of powerful features to help you create a variety of web applications. Express also provides rich HTTP tools which allows you to quickly build a fully functional website.

Analysis

Compatibility is one of the most important characteristics we need to consider. Although the difference in compatibility of each framework is small, Flask has better compatibility according to our survey. Flask is a micro back-end framework in Python, and the AI training framework we will be using will also use Python as the programming language. This will increase compatibility between the two, so Flask has the best compatibility. Although Laravel and Express are both popular back-end frameworks in the market, these two backend frameworks which contain so many functions, are too large, resulting in excessive functions. Therefore, despite their good compatibility, we still decided to give up on them.

When it comes to scalability, after the research of three backend frameworks. We decided Laravel has the best scalability. Laravel focuses on modularity and extensibility. You can find any file you want to add in the Packalyst directory, which contains more than 5,500 packages. Laravel's goal is to make it possible for you to find any document you want.

Laravel will be a great way to extend our product. Customers can add whatever data they want and then manipulate it in the Laravel extension package. By default, Flask does not contain database abstraction layers, form validation, or any other nice functionality that has been handled in other libraries. Instead, Flask supports adding these functions to applications by extension. Numerous extensions provide database integration, form validation, upload processing, and a variety of open authentication technologies.

Flask may be "tiny", but it is ready to be used in a complex production environment. Express also has good scalability, in which it provides all the functionality for developing robust Web/mobile applications, as well as APIs. Developers can easily add Express capabilities by developing plug-ins and extensions for it. Here is a list of some of the basic features that Express provides:
Communication with any third-party database, variety of user authentication methods, template engines that conforms to the Express interface definition, and definition of the project directory as needed. In conclusion, the scalability of Flask is about the same level of Express.

In addition to scalability, another important characteristic that many users will care about is security. According to research, Laravel has very good security and is one of Laravel's best features. The Laravel framework provides advanced and powerful web application security that developers can safely use. Laravel security makes effective use of salt hashing and password adding mechanisms, so it does not save passwords as plain text to the user database. It also USES the "Bcrypt hash algorithm" to create an encrypted password. In addition, Laravel Web development framework utilizes SQL statements to secure SQL injections. Flask also has good security, it provides many Network security facilities. For example: Cross-site scripting (XSS) which is injecting arbitrary HTML (including JavaScript) into the environment of a web site.Cross-site request Forgery (CSRF) which can secure your website cookies and JSON security which can secure the JSON data transfer process. And Express also has the same security features like Flask, it can also defend against some XSS attacks and the protection of cookies. So we think that Flask and Express have the same level of security.

Lastly, we will consider the speed, Speed is a shortcoming of Laravel, Laravel has many functions, but according to the requirements of our project, we tend to use less than so many functions, in some specific service maybe we will only use the partial function of Laravel, but Laravel will default to load all the functions, this led to Laravel speed too slow and resource consumption, and thus in the speed field that Laravel has no advantage. Flask is a significant advantage. Flask is a micro framework developed by Python, so it's very flexible and fast.Flask's simplicity allows experienced developers to complete smaller applications in a short amount of time. And because there aren't too many plugins and functions, flasks run very fast and load quickly.Express also has good speed, with its small and flexible Node.js Web application framework and extremely fast I/O, so it has a strong edge on speed.But I still don't think he's faster than a Flask.The reason is that the JS language doesn't fit our AI training and database as well as the Python language used by Flask.

	Laravel	Flask	Express
Compatibility	XX	XXX	XX
Scalability	XXX	XX	XX
Security	XXX	XX	XXX
Speed	XX	XXX	XX

Chosen Approach

By comparing the desired characteristics of each alternative, we finally decided to choose Flask as our backend framework. Not only because it has faster speed, good security and compatibility. What we need most is the compatibility between the parts. We should know that the quality of a product depends on its whole, not on any one part.
Therefore, only better compatibility can guarantee the quality of products. But in this respect, Flask is undoubtedly superior to the other two alternatives. Because Flask uses Python as its language. While Laravel and Express use PHP and Js. But the language of most today’s AI deep learning is Python. In order to match our AI deep learning framework, we believe Flask will be a better choice. Even though Laravel has a better scalability, this will make it slower and with too many useless plugs which we will not need in our future development.

Proving Feasibility

In our demo, we'll design a basic website using Flask. We're going to create one simple connection that connects the database to our website, allowing information to be exchanged between the two. Our users will upload videos of their birds on our website and use our AI training framework to identify the species.

3.4. Frontend Interface

Intro the Issue

Lastly, the engineering done to solve the AI and backend problems must be seamlessly integrated into a clean user interface (hereafter referred to as UI). This means that the technologies used for frontend development must play nice with all technologies used for the backend. Additionally, given that the average age of birdwatchers is around 50 years old, our UI should be simple, elegant, easy to navigate, and hide any implementation details that would otherwise confuse users of our application.

Desired Characteristics

When choosing a frontend framework, one of the major characteristics to consider is efficiency in development. Frameworks in general are designed to give developers structure when creating user interfaces, but this doesn't necessarily mean that every framework allows for the proper structure. For our project, it is crucial that we have a framework that is easy to use, but also provides us with the proper tools to develop our UI. With that being said, having built in components and libraries would allow us to write our code faster and effectively. This leads to another important thing to consider which is cost. UI development can become awfully pricey without the proper tools. Being able to write code faster and more efficiently by using added components or libraries would allow for a decrease in the overall time it takes to write an intuitive UI. This could result in a decrease in the overall cost of a product.

Although a framework with additional components and libraries is important to us, we don't want something that is overly complicated. For our project, it is not a necessity to have a framework that has *thousands* of extra features. We need something that is going to ensure we will be able to develop our web application by providing us with necessary built-in components.

Additionally, a framework with a variety of features that has a slow learning curve is not ideal for our situation. Considering our time constraint for developing this web application, we need the learning curve of a framework to be relatively fast. That would guarantee we would spend less time learning how to use the framework, and more time developing our project. The framework we choose should be simple and straightforward with implementation.

Since we will be creating a UI that allows for user data to be submitted (ie. user login, user photos), security is another characteristic to consider. Although it is often overlooked, frontend security is equally as important as back end security. Even though the data users are inputting goes directly to the backend infrastructure of our application, if a perpetrator inputs malicious code into our program, they can still gain access to a user's information once the code has been executed (Singh 2019). Therefore, we want a framework that handles security automatically to secure user information.

An additional characteristic to consider is browser support. When creating a product for someone, it is important to consider the idea that not everyone is going to be using the same products. Since we will be developing a web application, it is ideal to note that the framework we choose, allows for the support of multiple browsers. It would be incredibly unfortunate to develop an amazing application that isn't available for everyone to use. Moving forward, we will be considering browser support when choosing our frontend framework.

At the moment, this specific characteristic does not have as high of a priority as the previously stated characteristics, but extensibility is something we felt was important to simply review**.** Considering we are creating this program for a startup company, it would be naive to disregard the expansion of this product. In the future, if any newer technologies were to be introduced that would improve our product, we would want our framework to be able to handle the integration of said newer technologies. We would continue to follow the desired attributes of our UI to be simple, elegant, and easy to navigate.

Alternatives

Vue.js

Vue is a modular frontend framework that has gained popularity over the past few years. It has reached such popularity as it is being compared to credible frameworks such as React.js and Angular. Although Vue is a somewhat newer framework, large companies like Apple, Nintendo, and Adobe have implemented Vue into their UI. When implementing Vue, the prerequisite knowledge is just basic javascript and html which makes the learning curve faster. Additionally, Vue requires minimal code which allows for better performance and readability.

React.js

React is one of the most popular frontend frameworks to date. React.js was designed by Facebook, which is a very well known company that needs a solid UI. The reason React is so popular is because of its rendering performance. React is known for breaking down a web application into smaller components, so the user can focus on smaller individual projects. Similarly to Vue, React also allows for minimal code.

Angular

Angular is one of React's top competitors. This framework was released in 2010 by Google developers and allowed for the development of web applications to be incredibly faster. Over the years, there have been plenty of new releases, and over 6,723 companies have reported that they use Angular. Similarly, Angular allows for minimal code by the use of data binding and "dependency injection of Angular.js".

Analysis

Efficiency is one of the most important characteristics for us to consider. After watching a variety of youtube videos of people testing all three of the alternative frameworks listed above, the efficiency of each framework is incredibly similar. The major difference between Vue and Angular or React in regards to efficiency, is the amount of added components or libraries that are available. Angular and React are known for their widespread attributes. Although Vue is a newer framework, it has plenty of additional features for us to use. Thus, all three of these frameworks appear to have the necessary components to ensure efficient development within our project (Daityari 2020).

This leads to our next crucial characteristic; the learning curve. Angular and React have a wide variety of tools to use to create a web application but this can lead to an over complicated framework. Many people have stated that Angular is one of the best frameworks to use, but it will take a few months to fully understand how to integrate it. Considering we are limited on time, 2 months to understand a framework is too long. We need something quick and easy to learn. React is similar to Angular in the sense that they both use JSX, a JavaScript syntax extension. This is a factor in the slower learning curve that Angular and React both have (Vue.js 2020). Since Vue does not require JSX, the templates are much more simple and easy to understand.

It is important to consider the browser compatibility for each of these frameworks. As all three frameworks are incredibly popular and used by well known companies, they have been designed to have the best performance. Meaning, all three of these frameworks support all browsers that are ES5-compliant, but they do not support IE8 or below. Therefore, browser support will not affect our overall decision in choosing a framework.

Security is another major characteristic we have chosen to consider. Typically, frontend security is overlooked which can lead to XSS attacks. With further research, Angular, React, and Vue.js all have their own way to sanitize inputted values. Whenever a value is entered into the template being used, the frameworks automatically treat the value as untrusted. Once the value has been entered, they sanitize the value and discard the untrusted values. Since all three of these frameworks provide a way to sanitize values, security will also not affect our overall decision in choosing a frontend framework.

	Vue.js	React.js	Angular
Development Efficiency	XXX	XXX	XXX
Fast Learning Curve	XXX	XX	X
Browser Support	XXX	XXX	XXX
Security	XXX	XXX	XXX

Chosen Approach

After analyzing each framework, and the desired characteristics, we have chosen Vue.js as our frontend framework. Given that each framework essentially meets every characteristic we were looking for, Vue.js had the fastest learning curve out of all the frameworks. As stated previously, a faster learning curve is necessary in order for us to be efficient in development. Even though Vue.js is a newer framework compared to Angular and React, its simplicity in design has been the deciding factor for us.

Proving Feasibility

In order to prove that this framework is feasible, we will be providing a demo within the next few months. We will go through a series of phases to test that this framework is suitable for our project.

The first phase of the demo will be creating wire frames. This is where we will be using Vue to create a basic sketch of what the UI will look like. At this stage, all that is necessary is the placement of each element such as the buttons, text, pictures, or any links. Once we have created our UI prototype, we will review the design and make sure they align with our requirements. From there, we will be able to move on to phase two.

Phase two is going to focus on the visual aspect of our user interface. This is where we would use Vue to allow our UI to come to life. We will be using the provided design tools from Vuetify or Bootstrap-Vue to make the webpage look as described in the requirements. Once we have reviewed the visual design, we will move on to phase three.

The final phase is where the elements on our webpage obtain functionality. By using Vue’s component framework Vuetify, we will easily be able to implement the backend infrastructure as well as the AI model, to our frontend infrastructure. We will then test that the AI, frontend, and backend all integrate properly, and that every component maintains its functionality.

3.5. Database Management

Intro the Issue

For our web application we need a database that will be able to store user profiles with their included information, and images of birds to be further processed by our AI model. It is important that we choose a database that will best fit the scale of our project. To help us select the database that will best fit our needs we will first analyze the CAP theorem. The CAP theorem is simply a thought process that helps us layout the needs of our database and choose the best option for those needs. Focused on comparing tradeoffs of different databases rather than attempting to point us straight towards one or the other. CAP stands for Consistency, Availability, and Partition Tolerance -- these three principles will lay down the foundation to aid us in our final selection.

Desired Characteristics

Before we analyze CAP we would first like to address the issue of security. Security is detrimental to any database as breaches of any kind can make the final product suffer severe consequences. By definition, a data breach is a failure to maintain the confidentiality of data in a database. As previously stated in the introduction we will be working with user data. It is important that our database has the security to maintain this data, whilst also maintaining the transfer of data. As it stands our project is small in scale and thus it will not have a high transaction rate. With that said, we will require a database with a baseline level of security including encryption of moving data, encryption of data at rest, and an admin system to prevent the unauthorized access of said data. Although common in most popular database options, it is important that we choose something with a minimum of “server-side” encryption thus denying strangers from querying it for information. Additionally, a database with a “Role-system” allowing for the selection of admins vs non-admins will also be a crucial point in our selection. Finding a database with both these features should set up the basic security that we require.

From the CAP theorem we first have Consistency. This refers to the requirement that any given database transaction must change affected data only in ways that are allowed. All data must be valid according to all defined rules. This essentially means the database will always have a guaranteed outcome. For our database this will be important. The user will not be performing a large amount of transactions so it is important that this minimal amount be executed consistently. To capitalize on this concept it would be best to select a database that will allow us to perform transactions in isolation (or as close to isolation as possible) minimizing the amount of concurrent transactions. We want to focus on strong consistency as opposed to causal consistency where we wouldn't be too worried if things didn't go as planned. Consistency is prominently higher in SQL databases so this must be taken into consideration.

Availability simply put, is how available the database will be to the user. Ensuring an agreed level of operational performance and up time. Once again our project is currently on a small scale and thus should have a high availability. We want to avoid the loss of service to our users as much as possible. For our project I think these first two Characteristics of CAP are most important to provide the best experience to our bird watching community.

Last in the CAP theorem we have partition tolerance. Meaning that the database must continue to function despite any number of communication breakdowns between nodes in the system. This is normally more important on large multi server systems with multiple machines splitting the workload. Although this is an important characteristic to take into account. For our case we will not be expecting thousands of transactions per minute and thus is not as important as the previous two characteristics. This is the “tradeoff” referred to in the intro. We are hoping to select a database that will trade partition tolerance in exchange for better consistency and availability.

We also desire a database with scalability and speed. Our web app functionality will rely heavily on the scalability and speed of the database. If the database is too slow we will hurt the users overall experience with the product. For example, our front end may need extra time to render if our backend is too slow. Additionally, we are hoping to help build a community around the finished product, so selecting a database that with solid scaling will allow the amount of data stored to grow with the community.

Being that we do not have infinite developing time, it is important that we take the learning curve of our selected database into account. As previously mentioned in section 3.3.2 (Desired Characteristics of our Frontend) we only roughly have a short time period to design and create our project. This being said it might be beneficial to choose a database model we are experienced in and compromise some of our wanted characteristics, rather than pick the database model that has anything and everything and learn it on the fly.

Alternatives

MongoDB
MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

MySQL
MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

Cassandra
Apache Cassandra is a free and open-source, distributed, wide column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.

Analysis

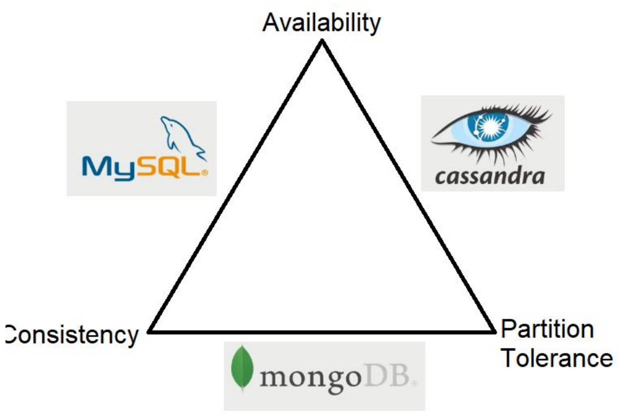
Analyzing the above alternatives in accordance to our desired characteristics. First We will analyze each selection using the CAP theorem. This will provide a good base to further our selection. Followed up by its overall security, scalability, and speed.

Starting with MongoDB, this would be considered a “CP” in relation to CAP. Trading availability for focus on consistency and partition tolerance. MongoDB ensures consistency by assuring reads and writes are issued to the primary member of a replica set. Applications can optionally read from secondary replicas, ensuring that data is eventually consistent by default. For partition Tolerance MongoDB is able to always select a primary set as long as half the replica sets are connected. To ensure two separated networks can not both choose a new primary. When not enough secondaries are connected to each other you can still read from them (but consistency is not ensured), but not write. For scalability and speed MongoDB supports horizontal scaling through means of sharding, distributing data across several machines and facilitating high throughput operations with large sets of data. Lastly MongoDB basic encryption for traveling and stored data, and “Role Based access” which allows for easy implementation of the admin system we are looking for.

Next, mySQL. Considered a “CA” database when using the CAP theorem. Focusing on consistency and availability while sacrificing partition tolerance. Using a relational database model mySQL is able to isolate transactions by running them separately from other concurrent transactions thus preventing interference. For availability, MySQL ensures client requests are load balanced and routed to the correct servers in case of any database failures. Their servers use group replication to replicate data to all members of the cluster while providing fault tolerance. MySQL does have the ability to scale, however if not properly set up things can quickly turn south once the tables have grown too large. This concept remains the same with the speed of the database, it is very CPU reliant and the tables must be optimized around their primary keys to perform efficiently. Additionally, mySQL provides a robust security package upfront including secure connections, authentication services, authentication management and controls, and data encryption.

Last, Apache Cassandra. Considered an “AP” database by the CAP theorem. Focusing on availability and partition tolerance while sacrificing consistency. Cassandra provides availability through linear scaling, this guarantees no single point of failure because data is automatically replicated to multiple nodes. Partition tolerance is handled through a partitioning algorithm which is configured at the cluster level while the partition key is configured at the table level. It is a NOSQL database, however it revolves around the same terminology of tables, rows, and columns. Due to its linear scalability, scaling is made easy by simply adding new nodes, it also has the ability to scale horizontally for easy addition of new data centers. For speed, Cassandra has a unique storage engine that uses Log-structured merge trees compared to the normal B+ trees of other databases. This allows for much faster speeds than most database systems. Lastly Cassandra provides a basic encryption package for data and a pluggable authentication service.

CAP Diagram:



	Consistency	Availability	Partition Tolerance	Scalability	Speed	Security
MongoDB	XXX	X	XXX	XXX	XX	XX
MySQL	XXX	XXX	X	XX	X	XXX
Cassandra	X	XXX	XXX	XXX	XXX	X

Chosen Approach

After our analysis and examination of each database framework, we have decided to use mySQL. Please note that although it may not reflect on our rating scale, all the databases we have considered are capable of performing our desired characteristics. MySQL best fits our desired CAP model while still having scalability, speed, and a high level of security.

Another factor in our decision was the learning curve required for the database. As stated previously, we have limited development time, are unfamiliar with developing in

NoSQL databases, and have limited experience with document based systems like MongoDB. Apache Cassandra was a valid contender as it followed the same table format that appears in relational database models. However, we chose MySQL over Cassandra because we value consistency more than partition tolerance. Not only do we all have experience in SQL and relational database models, but our project is not on a large enough scale to take advantage of the linear and horizontal scaling offered by MongoDB and Cassandra.

Proving Feasibility

To prove the feasibility of our database we will break it down into three separate phases. These phases will include: Building the database, data management, and database integration.

Since mySQL is a SQL database we will be using the relational database model. Before jumping straight into the build we need to consider the relationships that our database will contain. Establishing these relationships will make building the database much easier. It is important we utilize keys to link tables and give each one a fair representation.

Phase two will be data management. We need to construct the database to fit with CRUD programming. This stands for create, read, update, and delete. It should be intuitive and easy to read and understand, while at the same time be able to make quick references to any element in any table. Additionally since we are storing both user data and images it is important that the data stays related but does not interfere with opposing tables.

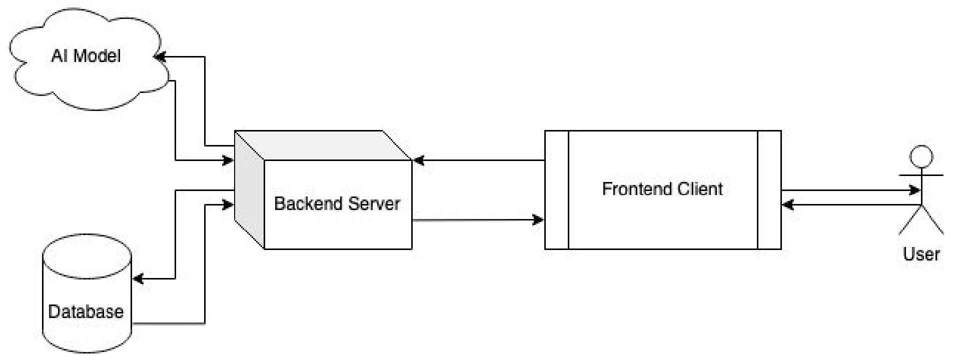
Lastly, will be integrating the database. For this we will use a data engine that will aid in optimizing the organizational aspect of our database. Next we will need to test the database with our front end and back end and make sure that it not only is online and working as a web application, but that users are able to upload and store data using our UI. This will most likely be done using a test account that we will create as a group.

4. Technological Integration

4.1. Putting It All Together

Up to this point, we have identified the specific challenges imposed by this project and systematically chosen technologies that will aid us in solving these problems. What is still missing, however, is an explanation of what roles these technologies each play and how they will coordinate with each other. After all, what is a frontend without a rock-solid backend, and vice versa? What good is an AI model with state-of-the-art accuracy if our users cannot access it? Now that we have objectively decided on the individual frameworks, we need to make sure that we can reasonably integrate these separate technologies together into the overarching solution we need. This necessity to integrate our pieces brings us to an important point: at a high level, what does our system look like?

4.2. Our System Diagram



4.3. Full Integration

Examining the first iteration of our system diagram, it is clear that the user only interacts with our frontend client, leaving all other components of the system transparent to them, which is how it should be. Our frontend should be nothing but a simple and elegant proxy for communicating with our backend (which in turn communicates with our AI model and database). Because of this design, which is common in web applications, it is critical that the frameworks we use to design both our frontend and backend do not interfere with the core communication. Because this design is so common, all the frameworks we have considered fulfill this requirement.

Now that we know our backend and frontend integrate smoothly, the only other concerns worth mentioning now are those between the backend, AI model, and database. Similar to close integration of frontend and backend frameworks, accessible databases are another key contributor to meaningful web applications. Because many of the most successful web applications are centered around the storage and application of user data (e.g., Facebook buying your search history to target you with specific ads), all of the database frameworks that we have considered are also deeply integrable within a web environment.

The last component that needs integration is the AI model. Because the entire AI module will likely be written in Python, it would be advantageous to also write our backend in Python so that we can ensure some uniformity. In addition to making our backend uniform, it will also ensure that any dependencies of the backend are easily ported to our AI module and vice versa. This is important because with different languages come different implementations, and it would only be a matter of time for bugs to start arising from the subtleties of those differing implementations.

By using tried and tested technologies for our frontend, backend, and database, the only real logistical challenge imposed by this project is the integration of the AI model. There are, of course, other logistical challenges, but those are the ones ever present in web programming, and for which there are endless resources on potential solutions. Because we have chosen Keras and Flask for our AI training and backend, respectively, we ensure a consistency with languages and dependencies, thus alleviating some of the conflicts that could occur with the use of different implementations over different languages.

In this way, we ensure a reasonable plan for the integration of all technologies required to solve the problem presented to us. Though the final measure of how well these technologies integrate will be the success of our prototype, we have at least ensured a higher probability of success in integration by considering each technology and the potential collisions it may have with the other technologies we plan to leverage.

5. Conclusion

David Plemmons, our client, has a vision that his bird feeder will be able provide an interactive and user friendly UI in combination with a species identification system to let his users know what kinds of birds they are feeding. After considering his desired specifications, we have broken up the project into four separate components. These include the frontend framework, backend framework, AI model, and the database. To conquer this challenging task we divided the four topics among each of our group members. Each member did specific research for their chosen topic while keeping other members in the loop about their findings and decisions, so that all research efforts could be coordinated. After collaborating once more we came to a final decision for each component.

Upon careful examination and analysis of our available options for each type of framework we will need to use, we have formed the foundations of our technology stack. We have chosen Vue.js for our frontend framework, Flask for our backend framework, Keras for our ML framework, and MySQL for our database system. We felt that these options best suited our needs, and will have the lowest combined learning curve while still providing all functionality that we require for the project.

Moving forward, Our next step is to begin making prototypes with our selected tools, and progress towards a completely integrated system. It is key that every component of the system is able to communicate well with one another to ensure that the software powering the pEEp feeder can last a lifetime. We envision that our finished product will have a user interface that is easy to navigate for our bird watchers. This user interface will communicate with the backend and database to store images and data that the user will generate through the use of the pEEp feeder. Lastly, what we believe sets our product apart from competitors is the AI model that will help users identify and learn about the birds that they feed everyday, providing a truly unique experience from the comfort of your home.

6. References

Amilkanthwar, Vivek. “Choosing a Deep Learning Framework.” *Medium*, In Pursuit of Artificial Intelligence, 28 Feb. 2019, medium.com/in-pursuit-of-artificial-intelligence/choosing-a-deep-learning-framework-5669a85ebc3f.

“Comparison with Other Frameworks - Vue.js.” - *Vue.js*, vuejs.org/v2/guide/comparison.html.

Daityari, Shaumik. “Angular vs React vs Vue: Which Framework to Choose in 2020.” *CodeinWP*, VertiStudio, 9 Aug. 2020, www.codeinwp.com/blog/angular-vs-vue-vs-react/.

Draelos, Author: Rachel, et al. “The History of Convolutional Neural Networks.” *Glass Box*, 30 Jan. 2020, glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/.

“Learning PyTorch with Examples¶.” *Learning PyTorch with Examples - PyTorch Tutorials 1.6.0 Documentation*, pytorch.org/tutorials/beginner/pytorch_with_examples.html.

“Libraries & Extensions :   TensorFlow.” *TensorFlow*, www.tensorflow.org/resources/libraries-extensions.

“Managed Apache Cassandra as a Service: Aiven.” *Aiven.io*, aiven.io/cassandra?utm_campaign=10954370547.

Singh, Manoj. “Understanding Frontend Security.” *Medium*, Medium, 9 Nov. 2019, medium.com/@manojsingh047/understanding-frontend-security-ff6585395534.

Team, Keras. “Keras Documentation: Simple MNIST Convnet.” *Keras*, keras.io/examples/vision/mnist_convnet/.

“The Most Popular Database for Modern Apps.” *MongoDB*, www.mongodb.com/.
MySQL, www.mysql.com/.

Sundaresh, et al. “Keras vs. TensorFlow - Which One Is Better and Which One Should I Learn?” *PyImageSearch*, 18 Apr. 2020, www.pyimagesearch.com/2018/10/08/keras-vs-tensorflow-which-one-is-better-and-which-one-should-i-learn/.

“Welcome to Flask.” *Welcome to Flask - Flask Documentation (1.1.x)*, flask.palletsprojects.com/en/1.1.x/.