Dataset of government-developed OS software

Ilma Jaganajec TU Delft 4911253

Angelos-Ermis Mangos TU Delft 5578337

Marvin Blommestijn TU Delft 5406019

Pravesha Ramsundersingh TU Delft 4834534

Abstract

Sustainability in software engineering is crucial when it comes to reducing the environmental footprint of our society. Therefore, creating more efficient and long-lasting software plays a key role in achieving this goal. Governments keep a lot of their software open-source, promoting transparency together with collaboration and showing their commitment to accessibility, accountability, and long-term sustainability. However, government open-source projects often lack accessibility, since datasets are often not available, there exist language barriers and often an absence of incentives for broad usage. These reasons make it hard for researchers to assess the sustainability of these government software projects. For this reason, this project proposes a dataset based on government-developed open-source software to make it more accessible for researchers researching sustainability practices of government software. The dataset has a focus on sustainability metrics based on development history, buildability and presence of documentation. Additionally, we propose a dashboard to visualize various repository metrics of government software projects, providing insights into the overall health, progress, and sustainability of their software projects. By exploring different open-source projects from multiple different countries, this dataset and dashboard provide insights into the sustainability of government software, making improvements in sustainability practices more accessible to researchers and governments.

1 Introduction

Our society becomes more dependent on digital infrastructure, making the environmental and social impacts of software development under more scrutiny. Recent digitalization strategies by the European Union and individual member states have emphasized the importance of reusability and transparency in public sector software systems ([1]). Additionally, there have been regulations from the European Union such as the Interoperable Europe Act which mandate the reuse and sharing of open-source digital solutions among public administrations ([2]). For this reason, there is a strong need for sustainability in software engineering practices, as it is crucial for minimizing our environmental carbon footprint and to ensure that our systems remain relevant and usable in the long-term. Therefore, Government-developed software plays an important role in promoting sustainable software engineering practices, since it can support sustainability through methods such as open-source development. Moreover, it can also avoid duplication of efforts in tasks and support long-term maintenance. Unfortunately, there is currently a big gap in understanding the sustainability practices of these government open source projects. This reduces the ability to evaluate the long-term impact of these projects in its effectiveness, and areas for improvement.

Several challenges make it hard to study the software practices of government-developed software. In the first place, there is a lack of accessible datasets that show insights into government development practices. Moreover, there are often language barriers since projects are documented in different languages, limiting global collaboration and understandability of projects. This language barrier also leads to a lack of incentives to adopt and reuse government-developed projects, since the documentation is often unclear and not accessible. Furthermore, without a clear picture of the current state of these projects, it is hard for developers and policymakers to identify areas of improvement to adjust their practices to a more sustainable manner. For these reasons, it is difficult to assess the sustainability practices within government software projects, increasing the need for clear, centralized data from which insights can be easily extracted.

In this project we aim to address these challenges by providing a centralized dataset that contains sustainability metrics of government repositories across different countries, making it easier for researchers studying the sustainability practices within government software. Additionally, we provide a dashboard that contains insights derived from these metrics, allowing governments to view their strengths and weaknesses in terms of sustainable software engineering practices. By providing this dataset together with the dashboard, we aim to promote more sustainable software engineering practices within government software projects, encouraging governments to not only build functional software but also software that is sustainable in the long term.

2 Methodology

2.1 Data Collection

We collected repositories from government organizations across five countries: the United States, the Netherlands, Greece, Germany, and France. The repositories were sourced from GitHub and GitLab using the respective platforms' APIs. In total, 1299 repositories were gathered, reflecting a broad spectrum of projects that are important for public administration efficiency, transparency, and citizen engagement. These repositories were selected based on use-cases and significance. Examples include the source code for the Netherlands' DigiD app [3] for secure digital identity management that is used by millions of users, the USA's centers for disease control and prevention (CDCgov [4]), Germany's Corona Warn App [5] for public health monitoring, France's udata-gouvfr [6] for open data access, and Greece's open-gov for transparent governance [7] to name a few. By focusing on such a variety of projects, we capture a dataset that tries to represent the real-world use cases as best as possible.

2.2 Sustainability Evaluation Framework

In order to compare the sustainability between the five countries, we evaluated the repositories of each country according to five dimensions of sustainability using the sustainability Awareness Framework (SusAF) [8].

2.2.1 Technical

The technical dimension tries to give an indication of the resilience of the software. Ideally we want to find patterns that could indicate potential areas of improvement in terms of documentation, modularity adaptability, etc. Our technical sustainability assessment is built on a two-step process: first, we systematically gather raw repository data from government-developed open-source software, and then we apply a suite of automated analyses to compute detailed technical metrics.

For Data Collection:

1. **Repository Identification**: We start by reading a curated CSV file that lists government repositories along with key metadata such as country, organization, and repository URL. This ensures our dataset covers multiple countries which includes the United States, the Netherlands, Greece, Germany, and France.
2. **Repository Cloning**: Each repository is cloned locally using Git. This step enables us to inspect the complete codebase, including all source files and supporting documents.
3. **File Extraction**: Using a targeted search based on file extensions (e.g., Python, JavaScript, Java, etc), we collect all relevant code files while excluding non-code directories (such as vendor libraries or build directories). This extraction is crucial for subsequent code analysis.

For Metric Computation and Analysis:

1. **Sustainability Scoring via AI**: A sample of code is aggregated and submitted to an AI model (Gemini 2.0 Flash) that returns a comprehensive set of sustainability scores. These scores cover dimensions such as documentation quality, testing robustness, modularity and design, error handling, security practices, scalability potential, environmental efficiency, and social inclusiveness. Additionally, the model highlights critical issues and offers improvement suggestions.
2. **Test Coverage Analysis**: By searching for test files and test patterns, the script measures the test-to-code ratio and detects the usage of common testing frameworks. This helps us evaluate the robustness of the testing practices in place.
3. **Dependency and Structure Evaluation**: We analyze dependency management by parsing configuration files (such as 'package.json' or 'requirements.txt') and examine repository structure by checking for the presence of essential files (like README, license, CI configurations, etc). The structural score is derived from the availability and quality of these components.
4. **Code Pattern Analysis**: Using regular expressions, our system scans the code for sustainable practices (such as well-documented comments, comprehensive test cases, modular design, and effective error handling) and flags unsustainable

patterns (like hard-coded credentials or excessive code smells).

For Visualization and Reporting:

1. **Radar Charts**: These illustrate the overall sustainability performance of each country by displaying an aggregation of key metrics.

2. **Bar and Pie charts**: Detailed views of code quality, complexity, test coverage, and repository structure are provided, allowing users to pinpoint strengths and weaknesses in the technical sustainability of the repositories.

This process is end-to-end, as we go from cloning the repos and parsing code files to applying AI-driven analysis, providing a sufficient framework and view of how government-developed open-source software measures up against modern technical sustainability standards.

2.2.2 Environmental

Environmental sustainability involves minimizing ecological impact through many factors such as resource efficiency, carbon footprint and total energy consumption. Our methodology is designed to capture both the dynamic and static aspects of a repository's energy consumption and its subsequent environmental impact. At a dynamic level, we assess the energy efficiency of runnable repositories by executing them locally and measuring real-time energy use with CodeCarbon [9] that additonally estimate the associated carbon emissions compared to profilers like Energibridge [10]. Runnability is determined by evaluating the dependency files we detected in the technical analysis. Dockerfiles, tech stack files (requirements.txt, package.json make files etc), or README instructions gave an indication of potentially runnable repositories. Profiling all repositories was impractical due to time intensive execution, dependency setup, and cloning, so instead we selected the five most promising repository projects for each country according to a runability score. This score is calculated by integrating several factors: the presence of dependency management files, the inclusion of test suites (and their ratio relative to the code), measured complexity within acceptable bounds, commit frequency indicating active maintenance (i.e. used in real life), code modularity derived from sustainable coding practices, environmental efficiency as assessed by Gemini scores, and adherence to size constraints. On the static side, our methodology integrates a language-based energy efficiency estimation. We leverage the findings from [11], which provides energy consumption coefficients per various programming languages. This approach assumes that the language's inherent energy efficiency, as captured by its file size-energy mapping, serves as an indirect indicator of potential energy consumption. For example, languages like C or C++ generally exhibit lower energy consumption per megabyte compared to higher-level languages such as Python.

With this indirect indicator we hope to achieve the same results as with our measurements and validate our proposed solution.

2.2.3 Economical:

To address the economical dimension of sustainability, this project aims to identify potential redundancies and similarities between government-developed open-source software repositories across different countries. By clustering and comparing repositories based on various metrics such as community engagement, popularity, and development activity, our goal is to detect overlapping projects that may be performing similar tasks independently. Reducing redundancy through this approach can potentially decrease the economic costs associated with software development, such as the number of developers required, server storage, and maintenance expenses. Countries can use previous implementations of others to improve efficiency, collaboration, and overall sustainability in their software engineering practices. This methodology may also uncover potential overlapping categories of software projects, such as public health, COVID-19 tracking systems, and data management tools, highlighting areas where resources could be combined or improved.

2.2.4 Social

The social aspect of software engineering within government software projects has its primary focus on the social dynamics within the developers within a project. Here we aimed to assess how developers interact and collaborate through evaluations of sentiment and inclusiveness. By analyzing GitHub comments, we can gather insights into the social dimension of a software project. This includes characteristics such as contributor diversity, community practices and communication patterns. Moreover, we can assess how welcoming, engaged, and sustainable these government developers' communities are from a social perspective.

2.2.5 Individual

The individual dimension of software engineering focuses on the personal and professional sustainability of contributors involved in government open-source software projects. Here we evaluate indicators such as engagement levels and community openness to collaboration in order to understand the social health of these projects for individual developers. We provide metrics that help to assess if government project developers are welcoming to contributors, and if they are empowered to participate consistently and maintain the software over time, ultimately leading to a stronger and more sustainable ecosystem.

2.3 Data Visualization

The sustainability dimensions and corresponding metrics were visualized using an interactive dashboard built with Streamlit. This dashboard provides a concise yet comprehensive overview of the five evaluated sustainability dimensions—Technical, Environmental, Economical, Social, and Individual—through intuitive visual representations such as radar charts, bar graphs, pie charts, and treemaps mentioned in 3.2.1. Users can interactively filter the data by country or organization, enabling targeted insights into repository performance and highlighting differences and similarities between countries. By presenting sustainability metrics clearly, our users could gain insights into the strengths and weaknesses of the various OSS projects. The data that is visualized is our proposed dataset that could also be found on our github [12] to use for other purposes.

3 Results

3.1 Technical

Our technical sustainability assessment evaluated government repositories across five countries (the United States, the Netherlands, Greece, Germany, and France) using multiple metrics related to code quality, maintainability, and development practices. The analysis provides valuable insights into the technical sustainability of government-developed open-source software.

Figure 6 presents hexagon radar charts for each country (which is an aggregation of all the more sophisticated metrics), visualizing their performance across key technical sustainability metrics. These visualizations reveal distinct patterns in how different governments approach technical sustainability in their software development practices.

We can see from the radar charts that The USA and the Netherlands have the most balanced metrics, with Germany have the least balanced overall. The Netherlands seem to excel in documentation quality as in average, their repositories have the highest amount of documenting while the other countries lag behind. All countries also have low testing standards with the USA leading the group (with the highest).

Below in figure 7 you can see in more detail just the technical metrics.

Greece and the Netherlands have the highest overall metrics in the technical dimensions, with Germany having the lowest overall. All the codebases in the dataset seem to be highly modularized, which is good in terms of maintainability.

The sustainability scores for each country are summarized in Table 2.

Country	Score
France	62.7
Germany	53.3
Greece	60.0
The Netherlands	66.7
The USA	63.7

Table 1: Sustainability scores for each country

It is important to note that these results might not paint the full picture in terms of OSS for each country (specially Germany). There are many organizations that work under the governments and provide OSS but we have not included all of them, as computing these metrics takes time and the organizations don't all have a common api to scrape (or clone) the repositories from. This means that a lot don't host their repos in Github or Gitlab and making a custom solution for all of them induces a heavy overhead on our workload which is beyond the scope of the project. Instead we provide the methods, scipts and toolchain we used to compute our metrics, that can be used to extend the dataset significantly, for someone with more computational resources.

3.2 Environmental

For the dynamic assessment, CodeCarbon gave the following result to monitor the average power consumption and corresponding carbon emissions during repository execution. Figure 1 shows the average total power (in Watts) and average emissions (in kg CO_2) per country.

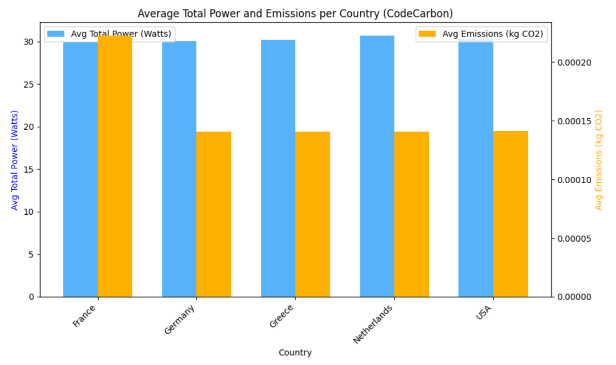


Figure 1: Measuring consumption with CodeCarbon

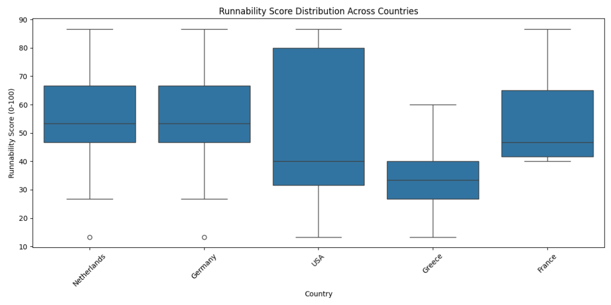


Figure 3: Potential runnability of repositories

Our static analysis estimates the energy footprint of repositories by mapping programming languages to predetermined energy coefficients based on[11]. Figure 2 presents a stacked bar chart that breaks down the assumed energy consumption by programming language for each country. This figure reveals that Governments should take into account what programming language to use as can be seen that it can dramatically impact your energy consumption.

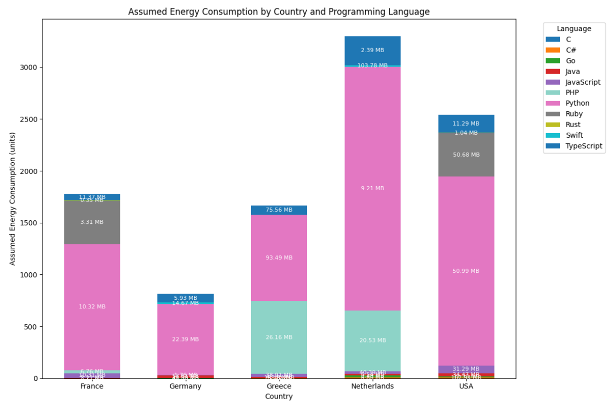


Figure 2: Static analysis

Additionally, Figure 3 illustrates the potential runnability of the repositories by country. Approximately 45% of the projects were identified as runnable based on the presence of indicators in the dependency files, whereas the remaining 55% lacked such clear indicators.

Because 1 did not show clear patterns or indications to compare on we decided to also via the GeminiFlash 2.0 evaluate the Enviromental impact of the github OSS projects. We let the model estimate this dimension by metrics like repository size, larger sizes suggest higher storage energy costs, dependency counts, more dependencies imply greater installation and runtime resource use. We also categorized code into three dimensions: high computation (e.g., ML training, GPU usage, infinite loops), which signals high energy demands; resource efficiency (e.g., generators, streaming, lazy evaluation), which reduces resource waste and thus energy use; and energy awareness (e.g., power management, sleep modes), which reflects intentional energy-saving design. These static indicators complement CodeCarbon's dynamic data, offering a fuller picture of each repository's environmental impact. It gave the following results. Note that a higher score in our analysis indicates better performance which is slightly in line with our rough estimates (france is the most consuming in 1). At first sight, this seems to contradict Figure 2; however, a closer examination reveals that although the Netherlands exhibits the highest volume—and, therefore, the highest consumption—it could be argued that the Netherlands is relatively more efficient, as demonstrated by Table 2.

Country	**Score**
France	70.33
Germany	71.667
Greece	75.1677
The Netherlands	76
The USA	73.69

Table 2: Sustainability scores for each country using Gemini Flash 2.0

3.3 Economical

To address the economical dimension of sustainability, we implemented a two-step approach involving clustering analysis and textual similarity detection to identify potential redundancies within government-developed open-source software repositories. Clustering analysis was applied to group projects based on various numerical metrics such as community engagement (number of contributors), popularity (number of stars), development activity (commit frequency), and collaboration metrics (merged pull request percentage, external contributions). The aim of this clustering process is to detect overlapping projects that may be performing similar tasks independently, thereby highlighting areas where development efforts could be consolidated. In addition to this, clustering allows us to categorize projects with similar characteristics, making it easier to identify which projects are more active, popular, or collaborative, and which may be obsolete or in need of improvement.

Complementary to clustering, we implemented a textual similarity detection process using TF-IDF vectorization to analyze the content-based similarities between repositories. By comparing repository names and descriptions, we generated a similarity matrix that identifies repositories with overlapping purposes across different countries. This approach highlights potential redundancy by revealing projects with similar functionalities that may be developed independently, often due to language barriers, lack of accessibility, or unawareness of existing solutions. Moreover, this similarity analysis can uncover broader categories of overlapping projects, such as public health, COVID-19 tracking systems, or data management tools. These findings can be used to promote collaboration between countries, reduce economic costs associated with development and maintenance, and improve the overall sustainability of government software engineering practices.

The clustering results in Figure 4 reveal interesting patterns in the distribution of projects across countries. Most repositories are populated within the range of 0 to 100 stars and 0 to 100 contributors, indicating that the majority of projects have limited popularity and medium-sized development teams. Germany stands out with its repositories spreading further across the x-axis (number of stars), suggesting that German projects tend to attract higher public attention, even though the number of contributors remains relatively equal to other governments (below 100). In addition to this, France has one outlier repository with a significantly high number of contributors but a low amount of stars (0-100), which may indicate a project with heavy internal government development but limited public recognition or adoption. These results highlight potential inefficiencies where projects with high development efforts are not widely used or where popular projects are not adequately maintained, providing opportunities for improving sustainability through collaboration or better resource allocation.

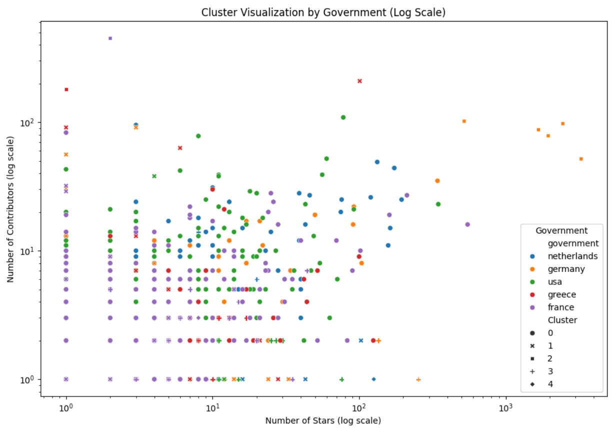


Figure 4: Cluster Visualization by Government

Based on the similarity matrix, we will analyze repository pairs with similarity scores above 0.5 as shown in Figure 5 to identify projects that may be addressing similar objectives. This involves manually reviewing repository titles and descriptions to detect overlapping themes and categories. While scraping or thoroughly tracking all the code within these repositories is not feasible within the current time constraints, examining similarities through names and descriptions provides a practical starting point. This approach will help uncover potential redundancies and common areas of interest between countries, highlighting opportunities for collaboration or consolidation of efforts.

The similarity analysis revealed several categories of repositories that appear to address similar goals or stem from shared origins. A recurring theme is that many projects represent variations of the same software, tailored to different systems or use cases — such as portals, templates, or design frameworks — often within the same government body. For example, the Dutch repositories ***MinBZK/design-system*** and ***MinBZK/NL-design-system*** show high similarity (0.87), as do ***MinBZK/mijn-bureau*** and ***MinBZK/mijn-bureau-portal*** (0.86), suggesting internal reuse of design elements or architecture. Another clear theme is COVID-19–related applications, including notification apps (***minvws/nl-covid19-notification-app-android*** and ***corona-***

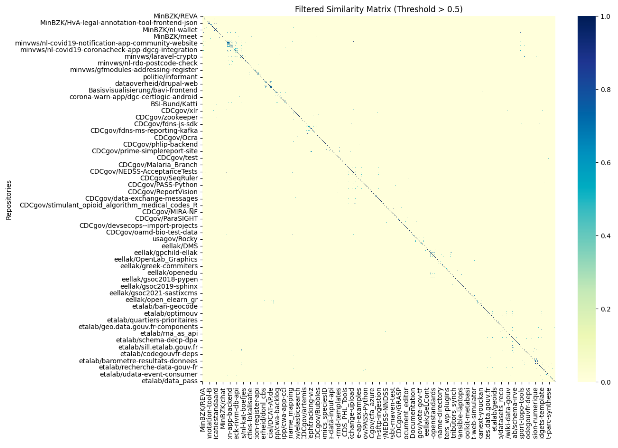


Figure 5: Filtered Similarity Matrix where the Threshold is above 0.5

warn-app/cwa-app-android, 0.54) and dashboards (***minvws/nl-covid19-data-dashboard*** and ***etalab/covid19-dashboard***, 0.76), which reflect how different governments independently developed tools for similar pandemic-related needs. Technical overlaps also emerged, such as extensions for the CKAN data platform (***dataoverheid/ckanext-donl*** and ***etalab/ckanext-etalab***, 0.51), and country-specific adaptations of the European open data standard DCAT-AP (***dataoverheid/DCAT-AP-NL*** and ***GovDataOfficial/DCAT-AP.de***, 0.84). These findings demonstrate both parallel efforts and opportunities for alignment or code reuse, pointing to potential efficiency gains through greater awareness and collaboration across government open-source ecosystems.

3.4 Social

The social side of software engineering refers to interactions and collaborative behavior between developers working on a software development project. Good communication and collaboration within software development practices increase cohesion between developers and productivity, which, therefore, leads to more success and sustainability of open-source communities. Investigating the tone and inclusiveness of developers within repositories can yield insights into how well a community is fostering collaboration within its project.

To assess the social well-being of software projects within government open-source development projects, we performed a sentiment analysis on commit comments from the GitHub repositories of these governments of different countries. This approach is based on the methodology of the paper by [13], where *SentiStrength* was used to evaluate the sentiment polarity of developer communication. The *SentiStrength* approach assigns sentiment scores to GitHub communication, with values ranging from highly negative to highly positive. In this way, it allows us to examine the social aspects of the development processes of governments, quantify the emotional tone of the repositories, and thus assess social well-being.

To measure sentiment, we extracted commit comments from the repositories and calculated the following metrics for both country-level aggregates and individual repositories:

- **Average Sentiment:** Mean emotional tone; higher values indicate more positive language in communication.
- **Positive and Neutral Ratios:** The proportion of comments classified as positive or neutral, providing insight into the overall mood of communication.
- **Negative Sentiment:** Indicates minimal conflicts.

Additionally, we complemented the sentiment data with social indicators derived from repository metadata:

- **Social Inclusiveness:** Reflects the proportion of contributors outside the core team.
- **Community Engagement:** Measures the activity of the community and interaction over time.
- **Diversity Index:** Quantifies contributor variability.
- **Contributor Retention Rate:** Indicates how consistent developers are in being active within the project.
- **Governance Indicators:** Binary flags for the presence of a Code of Conduct and a Contribution Guide.

Table 3 demonstrates the aggregated sentiment and social health metrics by country. The results show that the Netherlands, together with Germany, scored highly on average sentiment. Moreover, these countries performed better than others on social indicators such as inclusiveness and engagement. However, the United States showed the highest positive sentiment ratio but had lower inclusiveness and contributor retention.

3.5 Individual

To ensure that government-developed software can achieve long-term impact, it must demonstrate individual sustainability. Individual sustainability focuses on the health and well-being of the developer community around a software project, ensuring that contributors can engage meaningfully, be easily incorporated, and maintain the project over time. To make it easier to assess how different government software projects perform in terms of individual sustainability, we collected various metrics from the GitHub repositories of the projects for our dataset. Although these are technical attributes, they serve as indicators for community health, openness, contributor experience, and long-term participation.

Table 4 in the appendix outlines the metrics we extracted from the government repositories, together with their interpretation from an individual sustainability perspective. 5 Shows the evaluation on our repositories.

We examined these metrics for multiple repositories across various countries, aiming to map the technical state of the government repositories. Additionally, we aimed to demonstrate patterns in individual sustainability, such as how accessible, collaborative, and maintainable these systems are for contributors over time. With these insights, governments can more easily understand the strengths and weaknesses of their software project practices. Identifying areas of improvement will be simplified, together with promoting a more resilient and contributor-friendly open-source ecosystem. Moreover, this dataset gives researchers access to structured and clear data supporting individual sustainability studies in government open-source software, enabling deeper analysis of developer experience, policy impacts, and community health across nations.

4 Discussion

4.1 Empirical Analysis

This analysis provides a structured overview of how different governments approach sustainable software development and where improvements can be made. It has been split into technical, environmental, economical, social, and individual analysis.

4.1.1 Technical

The technical sustainability analysis revealed significant differences in how governments maintain and develop their open-source projects. Countries like the Netherlands and the USA exhibited more balanced performance across key technical metrics, suggesting relatively mature and consistent development practices. In contrast, Germany's repositories displayed less balanced scores, possibly reflecting inconsistencies in practices or varying levels of maintenance across its projects. Documentation quality was notably higher for the Netherlands, indicating better support for understandability and long-term maintainability. Testing coverage, however, was generally low across all countries — with the USA leading but still falling short — highlighting an area for substantial improvement. Modularization was a positive finding across the board, suggesting that most projects are structured in a way that supports easier updates and future scalability.

4.1.2 Environmental

From an environmental sustainability perspective, several indicators were extracted to evaluate potential energy consumption and runtime efficiency. Around 45% of repositories were identified as potentially runnable, which leaves room for improvement in terms of reproducibility and ease of deployment. Furthermore, approximately 55% of the repositories had no identifiable tech stack, creating uncertainty around environmental impact due to unclear deployment environments and possible inefficiencies in resource use. Among those that did expose a tech stack, Docker-based setups were observed, which suggest leaner operational models via containerization. These setups can reduce energy consumption by optimizing resource isolation. Additionally, repositories developed in Go or C indicated a preference for low-overhead and performance-efficient technologies — an environmentally conscious choice when compared to heavier runtimes.

4.1.3 Economical

To address the economical dimension of sustainability, we implemented both clustering and textual similarity analysis to identify potentially redundant projects. Clustering grouped repositories by development activity, popularity, and collaboration metrics, helping us distinguish between active, successful projects and those that may be outdated or underutilized. This revealed clusters where multiple projects across countries shared similar engagement levels, indicating areas of overlap that could be streamlined. The similarity matrix analysis further revealed textual overlaps in repository names and descriptions, which pointed to functional redundancies. For example, COVID-19 dashboards and notification apps were developed independently by several governments, while reusable design systems and CKAN extensions appeared in both Dutch and French repositories. These overlaps highlight opportunities to reduce economic costs by encouraging cross-country collaboration, reusing templates, or consolidating efforts where functionality already exists.

4.1.4 Individual

Individual sustainability was assessed by examining community-centric metrics that reflect how open, accessible, and collaborative the development process is for contributors. By analyzing data like the number of contributors, external contributions, and pull request activity, we were able to form a picture of the health and openness of these government projects. While some repositories showed signs of active contribution, many lacked reuse instructions or contribution guidelines, indicating that the primary goal was transparency rather than community building. This creates barriers for onboarding new developers and sustaining long-term maintenance through an engaged open-source community. The dataset developed in this project not only highlights these trends but also provides a foundation for further research on developer experience and community health across nations, helping governments strengthen contributor support and create more resilient projects.

4.2 Limitations & Future work

A key limitation in our technical analysis is that the majority of repositories lacked clear reuse instructions, often stating that their code was open-sourced primarily for transparency rather than reproducibility, an example [3] [14]. This limits the ability to generalize the evaluation of how maintainable or extensible the software actually is in practice. Therefore the results of the environmental measurements are not inline with expectations. In addition to this, some government projects are large scale systems — such as national databases or applications serving millions of users — which are difficult to assess accurately due to their size and complexity. These projects often include extensive dependencies or distributed components that span multiple repositories, making it difficult to perform a one-to-one comparison across countries. This is particularly evident in domains like COVID-19 tracking, where some countries bundle everything into one repo, while others split functionality across many. Combined with limited documentation and the absence of standardized deployment methods, it becomes challenging to reproduce runtime environments or generalize findings across all government software.

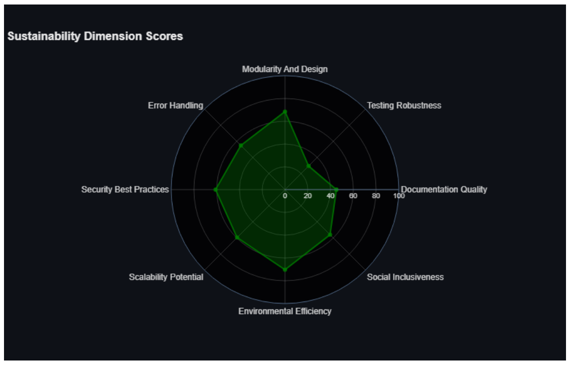
In terms of economical sustainability, limitations arose from differences in how countries name and describe their repositories. Language barriers, inconsistent use of acronyms, and country-specific naming conventions made it difficult to consistently interpret and compare repository purposes. While our textual similarity approach identified overlapping project names and descriptions, it was not feasible within the scope of this project to dive deeper into code-level comparisons. Manual inspection of repository contents was time-consuming, and a deeper analysis — such as checking for duplicated codebases — was out of scope due to time constraints. Future work could involve advanced code similarity analysis or even clone detection to uncover deeper redundancies, assuming privacy and licensing conditions allow.

Lastly, in terms of social sustainability, there is a limitation in SentiStrenght's ability to detect technical language or sarcasm, leading to potential inaccuracies in emotional tone classification. Moreover, the evaluation relies primarily on commit messages within the government GitHub repositories which might not capture the full scope of developer interactions which might be more visible in pull requests, or external communication channels.

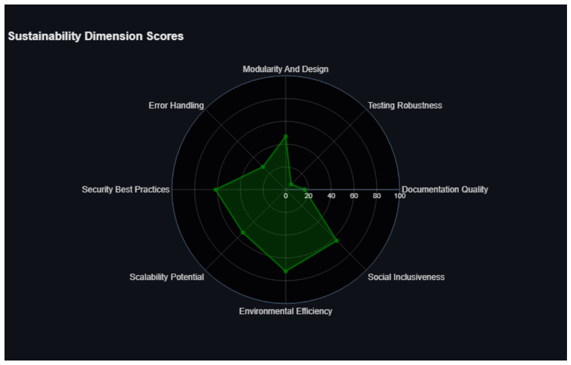
5 Conclusion

In this project, we approached the challenge of evaluating government-developed open-source software through the lens of multiple sustainability dimensions — technical, environmental, economical, social, and individual. By combining metric-based analysis with clustering and similarity detection, we explored how governments can better collaborate, reduce redundancy, and improve long-term software practices. Our dataset [12] and methodology lay a foundation for future research, enabling deeper exploration into sustainable software development in the public sector. For users that are interested in using our dashboard can find the application here [15].

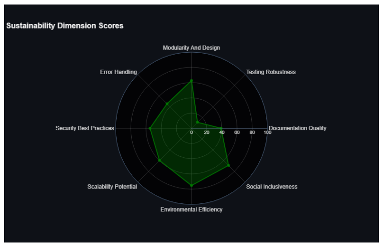
A Technical Metric Graphs



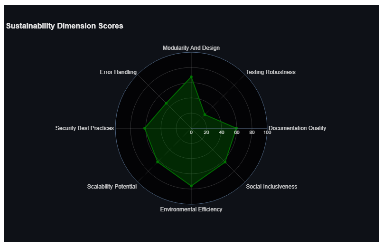
(a) France



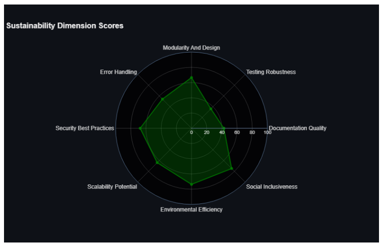
(b) Germany



(c) Greece

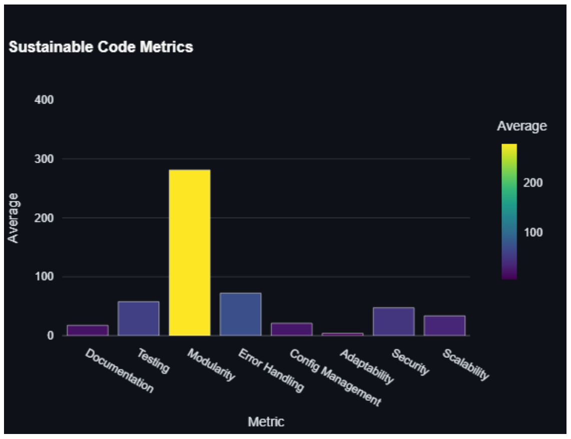


(d) The Netherlands

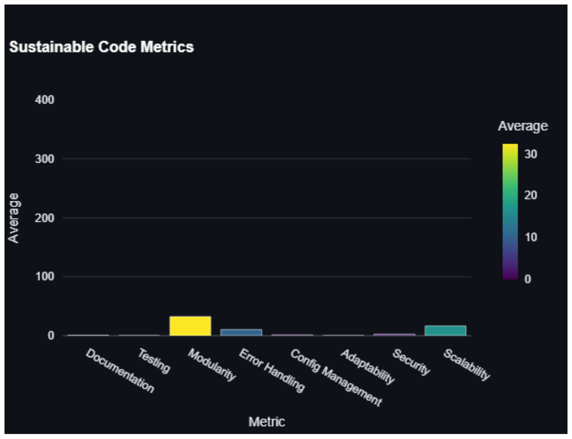


(e) The USA

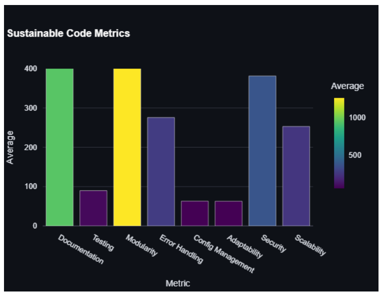
Figure 6: Hexagon radar charts for individual countries



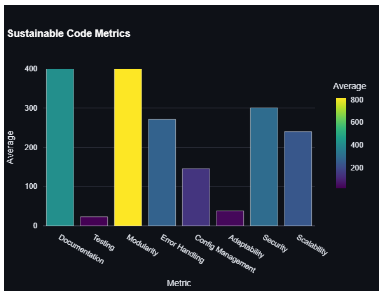
(a) France



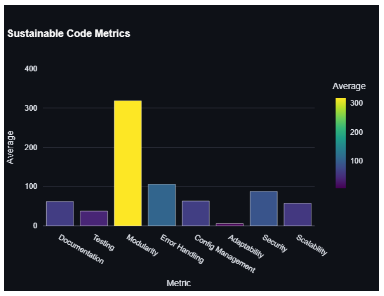
(b) Germany



(c) Greece



(d) The Netherlands



(e) The USA

Figure 7: Technical metrics graphs for individual countries

B Social Dimension Metrics by Country

Country	Avg Sentiment	Pos Ratio	Neutral Ratio	Contributors	Inclusiveness	Engagement	Diversity	Retention (%)	Code of Conduct	Contrib. Guide
France	0.010	0.005	0.728	141	41.833	0.224	0.364	58.737	No	No
Germany	0.019	0.009	0.824	1199	56.333	0.461	0.708	55.602	Yes	Yes
Greece	0.006	0.002	0.731	81	46.167	0.035	0.319	56.563	No	No
Netherlands	0.021	0.010	0.857	326	57.167	1.417	0.727	59.409	Yes	Yes
USA	0.024	0.012	0.388	134	30.667	0.160	0.264	30.610	No	Yes

Table 3: Aggregated sentiment and social health metrics by country

C Individual Dimension Metrics

Collected Metric	CSV Metric Name(s)	Individual Dimension Relevance
Development history	`total_commits`, `repo_age_months`	Indicates openness, transparency, and activeness of the repository; essential for understanding project evolution and sustained engagement.
Number and diversity of contributors	`num_contributors`, `external_pr_percentage`	Reflects whether development is centralized or open to outside contributors; diversity supports resilience and long-term sustainability.
Commit activity / most recent commit	`commit_frequency_per_month`, `days_since_last_commit`	Shows if the project is actively maintained and whether contributors are regularly engaged.
Documentation (README, contributing guide, etc.)	`has_readme`, `has_contributing`, `has_license`	Helps new contributors onboard easily; accessible documentation improves developer experience and community inclusiveness.
Buildability of the software	`has_cicd`	Ensures that contributions can be reliably tested and integrated, reducing friction in the development process.
Open issues and pull requests	`open_issues`, `closed_issues`, `merged_pr_percentage`	Reflects openness to external input, feedback loops, and community collaboration on the project.

Table 4: Individual sustainability-related metrics extracted from government repositories

Country	Total Commits	Commit Frequency Per Month	Has Readme	Has Contribution Guide	Has Code of Conduct
France	198.166667	4.012504	0.433333	0.033333	0.000000
Germany	208.833333	41.042321	1.000000	0.166667	0.166667
Greece	16.733333	1.207077	0.900000	0.000000	0.000000
Netherlands	29.733333	0.868840	0.733333	0.200000	0.000000
USA	814.521739	9.645142	0.739130	0.217391	0.000000

Table 5: Aggregated individual dimension metrics by country.

References

[1] European Commission. *Open Source Software Strategy.* Accessed April 4, 2025. 2020. URL: https://commission.europa.eu/about/departments-and-executive-agencies/digital-services/open-source-software-strategy_en.

[2] European Commission. *Interoperable Europe Act: Implications and Impact on EU's Digital Future.* Accessed April 4, 2025. 2024. URL: https://data.europa.eu/en/news-events/news/interoperable-europe-act-implications-and-impact-eus-digital-future.

[3] MinBZK. *woo-besluit-broncode-digid.* URL: https://github.com/MinBZK/woo-besluit-broncode-digid.

[4] Centers for Disease Control and Prevention (CDC). *CDCgov GitHub Repositories.* URL: https://github.com/CDCgov.

[5] Corona-Warn-App Project. *Corona-Warn-App.* URL: https://github.com/corona-warn-app.

[6] Etalab. *Etalab GitHub Repositories.* URL: https://github.com/etalab.

[7] EELLAK. *EELLAK GitHub Repositories.* URL: https://github.com/eellak.

[8] SUSO Academy. *Sustainability Awareness Framework (SusAF).* URL: https://www.suso.academy/en/sustainability-awareness-framework-susaf/.

[9] mlco2. URL: https://github.com/mlco2/codecarbon?tab=readme-ov-file.

[10] Tu Delft. URL: https://github.com/tdurieux/EnergiBridge.

[11] Sustainable Computing Lab. *Energy Efficiency of Programming Languages: An Analysis.* URL: https://sites.google.com/view/energy-efficiency-languages.

[12] Group 9. URL: https://github.com/IlmaJaganjac/project_2_group_9.

[13] Emitza Guzman, David Azócar, and Yang Li. *Sentiment analysis of commit comments in GitHub: An empirical study.* 11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings. DOI: 10.1145/2597073.2597118. 2014. URL: https://doi.org/10.1145/2597073.2597118.

[14] open overheid. URL: https://open.overheid.nl/documenten/530f2237-4168-44b7-ac41-461ced7a8977/file.

[15] Group 9. URL: https://project2group9-t4ammavu3jyeihygjyelua.streamlit.app/.