**Trackr Package Research 2023**

**1. Introduction.**

Track is an open-source data visualization tool developed by students in University of Manitoba to provide easy access to data that is being formulated by Arduinos and RaspberryPis. The project is created as a web application that uses React for the front end, and GoLang for the backend. The idea of this research is to provide insightful ideas on how to implement the communication of Trackr with other programing languages using abstraction. The reason why this research is important is the possible scalability of Trackr, as well as ease of use and implementation in various tech stacks by providing possible solutions to the problem of connecting Trackr to other programming languages.

**2. RESTful API with HTTP**

RESTful API with HTTP endpoints is technique that is used for building APIs that can be accessed with different programming languages. It is already implemented in Trackr and can be used for communicating with other languages outside of Trackr scope. The frontend code contains multiple static methods that make HTTP requests using the Axios library to interact with a backend API. HTTP methods like POST, GET, PUT, and DELETE are used to create, read, update, and delete resources. Other files have other methods, but the logic is the same. *Figure 1* shows the code for VisualizationsAPI, which is used to communicate actions for visualizations for Trackr. There are files similar to this that are used to communicate actions to other parts of the Trackr application.

Other API techniques also exist other than just REST APIs, which are described below. There are multiple reasons why the future team would implement other types of APIs in addition to the current RESTful API. For example, there are advantages that other techniques have that RESTful API lack, such as using a GraphQL API which allows for larger data retrieval per API call, which drastically reduces the amount of API calls needed.

Text

Description automatically generated

Figure 1

**3. Layer of Abstraction**

One of the possible ways to provide access to the Trackr API is to create a class or a module in each programming language that abstracts the interaction with the Trackr API. Users can then import and instantiate this class or module with their API token and call various methods to interact with different functionalities of the Trackr API, such as creating organizations, projects, or adding submitting data to projects. An example of the pattern using JavaScript is provided in *Figure 2*. Before attempting to implement this technique, it’s important to carefully design and thoroughly test language-specific libraries or packages to ensure they are reliable, secure, and provide a seamless experience for developers who want to use Trackr API from different programming languages.

Text

Description automatically generated

Figure 2

**4. GraphQL**

GraphQL is another popular approach for building APIs that can be accessed from different programming languages. It allows clients to specify the exact data they need and reduces over-fetching and under-fetching of data.

To implement a GraphQL API for Trackr API, we can use a GraphQL server library such as Apollo Server or GraphQL-Go. We would need to define the GraphQL schema for Trackr API. The schema defines the types and fields available in API and their relationships. We would also need to implement the resolver functions that correspond to the fields in the schema. The resolver functions implement the actual logic for retrieving or mutating data in Trackr API.

As was mentioned above, GraphQL has an advantage of larger data retrieval per API request, which reduces the amount of API calls needed for the Trackr app. Additionally, GraphQL APIs are easier to maintain, as changes to the schema can be made without breaking backward compatibility.

**5. Language specific SDKs**

Language-specific SDKs are libraries or modules that are designed to help developers easily interact with APIs. SDKs provide functions or classes that abstract away the details of making HTTP requests and parsing responses, making it easier to work with APIs in different programming languages.

To create language-specific SDKs for the Trackr API, we could create a library or module for each programming language that wraps the functionality of making HTTP requests to the Trackr API and handling the responses. *Figure 3* shows sample code that is a python SDK that could be expanded and implemented into handling the entire Trackr REST API. Different functionality/SDK versions would be required for the GraphQL API.

Text

Description automatically generated

Figure

This method could be used to create SDKs for multiple coding languages using appropriate HTTP request libraries and language specific syntax. The benefit of using this SDK method to provide package development is allowing users to invoke methods from these classes and to not have to worry about handling network errors, or other API-specific details regarding the Trackr APIs.

**6. Conclusion**

In conclusion, there are multiple ways to connect to Trackr to other programming languages. By implementing these techniques, Trackr can be easily integrated with various tech stacks, making it more scalable and accessible to developers. All techniques are viable, and all have their advantages, so consideration of all of these discussed techniques is important before the implementation phase.