

B.Sc. (Hons) in Software Development



Ollscoil  
Teicneolaíochta  
an Atlantaigh

Atlantic  
Technological  
University

## PROJECT SONIC BLOOM

**By**  
**KEVIN MCSHANE**

April 27, 2025

### **Minor Dissertation**

**Department of Computer Science & Applied Physics,  
School of Science & Computing,  
Atlantic Technological University (ATU), Galway.**

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Project context . . . . .	2
1.2	Scope analysis . . . . .	3
1.2.1	Defined scope . . . . .	3
1.2.2	Prioritize key functionality first . . . . .	3
1.3	Project objectives . . . . .	4
1.3.1	How it is meant to work. . . . .	4
1.4	Project resources . . . . .	5
1.4.1	Front-end . . . . .	5
1.4.2	Back-end . . . . .	5
1.5	Document structure . . . . .	6
1.5.1	Introduction . . . . .	6
1.5.2	Methodology . . . . .	6
1.5.3	Technology Review . . . . .	6
1.5.4	System Design . . . . .	6
1.5.5	System Evaluation . . . . .	6
1.5.6	Conclusion . . . . .	6
<b>2</b>	<b>Methodology</b>	<b>7</b>
2.1	Project management methodology . . . . .	7
2.2	Research methodology: An empirical approach to software develop- ment . . . . .	7
2.2.1	Experimentation . . . . .	7
2.2.2	Testing: Observation, trial and error . . . . .	8
2.3	Agile/Incremental and iterative development . . . . .	8
2.3.1	Planning . . . . .	8
2.3.2	Implementation . . . . .	8
2.3.3	Implementation: lessons learned . . . . .	9
2.3.4	Feedback . . . . .	9
2.4	Github documentation . . . . .	9
2.5	Learning outcomes . . . . .	9
2.6	Methodology and project management approach validity . . . . .	10
2.6.1	Advantages . . . . .	10
2.6.2	Disadvantages . . . . .	10
<b>3</b>	<b>Technology Review</b>	<b>11</b>
3.1	Python . . . . .	11
3.1.1	conceptual foundations . . . . .	11
3.2	Flask framework . . . . .	12
3.2.1	What is flask framework . . . . .	12
3.2.2	Comparative performance review: Python vs node.js . . . . .	12

3.2.4	Conclusion: Why flask over Node.js . . . . .	14
3.3	TypeScript . . . . .	14
3.3.1	Comparative performance review: JavaScript vs TypeScript	14
3.3.2	Conclusion: Why TypeScript over JavaScript . . . . .	15
3.3.3	Hosting . . . . .	15
3.4	Ionic angular . . . . .	15
3.4.1	What is angular? . . . . .	15
3.4.2	Why choose angular over react or vue? . . . . .	15
3.4.3	Deployment . . . . .	16
3.5	Google maps API . . . . .	16
3.5.1	Types of development keys acquired . . . . .	16
3.5.2	What is google cloud services . . . . .	16
3.5.3	Securing a free trial . . . . .	17
3.5.4	Did the upgrade help . . . . .	17
3.6	Screenshot the map: Html2canvas and selenium . . . . .	17
3.7	RXJS: State synchronization . . . . .	18
3.8	Github pages . . . . .	18
3.9	Github workflows . . . . .	19
3.10	Kaleidoscope: Bio-acoustics Sound Analysis . . . . .	19
3.10.1	The expected format from the user's kaleidoscope analysis .	19
3.11	Pandas . . . . .	20
3.11.1	Key Features . . . . .	20
<b>4</b>	<b>System Design</b>	<b>21</b>
4.1	Overall Architecture Overview . . . . .	21
4.2	Component Coupling Strategy . . . . .	22
4.2.1	REST APIs . . . . .	22
4.2.2	Platform as a service deployment: Github pages . . . . .	22
4.3	Front-End Architecture . . . . .	22
4.3.1	App type . . . . .	22
4.3.2	Important files . . . . .	22
4.3.3	Pages . . . . .	23
4.3.4	Components . . . . .	23
4.4	Back-End Architecture . . . . .	24
4.5	Architectural Trade-offs . . . . .	24
4.5.1	Front-end tradeoffs . . . . .	24
<b>5</b>	<b>System Evaluation</b>	<b>25</b>
5.1	Project goals evaluation . . . . .	25
5.1.1	Scope: what has been implemented . . . . .	25
5.2	Software testing . . . . .	25

5.2.1	Automated tests: progression status explained . . . . .	25
5.2.2	Manual testing . . . . .	26
5.3	Project weaknesses and limitations . . . . .	26
5.3.1	Project management: schedule creep . . . . .	26
5.3.2	Reasoning for this pitfall: The progress stop surrounded by google maps . . . . .	26
5.3.3	Timeline creation functionality . . . . .	27
5.4	Lessons learned . . . . .	27
<b>6</b>	<b>Conclusion</b>	<b>28</b>
6.1	Summary of Context and Objectives . . . . .	28
6.1.1	The primary objectives . . . . .	28
6.2	Key Findings from System Evaluation . . . . .	28
6.2.1	Implemented Features . . . . .	29
6.2.2	Unfulfilled Goals . . . . .	29
6.2.3	Problems . . . . .	29
6.2.4	Lessons Learned . . . . .	29
6.3	Serendipitous Insights . . . . .	29
6.3.1	Technology Trade-offs . . . . .	29
6.3.2	Third-Party API Realities . . . . .	30
6.3.3	Agile Flexibility . . . . .	30
6.4	Possible future developments for this project . . . . .	30
6.4.1	Back-End Completion . . . . .	30
6.4.2	Community Features . . . . .	30
6.5	Closing Remarks . . . . .	30
<b>7</b>	<b>Appendix</b>	<b>31</b>
7.1	Github organization link . . . . .	31
7.2	Github ionic front end link . . . . .	31
7.3	Github flask back end link . . . . .	31
7.4	Github pages ionic deployment link . . . . .	31
7.5	Screen-cast link . . . . .	31

# List of Figures

1.1	Example output from kaleidoscope . . . . .	2
3.1	Gaining free embed key for google maps from share or embed map .	16
3.2	A screenshot of id.csv open in excel . . . . .	19
4.1	System architecture: Relationships between components . . . . .	21

# List of Tables

5.1	Scope progress . . . . .	25
-----	--------------------------	----

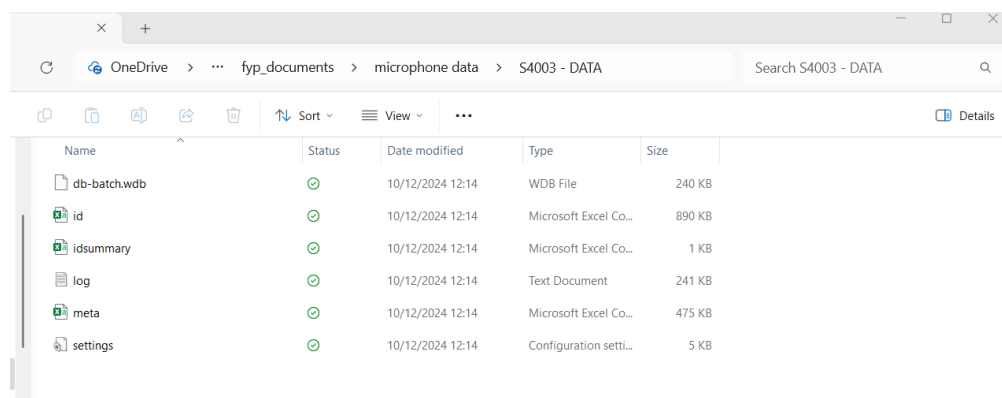
# Chapter 1

## Introduction

Project Sonic Bloom was carried out with the end goal of producing a software solution for ecology researchers. The softwares primary function is to take a list of events recorded by the user and display these events visually on a map for the user to see in chronological order.

### 1.1 Project context

In the field of ecology, making observations about animal behavior is very crucial to the process of wildlife conservation. There are existing applications available for research students to utilize in order to understand their recordings. A prominent example of one of these applications is called kaleidoscope. One main feature this application has is its ability to take a folder containing many audio files in .wav format and analyze the data.



The screenshot shows a OneDrive file explorer window with the address bar indicating the path: OneDrive > ... fyp\_documents > microphone data > S4003 - DATA. The search bar contains 'Search S4003 - DATA'. The file list is displayed in a table with columns: Name, Status, Date modified, Type, and Size. All files have a status of 'OK' (green circle with a checkmark) and were modified on 10/12/2024 at 12:14.

Name	Status	Date modified	Type	Size
db-batch.wdb	OK	10/12/2024 12:14	WDB File	240 KB
id	OK	10/12/2024 12:14	Microsoft Excel Co...	890 KB
idsummary	OK	10/12/2024 12:14	Microsoft Excel Co...	1 KB
log	OK	10/12/2024 12:14	Text Document	241 KB
meta	OK	10/12/2024 12:14	Microsoft Excel Co...	475 KB
settings	OK	10/12/2024 12:14	Configuration setti...	5 KB

Figure 1.1: Example output from kaleidoscope

Note that the files appearing as Microsoft excel files are in .csv format. The file id.csv contains details about each recording such as auto identification of species and the date and time of the recording.

Each id.csv file is associated with one microphone's recordings.

Although kaleidoscope is an extremely useful tool for the analysis of ecological data there are no data visualization functions that give the user an idea of when the events happen in relation to each other, aside from the timestamps in the processed data.

However, there are tools that can be used to manually re-create the timeline of events graphically. This can be done using R (programming language) libraries such as ggplot2 or leaflet, although this requires extensive effort the user and programming skills.

This project was developed to extract the analyzed data and autonomously create a visual timeline of events, with the aim of resolving this issue.

## 1.2 Scope analysis

The scope of the project was very lightly documented during the software development life cycle (SDLC), it was only defined officially through the gantt charts required for the project proposal document and the Christmas seminar presentation.

### 1.2.1 Defined scope

- The ability to create a visual timeline of ecological events from the user data.
- Deployment of the application to the public domain for user accessibility. This was achieved for the front-end application, deployed to the website <https://project-sonic-bloom.github.io/Ionic-Front-End/tabs/home>
- The application must employ good system architecture (two-tier client-server structure, see term explanation in the system design chapter).
- The implementation of automated code tests.

### 1.2.2 Prioritize key functionality first

The first part of the scope to be implemented was the setting up of the entire development environment, this used things that are considered good practice such



as continuous integration and continuous development (CI/CD) testing for the backend server.

Unfortunately the time that went into setting up this feature was never capitalized on as back-end development was limited due to schedule creep caused by implementations of the google maps API that were not going as planned.

## **The learning outcome**

For time constrained academic projects, development environments should evolve iteratively. Initial efforts should focus on validating high-risk components, with infrastructure automation introduced only after core functionality is stable. This balances rigor with pragmatic delivery.

## **1.3 Project objectives**

The goal of this project was to become an application that could visualize the data in id.csv for presentation purposes.

### **1.3.1 How it is meant to work.**

The application was deployed to a webpage containing a satellite view of the google map API, where the researcher could locate where the field they worked in, and then drag and drop their csv files where the microphones had been recording events.

After the file drop, a marker is made on the map with the name of the microphone beneath it. When the the marker is dropped, the user is then ready for the data to be visualized and the csv files are sent to the server as well as the state of the map.

As this feature did not get implemented successfully, the following is a description of the prototype that would be the next goal for the project's development. The csv files would be read for the events, the next step being to sort the events to appear in chronological order.

After the image of the map would be slightly adjusted representing what happened around each microphone, the output would be in the form of many images and each image would represent all of the events that happened within 15 minute time allotments.

## 1.4 Project resources

There are two repositories to split the code-base into it's front-end and back-end applications

### 1.4.1 Front-end

Front-end URL:

<https://github.com/Project-Sonic-Bloom/Ionic-Front-End>

#### Front-end file structure description:

- `.github/Workflows/deploy-to-gh.yml` - this is a script that builds the app and pushes it to the website <https://project-sonic-bloom.github.io/Ionic-Front-End/tabs/home>
- `Sonic-Bloom-Client` folder contains the ionic app. The app contents are described in the system design chapter.
- `README.md` - This file contains instructions on how to get the app working on your local machine (note you must follow the quickstart guide, if you want to run it locally, the backend server can be tested with the deployment, see the secreted environment file in the appendices)

### 1.4.2 Back-end

Back-end URL:

<https://github.com/Project-Sonic-Bloom/Flask-Back-End>

#### Back-end file structure description:

- `.github/Workflows/run-tests.yml` - This script runs code tests found in `Flask-Back-End/scripts/test/`
- `Flask-Back-End/scripts/main` - This is where methods for processing data is held.
- `Flask-Back-End/scripts/test` - An empty file that run after every commit, it is ready to implement software tests for the code in the main folder.
- `Sonic-Bloom-Server.py` - The flask server logic containing the code required for launching the server communications and httplisteners.

## **1.5 Document structure**

### **1.5.1 Introduction**

This chapter will inform the reader of this project's goals and the context for its existence.

### **1.5.2 Methodology**

The methodology informs the reader of the project management style of development and challenges to the schedule were handled.

### **1.5.3 Technology Review**

This chapter informs the user of the research of the technology undertaken in order to develop the project. The technologies contribution to the project will be analyzed and reported on for the user to gauge it's usefulness. It will warn other users of pitfalls that may occur when trying to use the technology in the incorrect way.

### **1.5.4 System Design**

This chapter outlines the way the technology is structured and how it works as a whole. It informs the user of the different web technologies and modern practices used when creating the code-base. A good systems design should give the reader intimate knowledge of how the application works, communication styles, hosting methods and versioning of the project.

### **1.5.5 System Evaluation**

The system evaluation contains further discussion of the system's implementation. It will elicit the project's findings about system design and the learning outcomes gained from the project and it's limitations.

### **1.5.6 Conclusion**

The conclusion will reflect on the goals outlined in this chapter and measure the successes or failures of the project, this chapter also highlights the learning outcomes outlined in the system evaluation.

# Chapter 2

## Methodology

This section outlines the development process, planning techniques, validation processes, and tools used during the project. It addresses the validity of the methodologies adopted and lessons learned.

### 2.1 Project management methodology

Iterative development was prioritized through weekly meetings and gaining feedback from the project supervisor.

This project management methodology is especially well suited for solo development. Scope changes can happen fast in a project so the weekly meetings replacing formal documentation such as JIRA saved time.

### 2.2 Research methodology: An empirical approach to software development

#### 2.2.1 Experimentation

The project used software development in practice as the main catalyst for research, using new libraries is simple when you find one good quality tutorial, it does not have to be from a formal source just an accurate one. An over an entirely research oriented approach is unrealistic as being too selective about which sources you investigate would make finding a solution too complicated for example YouTube tutorials are often overlooked as they take more time to research but they often contain niche answers to errors you may be facing.

Another benefit of this research methodology is that figuring out whether an approach will work is done faster in practice while using a tutorial rather than doing extensive research about the libraries before any code implementation which is seen in less adaptive such as waterfall.

### **2.2.2 Testing: Observation, trial and error**

Once the code is running, manual testing of the new features were conducted. Writing test code is a time expensive task for the development style this project uses. If the effects of the code were not immediately integrated with other features, effects of the code were seen via the console in the browser's developer options or the terminal. Unfortunately no automated tests were implemented despite the intentions to do so.

## **2.3 Agile/Incremental and iterative development**

An informal Agile process guided the project with incremental progress through continuous cycles of feedback. While no sprints or storyboards were documented officially, development was carried out in an iterative way:

### **2.3.1 Planning**

Goals were discussed during the weekly meeting with the project supervisor.

If the goal for the previous week was not met, the attempted implementation was discussed as well as the possible explanations of why it did not work.

A new approach would be discussed and set as the next week's goal despite it's effect on the scope.

### **2.3.2 Implementation**

The build made use of Ionic Angular as a front-end technology and Flask for back-end processing of data. A substantial amount of preliminary effort went into establishing an effective end-to-end development setup where the choices made were well researched and intentional as outlined in the technical review's comparative performance reviews.

Most of the development cycle was spent working on incorporating the map visualization logic properly and attempting to deliver the map state among other data from the front-end to the back-end. Such tech issues became the critical bottleneck

preventing further project developments. Further details about the progress block surrounding the google maps API have been documented in the system evaluation chapter under weaknesses and limitations

The iterative development process allowed continuous debugging of this feature through ongoing evaluation and correction cycles every week, however this approach was not enough to overcome this issue as it introduced schedule creep, this was the main flaw in the methodology and it is noted further under system evaluation.

### **2.3.3 Implementation: lessons learned**

I learned that it is crucial to address schedule creep and how to do it.

The first way of addressing schedule creep is to move onto another feature instead of continuing to pursue the development of a problematic feature, however this is not possible if it is a key feature the application cannot be complete without.

The second method of progressing past schedule creep is to resolve the problem by changing the implementation of the code drastically, for example finding an alternative library ditching the problematic implementation that was hindering progress.

These tactics only mitigate schedule creep and involve Fail-Fast Termination of the problem in order to work.

### **2.3.4 Feedback**

Weekly supervisor meetings substituted for formal sprint reviews, providing guidance for changes.

## **2.4 Github documentation**

There was a point where github issues was considered as a tool for document the changes week by week, however this idea was abandoned due to documentation being time-costly.

## **2.5 Learning outcomes**

I learned that iterative development can be good however to use this methodology effectively there must be an accurate project scope:time ratio or surplus time to

accommodate unforeseen challenges, an awareness of negative project management patterns such as schedule creep.

I gained more appreciation for project management practices and tools. While this style of informal project management and research methodology suited this project it would not be as applicable to larger scale projects as there would be multiple people working on the project. Customer reassurance and scope management must be more precise to deliver project tailored to the customers needs.

## **2.6 Methodology and project management approach validity**

This approach had both advantages and disadvantages, when compared to a more strict approach.

### **2.6.1 Advantages**

The advantages of this the research methodology is gaining quick experience with many new libraries.

The advantages of the iterative development project management style is that time and scope management are flexible as you are not required to spend extra time to amend information in project management tools such as jira.

### **2.6.2 Disadvantages**

The disadvantages of this style of project management is that it would not work in a group setting and it provides little documentation of the development process.

# Chapter 3

## Technology Review

### 3.1 Python

#### 3.1.1 conceptual foundations

Python, as a high-level, interpreted language, has emerged as a cornerstone in computation research and software development due to its design philosophy, flexibility, and extensive ecosystem. Here, its conceptual foundations are outlined, accentuating its relevance in academia and alignment with modern technological requirements.

#### Design Philosophy and Basic Principles

Python's language construction emphasizes readability, simplicity, and expressiveness. Its syntax is noted for minimalism by block structuring through indentation, reducing cognitive load and code maintainability effort. Python's object-oriented, functional, and procedural programming paradigms allow for problem-solving flexibility, making it accessible to a broad spectrum of applications from web development to scientific computing.

Dynamic typing and automatic memory management are among its strong points that make prototyping and iterative development easier. These features have made Python a cornerstone for rapid experimentation in academia, as can be seen from its integration with Jupyter Notebooks for interactive data analysis.

#### Evolution and Academic Adoption

Python's evolution reflects its accommodation of emerging computational needs. Early versions targeted scripting and automation, whereas newer versions sup-



port advanced features like asynchronous programming and type hints. Academic uptake has increased due to its adoption in data science and machine learning (ML) through packages like NumPy, pandas, and scikit-learn. An illustration is PyOphidia, a Python library for high-performance data analysis, being utilized as a showcase for its application in large-scale scientific data analysis on parallel computing architectures.[1]

## Key Libraries and Frameworks

**Data Analysis:** Pandas and Matplotlib offer intensive data manipulation and visualization, a must for empirical research. This project is using panda to analyze csv files from the user

**Machine Learning:** TensorFlow and scikit-learn are widely cited in ML literature to build models like decision trees and neural networks. At the beginning of this project before the scope was defined machine learning was considered as a possible feature.

## Academic and Research Applications

Python's simplicity and easy to read nature is why it is so popular, it's rapid development for projects make it an ideal candidate for small academic research and projects.

## 3.2 Flask framework

### 3.2.1 What is flask framework

Flask is a lightweight, modular Python web framework known for its simplicity and flexibility, Which is why it fit for this project, keeping code uncomplicated and short.

Another advantage of using the flask framework is that it is very widely adopted, so even if an error occurs there is a large user base. "According to the Stack Overflow Developer Survey conducted in 2021, Python is estimated to be the third most widely used programming language in 2021".[2]

### 3.2.2 Comparative performance review: Python vs node.js

[3]

Node.js is an event-driven, single-threaded, non-blocking I/O model and thus is

highly efficient for I/O-bound and real-time applications.

The cited source mentions the fact that Node.js is superior to Python-Web libraries such as flask at serving multiple requests simultaneously and benchmarks demonstrate Node.js serving 2–7 times more requests per second than Python in cases like database reading or high concurrency.

For example, in their "Hello World" benchmark, Node.js issued 3,703 requests/sec compared to Python's 559 requests/sec under the same conditions. That performance difference occurs because Node.js is an asynchronous platform that never blocks operation a disadvantage in Python's synchronous platforms like Flask.

However, Python is still useful for computation-intensive work with scientific libraries such as NumPy or rapid development iteration cycles.

For Sonic Bloom, Python simplicity and Flask's lightweight framework were sufficient for the requirement of rapid iteration in the project and integration of Pandas to handle Csv data.

Node.js, although better for I/O, would have added complexity for managing asynchronous code in an academic project of such small scale, also the requirement to read csv data is fulfilled easier with flask as Pandas is a very easy library to use.

### 3.2.3 Flask limitations

#### **Performance Bottlenecks:**

Python's Global Interpreter Lock (GIL) constrains multi-threading, limiting performance on CPU-bound tasks. In the case of large data sets or high concurrency, it is more advisable to use Node.js's event loop or compiled languages such as Go.

#### **Scalability:**

As demonstrated by the cited source, Python-Web does not perform well under high concurrent loads compared to Node.js. Even though Flask can be sufficient for the low-end backend needs of Sonic Bloom, scaling up to thousands of users would demand substitutes like asynchronous frameworks such as FastAPI or Node.js. However as of right now scalability is not a concern but it is a limitation worth discussing for future reference.

#### **Ecosystem Fragmentation:**

Python's large library ecosystem can lead to dependency issues, while Node.js's npm is closer integrated to full-stack JavaScript projects.

### 3.2.4 Conclusion: Why flask over Node.js

Python was used for the benefit of simplicity, robust data processing libraries like Pandas and Flask's light backend features.

While Node.js is better suited to I/O-bound applications with excellent performance, the project needs favor development speed over the performance concurrency based libraries offer. Python is the logical choice.

Later versions requiring real-time data processing or scalability will be cause for reconsideration of this choice, but for an academic prototype, Python's trade-offs were worthwhile.

## 3.3 TypeScript

Typescript is an open source and strongly typed superset of JavaScript developed by Microsoft. By allowing developers to define types TypeScript catches errors during compilation rather than at runtime.[4]

One of TypeScript's key advantages is its compatibility with JavaScript code which can be incrementally migrated to TypeScript, and all valid JavaScript is syntactically valid TypeScript. The language compiles down to plain JavaScript, ensuring cross-browser and cross-platform compatibility.

### 3.3.1 Comparative performance review: JavaScript vs TypeScript

The findings in this comparative performance come from the study titled "To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub".[5]

Here are the main findings of the conference:

#### Code Quality

TypeScript applications demonstrate a lot less code smells compared to JavaScript, suggesting improved coding practices. According to the study "code smells are indicators of low code quality which may impact maintainability".[5]

#### Code Understandability:

Cognitive complexity metrics tell us that TypeScript applications are easier to read due to features like static typing and real time compiler error detection.

### **Type Safety Benefits:**

Although TypeScript enhances type safety, the impact on bug reduction is nuanced. The study found that projects heavily using the any type—a way to bypass strict typing—exhibit diminished software quality. Limiting any usage correlates with better readability and fewer code issues.[5]

### **Limitations:**

Surprisingly, the study found no significant reduction in bug proneness or resolution times for TypeScript projects compared to JavaScript, challenging the assumption that static typing inherently ensures faster debugging.

### **3.3.2 Conclusion: Why TypeScript over JavaScript**

TypeScript also integrates seamlessly into frameworks like Angular, complementing its dependency injection patterns and modularity.

### **3.3.3 Hosting**

Although the angular client was deployed successfully to the web, it lacks functionality without the flask server being hosted.

The method that would have been used to host the flask server would have been through Heroku.

## **3.4 Ionic angular**

This project uses angular as it is very well supported on all devices and it serves as a very adaptable GUI for the user.

### **3.4.1 What is angular?**

Angular written in TypeScript is an open-source web application framework developed by Google. [6]

### **3.4.2 Why choose angular over react or vue?**

Unlike React/Vue, Angular provides built in routing making page generation easier. Although the cited source favors react and vue [6] Angular is still a useful tool which is easy to learn as it has a very defined structure.

### 3.4.3 Deployment

Using a very useful refactored script provided by a lecturer from Atlantic Technological University Galway, deploying the project to github pages was very simple and free of charge. This level of accessibility is very useful to the development of the project.

## 3.5 Google maps API

### 3.5.1 Types of development keys acquired

During the project's development a personal use key had been used at the start without the use of google cloud services.

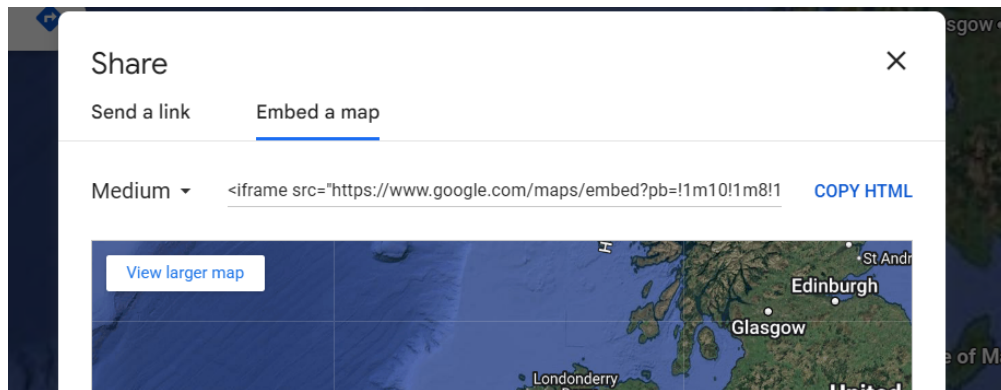


Figure 3.1: Gaining free embed key for google maps from share or embed map

This would have fulfilled the functionality the project needed however there was no way to record the map's state without better API privileges.

The original plan was to record the map's state via a client side screenshot that would be sent to the backend for processing, however google does not allow this. This solidified the need for better google API privileges, which is acquired through google cloud services.

### 3.5.2 What is google cloud services

Google cloud is a suite of online services provided by Google for computing, storage, networking, and in this case the google maps API.[7]

### 3.5.3 Securing a free trial

There is a free-trial period for google cloud services, where they provide 300 dollars worth (USD) of credits for a 90 period with no automatic charges after.

To get this you must use google cloud console with some credentials, google does this to mitigate fraud.

One condition is that you must sign into email that has been in use for an extended amount of time.

The second step for authentication is that you must create a billing account with a valid card.

Google does not allow the use of digital cards single or multiple use. However google does accept physical revolute card, this could help bypass any incurring charges should they happen.

### 3.5.4 Did the upgrade help

The upgrade did help as I could get the latitude, longitude, marker positions and zoom setting of the map to store the map state and replicate them using selenium in the back-end to get around the issues of capturing the maps state, which will be discussed under the 3.5 Screenshot the map: Html2canvas and selenium subheading.

## 3.6 Screenshot the map: Html2canvas and selenium

Html2canvas is a package for angular that could produce a screenshot from part of the page, while it is not actually taking a screenshot the page it does re-render the page through the data found in Document object model (DOM) then it takes that render of the page and turns it into an image.[8]

Although this is a useful tool, this approach did not work.

According to the documentation, "It doesn't magically circumvent any browser content policy restrictions either, so rendering cross-origin content will require a proxy to get the content to the same origin".[8]

I figured google does not allow the browser to view the contents of the embedded map due to their CORS policy after attempting to use it and getting a blank image back.

Selenium is an automated software testing tool, this could be utilized to open the web-app, restore the user's map using cookies and screenshot the state of the map.

However it is primarily used for software testing, if the front-end required automated software tests for the GUI and it's features, it could be utilized in the back-end to do so. Selenium by default opens a browser when it tests a web-page, however this can be bypassed to ensure the browser doesn't viably appear while selenium runs it's code, this visibility setting from selenium is called headless mode.

### 3.7 RXJS: State synchronization

Reactive Extensions for JavaScript (RXJS) is a very useful library that creates observable variables which update asynchronously. Angular supports RXJS with it's own built in library which I used when trying to store shared variables between components[9]

### 3.8 Github pages

GitHub Pages is a secure, developer-friendly host for static sites directly from GitHub repositories with seamless integration with the Git workflow. Ideal for personal portfolios, project documentation, or organizational sites, it simplifies web deployment by auto-updating with repository commits.

Its tight integration with GitHub makes it easy to keep updated to the latest build.

It does have some restrictions. It only supports static content, meaning no server-side scripting languages can be hosted, this means that in the case of the flask server it cannot be hosted using this feature.

Another limitation of github pages is the soft bandwidth caps at 100GB per month, and long build times could cap out big projects.

Despite all these constraints, GitHub Pages is perfect for projects of an academic setting or small scale such as this one, which need a simple graphical user-interface (GUI) with integrated version control inclusion and low-cost hosting.

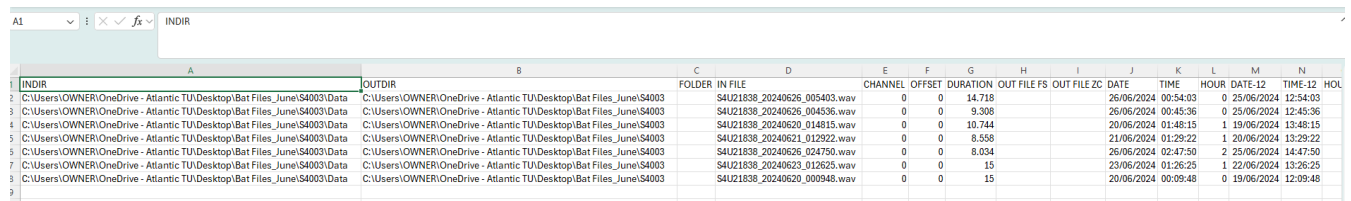
Another big advantage hosting on github pages brings to this project is the ability to hide API keys and sensitive variables using the github secrets feature.

## 3.9 Github workflows

## 3.10 Kaleidoscope: Bio-acoustics Sound Analysis

Kaleidoscope is an application for ecology students, it automates the evaluation of their recordings providing useful features such as the auto-identification of species. This software's output is crucial to the project as it is the format of data being visualized.

### 3.10.1 The expected format from the user's kaleidoscope analysis



INDIR	OUTDIR	FOLDER	IN FILE	CHANNEL	OFFSET	DURATION	OUT FILE FS	OUT FILE ZC	DATE	TIME	HOUR	DATE-12	TIME-12	HOUR
C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003\Data	C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003		S4U21838_20240626_005403.wav	0	0	14.718			26/06/2024	00:54:03	0	25/06/2024	12:54:03	
C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003\Data	C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003		S4U21838_20240626_004536.wav	0	0	9.308			26/06/2024	00:45:36	0	25/06/2024	12:45:36	
C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003\Data	C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003		S4U21838_20240620_014815.wav	0	0	10.744			20/06/2024	01:48:15	1	19/06/2024	13:48:15	
C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003\Data	C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003		S4U21838_20240621_012922.wav	0	0	8.558			21/06/2024	01:29:22	1	20/06/2024	13:29:22	
C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003\Data	C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003		S4U21838_20240626_024750.wav	0	0	8.034			26/06/2024	02:47:50	2	25/06/2024	14:47:50	
C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003\Data	C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003		S4U21838_20240623_012625.wav	0	0	15			23/06/2024	01:26:25	1	22/06/2024	13:26:25	
C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003>Data	C:\Users\OWNER\OneDrive - Atlantic TU\Desktop\Bat Files_June\S4003		S4U21838_20240620_000948.wav	0	0	15			20/06/2024	00:09:48	0	19/06/2024	12:09:48	

Figure 3.2: A screenshot of id.csv open in excel

## The useful attributes

### INDIR and OUTDIR

The in directory (INDIR) attribute contains the file location of each .wav file that was given to kaleidoscope for processing.

The OUTDIR is the file path where the processed output is going to be placed.

The useful part about this attribute is that the stored file path contains the label for the microphone.

### Date and time

Using the data manipulation library Pandas, we can sort the entries in order of time, which is crucial for creating a timeline.



## AUTO ID\*

Although this does not appear in the screenshot it exists. It contains a 6-letter abbreviation of what type of species it has identified. For example, the value MYODAU refers to the *Myotis daubentonii* species of bat.[10]

## 3.11 Pandas

Pandas is an open-source, high-performance Python library used for data analysis and manipulation that is popular for its ability to handle structured data effectively. Its data structures, `DataFrame` and `Series`, provide expressive ways of analyzing, cleaning, and transforming tabular data, making it an essential tool for project work that involves ecological or time-series data like the Sonic Bloom.

### 3.11.1 Key Features

#### Data Ingestion:

Pandas readily reads and works with `.csv` files, the primary output format of Kaleidoscope. For example, the `id.csv` files—carrying timestamps, species codes and microphone positions—are read in with efficiently.

#### Time-Series Handling:

The library’s intrinsic datetime operations enable chronological sorting of ecological events, a critical step for timeline visualization. `resample()` functions facilitate aggregation by hourly/daily intervals with ease.[11]

#### Data Cleaning:

Pandas performs especially well in handling missing values and filtering out unwanted entries ensuring only valid bio-acoustic data is being plotted.

#### Performance Considerations

While Pandas is designed for datasets of intermediate size, its in-memory processing does create a bottleneck for extremely large files. However, for Sonic Bloom’s academic scope, those limitations were offset by its simplicity and Python ecosystem integration.

# Chapter 4

## System Design

### 4.1 Overall Architecture Overview

The Sonic Bloom system follows a two-tier architecture pattern with clear separation between presentation and application logic.[12]

The design implements modern web development practices with:

Front-End Client: Ionic angular web-app

Back-End Service: Flask API

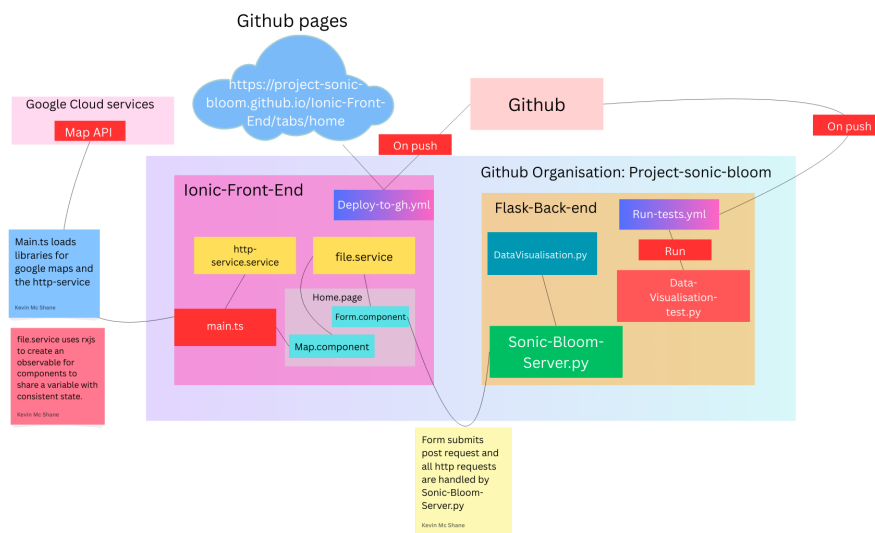


Figure 4.1: System architecture: Relationships between components

## 4.2 Component Coupling Strategy

The system employs loose coupling throughout the code-base by:

- Using REST API contracts such as Front-end to back-end communication through API endpoints.
- Dependency injection of Angular services like HTTP client and the embedded map.
- API key security API keys are kept secret by using github secrets. Asynchronous Processing: Long-running map rendering jobs use background workers.

### 4.2.1 REST APIs

Representational State Transfer (REST) "is a guideline of software architecture design constraints that bring about efficient, reliable and scalable distributed systems".[13]

This is a modern client server communication style that is very popular and widely applicable to many projects, including this one.

### 4.2.2 Platform as a service deployment: Github pages

Platform as a Service (PaaS) is a term used to describe a type of software hosted in the cloud and how it is utilized by a developer.

Github pages is an example of PaaS as it is a cloud platform that allows developers to host their static HTML.

## 4.3 Front-End Architecture

### 4.3.1 App type

This project uses ionic angular's standalone component structure with a tabbed routing template.

### 4.3.2 Important files

#### **main.ts**

responsible for loading in the app and it's dependencies (Google maps API, http-ProviderMethod).

### **home.page.html**

The main view available to the user, this renders the map and form components.

### **map.component.ts**

The code responsible for storing positional data of markers, the main logic for the google maps API.

### **form.component.ts**

The code responsible for listing the files received and implementing the http service in order to communicate with the back-end.

### **app.routes.ts and tabs.routes.ts**

Establishes the routing for the app.

### **http.service.ts**

A helpful package that helps generate and send all kinds of HTTP methods.

### **file.service.ts**

The service for collecting all of the user data into a manageable interface that can be shared with the rest of the project.

## **4.3.3 Pages**

- The home page, where the main functionality of the app is held.
- The forums page, it was supposed to contain a page where users of the app could share their findings. However due to development time this feature was never implemented.

## **4.3.4 Components**

The homepage was split into 2 components, one for the forms and another for the map.

## 4.4 Back-End Architecture

Find all Back-End related code in <https://github.com/Project-Sonic-Bloom/Flask-Back-End>

The back-end leveraged Flask for its lightweight design, which aligned with the project's need for rapid iteration. The logic for the REST API is held in the Sonic-Bloom-Server.py file.

Once a http request is received in Sonic-Bloom-Server.py the task of processing the request is delegated to the files held in the ./scripts/main folder.

Code tests for the files in ./scripts/main was initially planned to be held in ./scripts/test, which is set to run automatically using the .github/workflows/run-tests.yml script.

The idea for the system structure was for API communication to be completely separated from the more complex functionality of the server's code,

## 4.5 Architectural Trade-offs

### 4.5.1 Front-end tradeoffs

Angular's opinionated nature for example strict dependency injection and modular architecture improved maintainability but introduced complexity in inter-component communication.

Compared to React's loose state management or Vue's reactive properties, Angular requires explicit services or state management libraries for example NgRx to share variables between components.

This trade-off manifested itself in the map state logic, where the file dropped in the map or the form had to be stored in the same variable across the Map Component and Form Component a task that required a shared service.

Although this imposed separation of concerns, it added upfront development overhead.

# Chapter 5

## System Evaluation

### 5.1 Project goals evaluation

#### 5.1.1 Scope: what has been implemented

Goal	Status
Data visualization	incomplete
Deployment of front-end	complete
Deployment of back-end	incomplete
Two-tier system architecture set up	complete
Automated tests running	25% complete

Table 5.1: Scope progress

### 5.2 Software testing

#### 5.2.1 Automated tests: progression status explained

The 25% of progress attributed to the goal is attributed to the fact that the github workflow is set up to run the test script, the only thing required for the test logic to work is writing some code tests to be run.

selenium could also be used to test the front end code by using testing the build that was deployed to <https://project-sonic-bloom.github.io/Ionic-Front-End/tabs/home>

### 5.2.2 Manual testing

Due to most of the project's development time being devoted to the development of the ionic angular app, the front-end code is well tested so no additional automated testing is required for the front-end application.

## 5.3 Project weaknesses and limitations

### 5.3.1 Project management: schedule creep

Time developing the app could have been used more efficiently, for example instead of setting up google cloud for the google maps API key immediately a personal use API key for an iframe embedding was attempted to be used first.

### 5.3.2 Reasoning for this pitfall: The progress stop surrounded by google maps

#### Attempts to bypass the payed google cloud API key

In the early stages of the project when the main plan was to use the free API key, the plan was to use a html canvas to draw markers where the user dropped their file, however this would cause errors when the user moves the map and the point does not move with the map's view.

As discussed in the tech review the maps state was supposed to be given to the back-end via a screenshot on the client side, however the library responsible couldn't capture the contents of the map due to browser content policy restrictions, this made the only alternative screenshot-ting the map on the server side via selenium and restoring the maps state into the headless selenium browser (selenium headless browser explained in chapter 3: tech review). This restoration requires the variables stored in the map.

When the page loads the map it has a default center point and zoom setting, as the user moves the maps view this alters those variables.

This is an issue as the state of the map is reset when reloaded which means to communicate to the backend, the structure of the free embedded map does not contain these variables and google has very cleverly prohibited this content.

At this point it was clear that despite the limitations of the 90 day free trial API it provided the functionality required to solve the state access issues.

After that the html canvas drawn markers not moving with the map was solved as the html canvas plan was substituted to use the google maps API to draw markers.

## Agile process gaps

The project management methodology of weekly iterative developments that was once followed began to stray from it's principals as there was a failure to course correct due to the accumulation of this scope creep

### 5.3.3 Timeline creation functionality

By the time the project proposal was written, There was a plan to implement some bio-acoustics technology into the project in the form of the timeline, optimistically the audio would have been processed to somehow locate the direction and distance of the source of the sound. If this were the case an accurate heatmap could be drawn of the actual bat locations.

This was before the data was analyzed and the implementation of the project was thought out, I have since realized this type of timeline creation was unrealistic.

## 5.4 Lessons learned

When developing a feature such as the google maps API, take the path of least resistance and gain proper access to the tools required. Even when there is associated limitations and costs to the project, it is worth it as the time spent trying to develop your own solution will be taken away from other important features.

Planning a project can be exciting but it is important to maintain realistic expectations about implementation, things to be realistic about include:

- The scope of the project as if you include too much or too little you may over or under deliver on content.
- The resources available to you.
- Wether a code implementation will work.

Another lesson learned is that even when working alone taking the effort to set up and use proper project management ideologies could be well worth it.



# Chapter 6

## Conclusion

### 6.1 Summary of Context and Objectives

Project Sonic Bloom aimed to fill a necessary critical gap in ecological research tools. The lack of automated, intuitive visualization of bio-acoustic data.

Existing software like Kaleidoscope is perfect for analyzing audio recordings to auto identify an animal's species but poor at visualizing the whereabouts of these events without user intervention paired with programming skills.

This project attempted to bridge that gap by developing a web application that takes CSV outputs from Kaleidoscope and produces a group of images representing the different events happening over time.

#### 6.1.1 The primary objectives

- Automating the creation of chronological event timelines from user data.
- Deploying a two-tier client-server application with a front-end (Ionic Angular) and back-end (Flask).
- Strong system architecture and automated testing deployment.

The project was designed to reduce manual effort for researchers so that they could focus on ecological insights rather than data manipulation.

### 6.2 Key Findings from System Evaluation

The evaluation of Sonic Bloom revealed both accomplishments and challenges:

### 6.2.1 Implemented Features

Front-end deployment onto GitHub Pages.

Two-layer architecture with integration of REST API.

Integration with Google Maps API for placing the marker and map state management.

### 6.2.2 Unfulfilled Goals

Half-finished back-end deployment and data visualization programming.

Partial fulfillment of automated tests (25% done).

Rendering of timeline and heatmap features never came to pass.

### 6.2.3 Problems

Iterative planning was disrupted by consistent technical hold-ups, which delayed scope adjustments.

### 6.2.4 Lessons Learned

Early establishment of high-risk items, prevents delays.

Realistic scoping and planning for resources are critical to academic timelines.

Automated testing and documentation, while time-intensive, prevent long-term technical debt.

## 6.3 Serendipitous Insights

Despite setbacks and the outcome of the goals, the project yielded interesting serendipitous insights that are good to know going forward.

### 6.3.1 Technology Trade-offs

Comparative analysis of Flask vs. Node.js and TypeScript vs. JavaScript highlighted Python's flexibility to rapid prototyping and TypeScript's lead on code quality.

### 6.3.2 Third-Party API Realities

The limitations of using the free (non-google cloud) Google Maps embedding indicated the necessity of using paid APIs for features even on educational projects.

### 6.3.3 Agile Flexibility

While loosely based Agile styled project management can sometimes be good enough for single person development projects, formal tools such as JIRA would encourage responsibility in team settings and ensure project success if the project management tool is used correctly.

## 6.4 Possible future developments for this project

### 6.4.1 Back-End Completion

Host the Flask server at Heroku/AWS and finish data processing pipelines.  
Errors returned after the `generateHeatmap()` http listener would be fixed.  
Utilize the CI/CD pipeline set up.

### 6.4.2 Community Features

Add the forums page to facilitate user collaboration.

## 6.5 Closing Remarks

Project Sonic Bloom, although incomplete, demonstrates the promise in joining ecological data with modern technology even more. The challenges that were encountered particularly in API integration and scope management offer lessons to be learned for future academic and business endeavors.

Through balancing ambition and pragmatism, the future development could transform Sonic Bloom into an essential piece of software for ecological scientists that propels conservation data collection ahead.

# Chapter 7

## Appendix

### 7.1 Github organization link

<https://github.com/Project-Sonic-Bloom>

### 7.2 Github ionic front end link

<https://github.com/Project-Sonic-Bloom/Ionic-Front-End>

### 7.3 Github flask back end link

<https://github.com/Project-Sonic-Bloom/Flask-Back-End>

### 7.4 Github pages ionic deployment link

<https://project-sonic-bloom.github.io/Ionic-Front-End/tabs/home>

### 7.5 Screen-cast link

<https://youtu.be/f2i0Uocrjt0>

# Bibliography

- [1] Brian A. Malloy and James F. Power. Quantifying the transition from python 2 to 3: An empirical study of python applications. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 314–323, 2017.
- [2] Planeks Team. Why use flask framework for web development in 2023, 2023.
- [3] Kai Lei, Yining Ma, and Zhi Tan. Performance comparison and evaluation of web development technologies in php, python, and node.js. In *2014 IEEE 17th International Conference on Computational Science and Engineering*, pages 661–668, 2014.
- [4] Joshua D. Scarsbrook, Mark Utting, and Ryan K. L. Ko. Typescript’s evolution: An analysis of feature adoption over time. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pages 109–114, 2023.
- [5] Justus Bogner and Manuel Merkel. To type or not to type? a systematic comparison of the software quality of javascript and typescript applications on github. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, pages 658–669, 2022.
- [6] Jelica Cincović and Marija Punt. Comparison: Angular vs. react vs. vue. which framework is the best choice? *Zdravković, M., Konjović, Z., Trajanović, M.(Eds.) ICIST 2020 Proceedings*, pages 250–255, 2020.
- [7] Ted Hunter and Steven Porter. *Google Cloud Platform for developers: build highly scalable cloud solutions with the power of Google Cloud Platform*. Packt Publishing Ltd, 2018.
- [8] Niklas von Herten. html2canvas, 2020. JavaScript HTML renderer.
- [9] Angular Team. Rxjs library. Angular v17 Documentation.

- [10] Muriel Ritsch, Tom Eulenfeld, Kevin Lamkiewicz, Andreas Schoen, Friedemann Weber, Martin Hölzer, and Manja Marz. Endogenous bornavirus-like elements in bats: Evolutionary insights from the conserved riboviral l-gene in microbats and its antisense transcription in *myotis daubentonii*. *Viruses*, 16(8):1210, 2024.
- [11] pandas.DataFrame.resample. 2025.
- [12] Mr Geoffrey Mwamba Nyabuto, V Mony, and Samuel Mbugua. Architectural review of client-server models. *International journal of scientific research and engineering trends*, 10(1):139–143, 2024.
- [13] Mozilla Developer Network. REST.