## Project Overview

The primary objective of the Seamless Data Exchange API is to enable secure, standardized, and real-time access to statistical data published by the RAK Statistics Center. Instead of navigating the website manually, authorized users such as ministries, universities, and businesses will be able to retrieve targeted datasets and reports directly via API calls. This solution addresses the inefficiencies of traditional data-sharing methods and helps unify data access across institutional "islands."

## Use Case Summary

The API is designed for read-only access. Users will issue a simple GET command such as GET /info to receive structured datasets or statistical summaries, with no ability to alter or upload data. The API will be general meaning that any user can use it to access the center’s data. For example, a university would access a Specific dataset using the API with a command like “/data/education” to get a dataset about education.

## Types of Data to Be Exchanged

The API will serve different categories of data, including:

* **Statistical Reports** – Usually in Excel or PDF format
* **Raw Datasets** – Stored as CSV files or JSON objects
* **Metadata** – Structured descriptors to contextualize datasets

Data is mostly non-PII but may still require access control due to its strategic value or sensitivity. The access frequency will vary—some data may be requested monthly, others on-demand.

## Technology Stack

After reviewing multiple options, the following tech stack is selected based on flexibility, implementation simplicity, and community support:

* **Backend Framework:** Flask (Python) – lightweight and scalable for RESTful API development.
* **Database:** PostgreSQL – to store a wide variety of data types in it, even vectorized PDF files will be loaded without errors.
* **Web Scraping Tool:** Selenium – used to automate extraction from the existing RAK statistics website where needed.
* **API Documentation:** Swagger/OpenAPI – to generate interactive and accessible documentation for external developers and partners.
* **Authentication:** API keys using OpenAPI specifications – to restrict access to trusted users (e.g., government and institutional partners).

## Initial API Specifications

The endpoint would enter a command like (/api/v1/info) this would trigger the GET method, then there is an authentication that would be required which is the API key, the input format is URL params, meaning that the user would input these commands in the URL parameter, and the API would then deliver to the user the information required. So, the output format would either be a CSV file, a PDF, or a JSON object. This would help the user retrieve specific datasets or reports. In order to set permissions for the API to only allow specific users to access it, we first issue a unique API key to the trusted entity, this could be done by this command “x-api-key: YOUR\_API\_KEY\_HERE” If the key is missing, invalid, or not authorized for the requested dataset, the request is rejected with a 403 Forbidden. Then we would define the authorized entities and their API keys in a python dictionary or a PostgreSQL collection declaring the API key, role, and the datasets that each entity can access. The backend will enforce access controls to ensure only authorized partners can query sensitive datasets.

## Version Control

A GitHub repository has been set up to manage project code and documentation:

* README.md: Project summary and setup instructions
* /api: Source code for the Flask API
* /docs: API requirements and OpenAPI schema files
* /data: Placeholder folder for sample datasets and reports