SoilEye: An Interactive Web App Prototype for Soil Sealing Monitoring

Ahmad Asad[1,2], Alessandra Capolupo[2(✉)], and Eufemia Tarantino[2]

[1] Department of Agricultural and Environmental Sciences, University of Bari Aldo Moro, Bari, Italy
[2] Department of Civil, Environmental, Land, Construction and Chemistry (DICATECh), Politecnico di Bari, Via Orabona 4, 70125 Bari, Italy
alessandra.capolupo@poliba.it

Abstract. Climate change problems are intensifying rapidly, making it essential to monitor environmental and geospatial issues in real time. With the advent of cloud computing, researchers and experts can develop more sustainable solutions in a shorter timeframe. In the geospatial field, the introduction of Google Earth Engine has been a game changer. As a powerful cloud computing platform, Google Earth Engine enables large-scale geospatial analysis by leveraging extensive public data and open-source geospatial datasets. One of the most impactful features introduced by such a tool is creating a web-based application, known as Google Earth Engine Apps. These Apps allow users to visualize, modify, share, and interact with their analysis, results, and outputs in real time. This paper focuses on the creation of "SoilEye", an innovative Google Earth Engine App designed to observe soil sealing patterns over the metropolitan area of Bari. It provides a detailed, step-by-step methodology for developing the app, highlighting its potential and functionality. This app tracks land use changes over approximately six years, from 2018 to 2024, by combining satellite images collected by the Sentinel 2 mission with other geospatial data. It identifies places that have been impacted by soil sealing and evaluates its trend over time. As a result, this App will serve as a valuable tool for policymakers, offering an interactive platform to monitor soil sealing dynamics continuously. This will aid the sustainable urban environment management and help planners make more informed decisions.

Keywords: Cloud Computing Platform · Google Earth Engine (GEE) · Urban Sustainability · Big Data for Environmental Monitoring · Impervious Surfaces

1 Introduction

Urbanization, by altering natural ecosystems, has become one of the biggest challenges and obstacles to sustainable development [1–3], contributing significantly to climate change and global warming. Indeed, the soil ecosystem plays a vital role in regulating temperature and providing resilience against climate change [4, 5]. However, the increasing spread of impervious surfaces, particularly in coastal areas where this phenomenon is more intense [6], has led to the degradation of natural habitats and soil properties [7].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15891, pp. 207–221, 2026.
https://doi.org/10.1007/978-3-031-97617-9_14

The United Nations' (UN) Sustainable Development Goal 13 enhances the urgent need for actions against climate change, advocating for increased awareness, and developing innovative solutions to mitigate and adapt to its effects [8]. Additionally, the UN is making a lot of effort to face the soil consumption issue, by defining many goals directly or indirectly linked with soil consumption, including SDG 2, 3, 11, 12, and 15. Indeed, protecting soil cover and creating a sustainable balance between natural habitats and anthropogenic activities are their main topics [9]. Therefore, it appears necessary to detect effective and efficient methods to extract land use/cover information and monitor its changes over time.

Recently, advancements in remote sensing technologies have revolutionized the way Earth's dynamics are observed, making large-scale monitoring more efficient and accurate [10]. Open-source satellite missions, such as Sentinel, Landsat, and Moderate Resolution Imaging Spectroradiometer (MODIS), continuously provide high-quality images of the Earth's surface and atmosphere [11–13]. However, tracking land changes requires processing vast amounts of data, which exceeds the computational capacity of traditional desktop systems. This constraint was recently overcome thanks to the introduction of cloud-based environments, like Google Earth Engine (GEE) (https://earthengine.google.org). This powerful tool, indeed, takes advantage of many processors in parallel to running custom algorithms, developed both in Python and JavaScript languages in its Application Programming Interface (API) [14]. Additionally, it allows speeding up the procedure thanks to the integrated data catalogue, daily updated with 6000 freely available datasets.

A new opportunity concerning App creation has recently been provided by the GEE environment. Such an App allows users to interactively showcase their analysis and results performed in the GEE API, and to disseminate them among policymakers and the public. Users can interact with simple front-end User Interface (UI) features, while the backend implementation is handled by a developer [15] using both JavaScript and Python coding in accordance with their expertise. Thus, the backend development is performed in the GEE code editor directly, where all possible analyses, from geospatial analysis and visualization up to UI design and dynamic text integration, are implemented. This approach enables customization of the UI based on specific needs, such as adding logos, tailored functions, and other elements to enhance its appeal and usability [16]. Additionally, visibility settings could be adjusted to restrict access to a Google group or make the app public. When GEE web app access is set to "public", anyone with the app's link can view it. GEE also provides "access without signing into Google," allowing any users to visit it without a Google account. This significantly saves time and reduces "login-barriers", making it easier for policymakers, researchers, and experts to go through the produced results included in the App. Although there are no widely known uses of the GEE platform for soil sealing, several apps cover related themes such as urban growth and impervious surface mapping. For instance, the UrbanWatch 1m Land Cover and Land Use GEE app, in which developers used a fine-resolution, large-area Urban Thematic Information Extraction (FLUTE) framework, contains an extensive and large amount of data of 211 GB, which was reduced to 4.54 GB for GEE [17]. Another example closely linked to urban and non urban land monitoring using GEE is by [18], who used three decade Landsat data to monitor LULC in Brazil. Outside of GEE, few

specialized WebGIS platforms visualize and detect soil sealing. For instance, the Urban Atlas and the Copernicus Land Monitoring Service (CLMS) provide high-resolution layers of artificial surfaces across Europe, as these platforms offer valuable insights into urban sprawl [19] and Copernicus created High Resolution Layer Imperviousness service providing different datasets on the imperviousness density and built-up areas [20].

This study aims to design and implement "SoilEye", a novel interactive application leveraging GEE as a cloud computing platform to detect and monitor soil sealing in the Puglia Region (Southern Italy). The SoilEye framework was built on a custom JavaScript code within the GEE environment, integrating remote sensing data, specifically Sentinel 2 images, with geospatial analysis tools. Crucially, the application has been carefully optimized for both desktop and mobile platforms, ensuring full functionality and a consistent, user-friendly experience across devices. This cross-platform compatibility greatly enhances accessibility, enabling users to access insights anytime and anywhere, whether from an office workstation or directly in the field. By detecting the temporal trend of soil sealing from 2018 up to 2024, this approach supports data-driven decision-making for sustainable land management.

The paper is structured as follows: i) Sect. 2 describes the methodology adopted to design and implement SoilEye app as well as the datasets and the analytical methods used to extract soil sealing information and evaluate its trend over time in the metropolitan area of Bari; ii) Sect. 3, instead, presents and discusses the outcomes, introducing the innovative, interactive App and detailing the findings concerning soil sealing trend; iii) lastly, SoilEye app strengths and weakness are outlined in the Conclusion Section.

2 Materials and Methods

This Section presents the dataset and techniques used to extract soil sealing information and assess its trends over time, as well as the approaches employed to develop the innovative, interactive "SoilEye" App. As previously mentioned, the cloud-based platform was chosen because of its capacity to handle geospatial big data and support web App creation [21–24]. Thus, GEE was used to select and process the input data, generate and visualize the findings, and create the app.

The entire workflow is summarized in Fig. 1. It consists of three primary phases: i) core GEE script development: implemented in JavaScript code editor, it involves importing freely available Sentinel 2A satellite dataset collected from the European Space Agency's (ESA), applying preprocessing techniques, and extracting soils sealing information through the integration of Sealed Urban Index (SUI) with Classification and Regression Tree (CART) Machine Learning (ML) algorithm; ii) SoilEye interface design: this phase is focused on two main steps: a) Wireframing and layout definition, aimed at defining App structural blueprint, which was useful in organizing the flow of user interaction; and, b) Optimization for multi-device compatibility, ensuring App visualization on different browsers and devices; and, lastly, iii) SoilEye App logic integration, focused on implementing the app's core functionality and optimizing its performance.

More details concerning the various steps are reported in the following paragraphs.

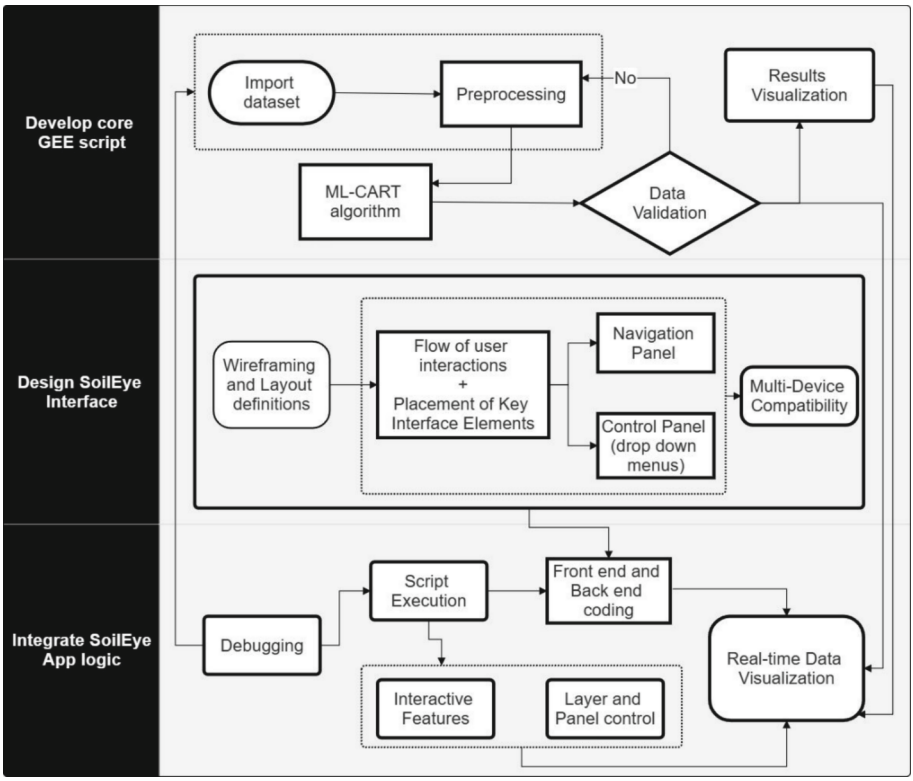

Fig. 1. GEE App creation workflow.

2.1 Development of Core GEE Script

The first step in the development of SoilEye App involved creating the core GEE script for generating soil sealing maps, essential to explore the trend of impervious surface in the metropolitan area of Bari (Fig. 2). To reach such a goal, the methodology proposed by [25] was adopted, which integrates the SUI with a ML algorithm, specifically the CART procedure. Indeed, as highlighted by [26], this combination has proved to be the most effective, yielding reliable results in investigating soil sealing dynamics from Sentinel 2A images This mission was preferred over other freely available satellite initiatives, such as Landsat and MODIS, because of its higher spatial resolution (10 m, 20 m, 60 m), frequent revisit time over the same area (5 days), and availability of more spectral bands (13).

Thus, to implement this approach, a custom JavaScript code was developed in the GEE environment to select the most appropriate dataset, compute SUI, and apply the CART algorithm.

A total of 28 Sentinel 2A images, covering the metropolitan area of Bari, were selected on the base of three primary criteria: i) study area (images had to fully cover the selected region); ii) cloud cover threshold (only images with a cloud coverage lower than 5 were considered. The only exception was reported by the image of May 2019

that features a cloud covered percentage equal to 18.34%); and, lastly, iii) timeframe (the selected images span from 2018 up to 2024 in order to ensure temporal consistency in the analysis). To satisfy these criteria, spatial, temporal, and cloud cover filters were carried out through the following instructions:

```
// Study area definition (Spatial Filter)
var studyArea = ee.FeatureCollection(bbari);
// Define the time periods for each year, specifically for February, May, July, and October
var years = [2018, 2019, 2020, 2021, 2022, 2023, 2024];
var months = ['02', '05', '07', '10'];
var timePeriods = [];
years.forEach(function(year) {
  months.forEach(function(month) {
    // Construct the start date
    var startDate = new Date(year, parseInt(month) - 1, 1);  // Months are 0-indexed in JS Date

    // Get the last day of the month by moving to the next month and subtracting 1 day
    var endDate = new Date(year, parseInt(month), 0);

    // Format dates as 'YYYY-MM-DD'
    var startDateFormatted = startDate.toISOString().split('T')[0];
    var endDateFormatted = endDate.toISOString().split('T')[0];

    timePeriods.push([startDateFormatted, endDateFormatted]);
  });
});

console.log(timePeriods);
// Loop through each time period, filter Sentinel-2 data, and merge the results
var s2Collection = ee.ImageCollection([]);
timePeriods.forEach(function(period) {
  var tempCollection = ee.ImageCollection('COPERNICUS/S2_HARMONIZED')
      .filterBounds(studyArea)
      .filterDate(period[0], period[1]);
  // Function to filter images, compute indices, classify, and return classified image
  function classifyTimePeriod(startDate, endDate, classifier) {
    var filtered = s2.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 5))
      .filter(ee.Filter.date(startDate, endDate))
      .filterBounds(studyArea);
```

Nevertheless, when necessary, cloudy pixels were cleaned out by using the algorithm proposed by [27], which relies on metadata and the Quality Assurance band (Band QA60) of Sentinel 2A. On the contrary, both atmospheric and geometrical corrections were not applied since the selected images were already satisfying from those points of view. The main features of the 28 picked-up images are reported in Table 1.

Table 1. Sentinel 2A image details.

Year	Acquisition Dates	Cloud Cover Range (%)
2018	04 Feb, 25 May, 14 Jul, 12 Oct	0.23–3.36
2019	19 Feb, 05 May, 24 Jul, 12 Oct	0–18.34
2020	09 Feb, 04 May, 01 Jul, 01 Oct	0.04–3.55
2021	03 Feb, 09 May, 01 Jul, 04 Oct	0–0.01
2022	03 Feb, 12 May, 03 Jul, 04 Oct	0–4.06
2023	01 Feb, 07 May, 03 Jul, 01 Oct	0–4.19
2024	03 Feb, 01 May, 05 Jul, 13 Oct	0–3.92

After preprocessing the input dataset, the SUI and CART algorithms were implemented. SUI was calculated through Eq. 1:

$$SUI = \frac{(Green - SWIR1) \times (SWIR2 - Red)}{(Green + SWIR1) \times (SWIR2 + Red)} + 0.5. \quad (1)$$

Such an equation was implemented in GEE environment using the following JS code snippet: var SUI = singleImage.expression(

```
        '(((Green - SWIR1) * (SWIR2 - Red)) / ((Green + SWIR1) * (SWIR2 + Red)))
+ 0.5', {
          'Green': singleImage.select('B3'),
          'Red': singleImage.select('B4'),
          'SWIR1': singleImage.select('B11'),
          'SWIR2': singleImage.select('B12')
        });
```

Subsequently, a systematic procedure was adopted to implement the CART algorithm on the basis of SUI findings and ground truths. This algorithm was implemented to classify land use and land cover (LU/LC), with a particular focus on identifying soil sealing. The classification divided the territory into four main categories: sealed surfaces, vegetation, bare soil, and water bodies. Training data were derived from authoritative LU/LC maps provided by the Istituto Superiore per la Protezione e la Ricerca Ambientale (ISPRA), which include high-resolution shapefiles annotated according to national and European standards. A stratified random sampling approach was applied to extract representative samples across land cover types. To enhance class separability and reduce overlaps among the four categories, the Otsu's method, an image thresholding technique aimed at minimizing intra-class variance, was applied, along with adaptive thresholding based on mean (μ) and standard deviation (σ) values.

The dataset was split into 70% training and 30% testing data, enabling the CART algorithm to partition the input into increasingly homogeneous subsets recursively. The model's performance was evaluated using a confusion matrix, reporting metrics such as Overall Accuracy (OA), User's and Producer's Accuracy (UA and PA), and the Kappa coefficient (K). Validation was conducted using 487 ground truth points extracted from the ISPRA maps, confirming the reliability of the classification approach and supporting robust mapping of soil sealing patterns.

Findings were visualized by using the Map.addLayer() function, which enables visual interpretation by overlaying layers.

Lastly, to reduce the number of iterations and minimize handling time, batch processing procedure was applied. It was preferred to process each image individually to reduce analysis iterations. Comprehensive comments were added before each step to improve readability, facilitate debugging, and enhance error handling.

2.2 Designing SoilEye Interface

After implementing the core functionalities, the design of the SoilEye App interface started by taking into account the requirements of the target users. The main objective of this process was to create a user-friendly interface based on usability principles, ensuring accessibility, clarity, and efficiency in data visualization [28]. The interface was structured to facilitate user interaction with soil sealing analysis results, providing an intuitive experience across different devices.

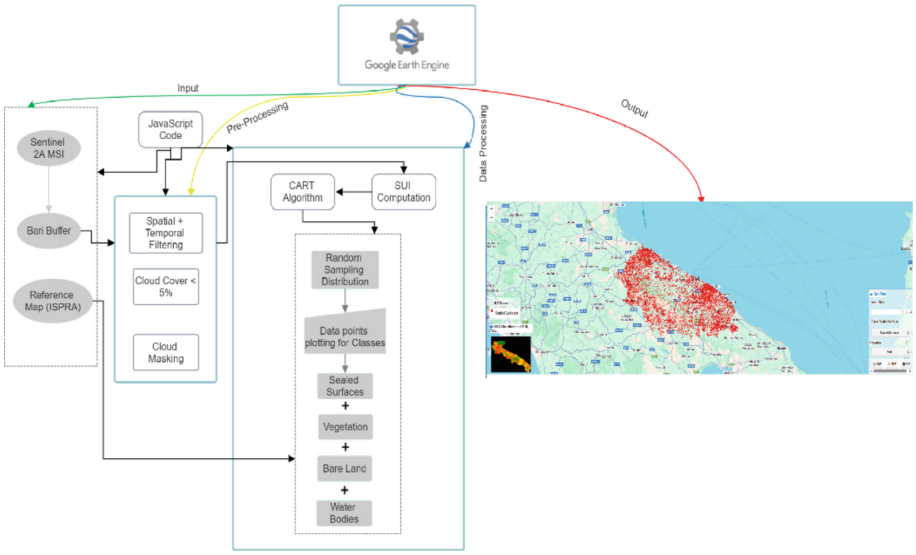

Fig. 2. Core GEE script development.

2.2.1 Wireframing and Layout Definition

The development of the GEE App followed a predefined wireframe, which was created to define the structural blueprint of the app. This served as a foundation for organizing the app's layout, ensuring that the interface components were logically placed for an intuitive user experience. In such a way, the wireframe establishes a clear structure, ensuring that Information Technology (IT) and non-IT skilled users can easily interact with the interface and allowing them to navigate the panel effortlessly and display information in the appropriate window [29].

Wireframe creation process consisted of several key aspects: i) investigating users' requirements and needs; ii) designing an initial UI layout by focusing on essential graphical and interactive tools; iii) ensuring consistency in spacing and responsiveness; and, lastly, iv) refining the design iteratively according to users' feedback [30]. Thus, this structured approach, based on the potential capabilities of integrating geospatial data processing with an intuitive user experience, was applied. According to both IT and non-IT skilled users requirements, three core elements were recognized as essential: i) visualization window (display area for maps); ii) navigation panel, allowing users to switch between functionalities; and, lastly, iii) Multi-device and multi-browser compatibility, ensuring accessibility across different platforms without structural limitations.

In the SoilEye app, the map window, strategically positioned at the center of the UI, displays real-time findings over a baseline map provided by OpenStreetMap. To its left, a control panel, with a drop-down button menu, supports users to filter data according to years and land use/cover classes, as well as to visualize accuracy metrics of the generated outcomes.

2.2.2 Optimization for Multi-device Compatibility

This step ensures GEE App compatibility across multiple devices and browsers, providing a seamless user experience. While desktop and laptop monitors offer large displays, the real challenge was optimizing the visualization on the smartphones' screens [31]. to address this, the app was built with flexible and scalable UI elements. Additionally, touch gesture support was implemented to ensure smooth interaction on touchscreens, increasing app usability on mobile devices.

Optimizing touch gesture support for multiple devices can be challenging, as key interactions, such as pinch-to-zoom, accessing panel features, and tap-to-select for feature identification, rely on precise touch input detection [32]. To provide seamless interaction, the size of the panel was adjusted to avoid obstructing the whole map while still allowing users to navigate efficiently.

2.3 Integrating SoilEye App Logic

This step is aimed at transforming SoilEye App's interface design into a fully functional interactive tool within the GEE platform, ensuring that user inputs were correctly processed and visualized. Its main objective was to run the core GEE script responsible for analysis and data processing [33], including the incorporation of datasets and the application of different filters. JS code was created for each step and analysis. The front and back-end coding facilitated the creation of the interactive interface (front-end) and handled the underlying logic and server communication (back-end) [34]. While implementing previous methods, debugging was consistently carried out to ensure the proper functioning of the GEE script, user interface, and app using the console log for troubleshooting. Thus, to prevent unexpected behaviors, such as crashes due to high memory consumption, and to ensure all steps functioned properly, a script execution was implemented [35].

To enhance usability, several interactive features were incorporated: i) real-time data visualization which immediately updates the map to the Bari study area, removing the need for users to search for it manually; ii) layer control allows users to enable or disable the different years and layers which are sealed surfaces, vegetation, bare land, and water; iii) accuracy assessment button containing OA and K values of each year which validate the results generated from each year.

2.3.1 Performance Optimization

The app was optimized by incorporating several performance optimization techniques, including asynchronous data loading. This step involved the use of asynchronous requests to fetch data from GEE, preventing UI elements from freezing during data retrieval. The optimal goal of any app is to reduce excessive computation that can cause unresponsiveness while ensuring smooth and fast responses for interactive functionalities [36, 37].

To minimize computation time, reduce memory load, and enhance responsiveness, instead of loading the entire satellite dataset, SoilEye applies strict spatial and temporal filters at the initial stage. This limits data size while maintaining results accuracy and enabling faster loading times across all devices. This optimization allows the app to

efficiently retrieve and process only the necessary satellite data, preventing it from being overwhelmed by the full dataset.

Moreover, SoilEye defers index calculations and classification until they are explicitly requested by the user, for instance, when selecting options from the dropdown menu to display results on the map. The map interface is also optimized to load only essential layers initially, preventing clutter and ensuring smooth interactions.

2.3.2 Final Adjustments and Debugging

To optimize the app, results, like maps, were checked to verify their correct alignment with expectations. Additionally, after each step, error handling was performed by reviewing console logs, which directly pointed out the issues in the script. Syntax errors were debugged, and variable names were double-checked for consistency throughout the script.

3 Results and Discussion

This SoilEye App developed for this study provides an interactive platform for analyzing soil sealing changes, implementing the CART algorithm with the SUI calculation, and visualizing geospatial data over the study area of Bari in the Puglia region. This app integrates multi-temporal Sentinel-2 imageries spanning from 2018 up to 2024 and, when necessary, employs cloud masking techniques to ensure data accuracy. It was created in the GEE environment to leverage large-scale satellite data access, cloud-based processing capabilities, and the ability to handle multi-scale and multi-platform geospatial data, while also enabling the creation of a web-based application. Notably, accessing the SoilEye WebApp does not require users to create a Google account, allowing them to explore parameters and results simply by following a link.

Both desktop and smartphone interfaces of SoilEye (see Figs. 3 and 4) are composed of various key elements, including a visualization window, a control panel with buttons, and zoom in/out controls. The visualization window is centered in the UI while zoom in/out buttons as well as the control panel are located on the left.

Figure 3 shows the SoilEye interface with the legend, which displays class colors and labels and clearly highlights sealed surfaces. Users can select classes, navigate through years, select a province, and select K and OA statistics and export results.

Figure 5 provides a detailed view of the control panel, featuring a button-up menu allowing users to select years, land cover classes, and metrics on desktop and smartphone, respectively. As a result, scrolling through the year selection allows users to visualize the resulting classified maps, which are based on the composite from February, May, August, and October of each year. Similarly, scrolling land cover classes enables users to visualize the entire map or extract specific categories among sealed surfaces, vegetation, bare land, and water bodies. Lastly, the metrics button allows users to check the accuracy of the visualized layers through the K and OA values. Moreover, two additional buttons were placed on the top of the main window to zoom in and out.

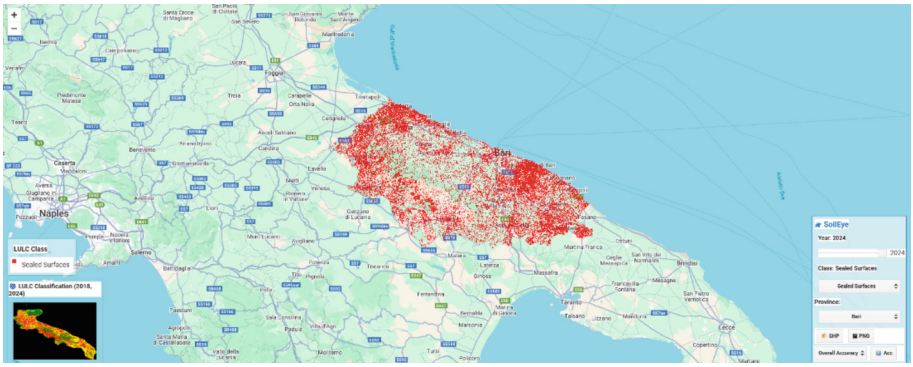

Fig. 3. SoilEye App interface visualized on desktop devices.

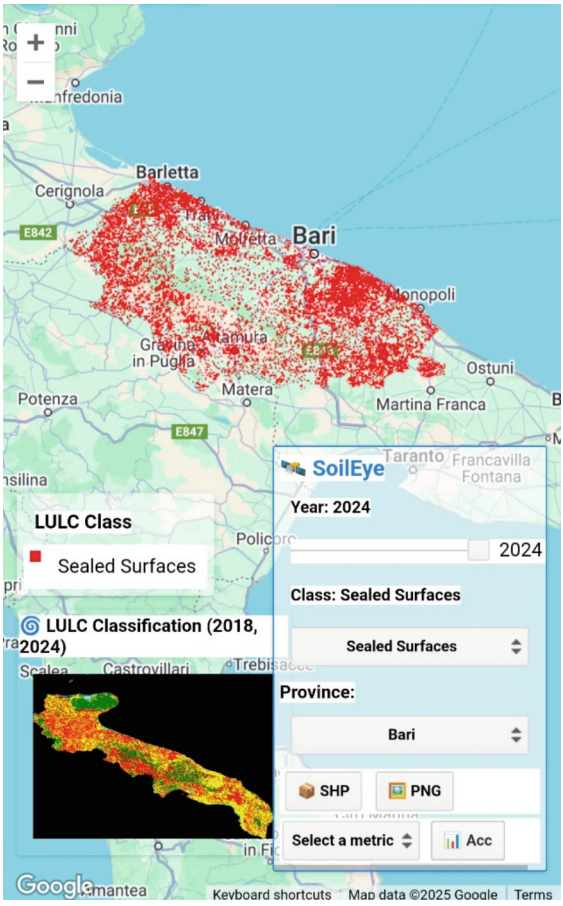

Fig. 4. SoilEye App interface visualized on smartphone devices.

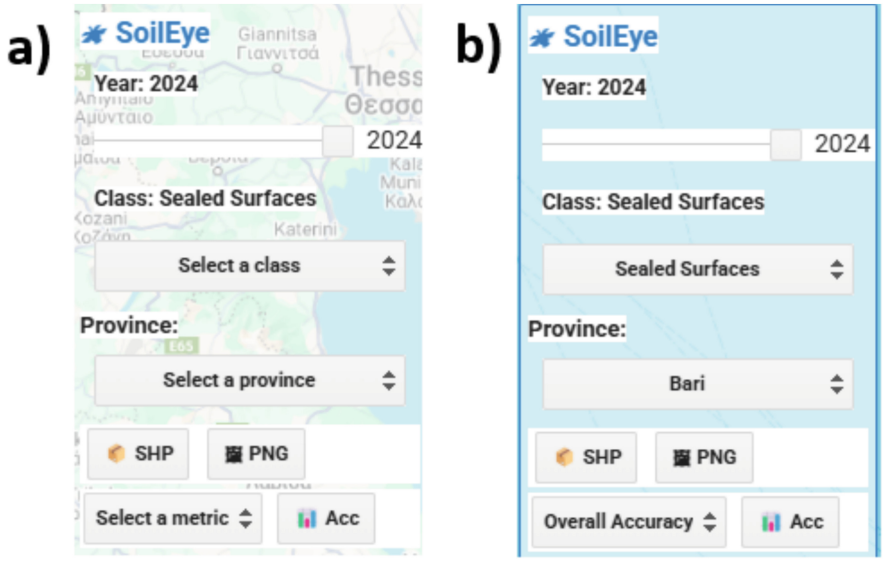

Fig. 5. Panel map with buttons for selecting specific years, land cover classes, province, export button for shapefile, png, and metrics (a). Example of the Panel map used for selecting the sealed surface class of 2024, Bari province with OA metric selection (b).

Figures 3 and 4 ensure that users can access and interact with the app seamlessly from both smartphones and desktops, regardless of their location. The layout is optimized for both screen types, allowing users to visualize results effortlessly, without unnecessary clicks. To handle computationally intensive tasks, SoilEye leverages GEE cloud computing, ensuring that heavy processing is performed remotely, preventing mobile devices from managing large computations directly.

There are several apps and platforms, like Copernicus Urban Atlas and CLMS, that provide static layers for sealed surfaces in Europe, but without using GEE. However, the previously mentioned platforms do not offer a dynamic selection of LULC classes and a province or local-level viewer focused on soil sealing trends. While apps built with the GEE platform like UrbanWatch, track urban growth using impervious surface data. The SoilEye app provides dynamic legends, class-wise selection to view each class separately, and allows users to explore urban sealing trends interactively with animation. However, the SoilEye App is still a prototype, and it has some limitations that need to be fixed. Enhancing interactivity is a key priority, which can be achieved by integrating additional features, such as time-series analysis and dynamic charts, to support users in investigating the findings more effectively. Furthermore, predictive solutions could be introduced to simulate future land cover scenarios.

Another aspect to be tested is related to its multi-scale capacity, which could expand its applicability. More features, such as a dedicated section for predicting land changes in Bari, could also be included. To further improve user interaction, a slider and auto-animation feature could be introduced, allowing users to visualize soil sealing changes dynamically within seconds. While the current app's version is ideally built for both

mobile phone and desktop screens, adding new features will require careful optimization to maintain usability across different devices. The app also has the potential to scale up to a regional/national level without significantly increasing computational demands. Finally, a downloading option could be implemented to allow users to export results, requiring the development of extra code.

4 Conclusion

Monitoring soil sealing over the past years is essential for understanding its trends, assessing its impact on natural habitat and predicting future consequences. Two main challenges should be addressed to meet such a goal: developing an effective and efficient method able to extract this information and ensuring widespread dissemination to raise awareness among policymakers and the public.

This research is aimed at facing the second issue by introducing SoilEye, an innovative interactive app prototype designed to facilitate soil sealing monitoring. This app, indeed, provides policymakers and experts with a valuable tool to detect sealed surfaces, monitor land cover changes and inform the public. By bridging the gap between complex remote sensing workflows and user-friendly data exploration, SoilEye empowers researchers and policymakers to make data-driven decisions.

Through an intuitive and interactive interface, this app successfully delivers the required results. Its temporal analysis tool allows users to seamlessly explore changes over time, while the land cover classification feature enhances usability by enabling users to visualize each class separately. However, the app still lacks dynamic features like automatic scrolling animation that could give a quick overview of changes in Bari. Alongside the animation feature, a forecast for the next few years may include the use of predictive models. Other potential enhancements include dynamic charts, time-series visualizations, and interactive slides for annual comparisons.

Despite these limitations, this app offers a scalable and efficient solution for monitoring long/short-term changes. With further improvements, it has the potential to evolve into a powerful tool at provincial or even regional.

References

1. Güneralp, B., Reba, M., Hales, B.U., Wentz, E.A., Seto, K.C.: Trends in urban land expansion, density, and land transitions from 1970 to 2010: a global synthesis. Environ. Res. Lett. **15**(4), 044015 (2020)
2. Vogler, J.B., Vukomanovic, J.: Trends in united states human footprint revealed by new spatial metrics of urbanization and per capita land change. Sustainability **13**(22), 12852 (2021)
3. Shao, Z., Ahmad, M.N., Javed, A., Islam, F., Jahangir, Z., Ahmad, I.: Expansion of urban impervious surfaces in Lahore (1993–2022) based on gee and remote sensing data. Photogramm. Eng. Remote. Sens. **89**(8), 479–486 (2023)
4. Thomas, A., et al.: Topsoil porosity prediction across habitats at large scales using environmental variables. Sci. Total Environ. **922**, 171158 (2024)
5. Criado, M., Santos-Francés, F., Martínez-Graña, A., Sánchez, Y., Merchán, L.: Multitemporal analysis of soil sealing and land use changes linked to urban expansion of Salamanca (Spain) using landsat images and soil carbon management as a mitigating tool for climate change. Remote Sens. **12**(7), 1131 (2020)

6. Isinkaralar, O., Sharifi, A., Isinkaralar, K.: Assessing spatial thermal comfort and adaptation measures for the Antalya basin under climate change scenarios. Clim. Change **177**, 118 (2024)
7. Yu, J., et al.: Environmental threats induced heavy ecological burdens on the coastal zone of the Bohai Sea China. Sci. Total Environ. **765**(15), 142694 (2021)
8. Take urgent action to combat climate change and its impacts. https://sdgs.un.org/goals/goal13. Accessed 1 March 2025
9. Mikhailova, E.A., Post, C.J., Nelson, D.G.: Integrating united nations sustainable development goals in soil science education. Soil Syst. **8**(1), 29 (2024)
10. Lupi, V., Morretta, V., Zirulia, L.: Earth Observation data, innovation and economic performance: a study of the downstream sector in Italy. Eurasian Bus. Rev. **14**, 103–136 (2024)
11. Landsat Missions. https://www.usgs.gov/landsat-missions. Accessed 2 March 2025
12. The Sentinel missions. https://www.esa.int/Applications/Observ-ing_the_Earth/Copernicus/The_Sentinel_missions. Accessed 1 March 2025
13. MODIS. https://modis.gsfc.nasa.gov/about/index.php. Accessed 3 March 2025
14. Velastegui-Montoya, A., Montalván-Burbano, N., Carrión-Mero, P., Rivera-Torres, H., Sadeck, L., Adami, M.: Google earth engine: a global analysis and future trends. Remote Sens. **15**(14), 3675 (2023)
15. Stromann, O., Nascetti, A., Yousif, O., Ban, Y.: Dimensionality reduction and feature selection for object-based land cover classification based on sentinel-1 and sentinel-2 time series using google earth engine. Remote Sens. **12**, 76 (2019)
16. Earth Engine Apps. https://developers.google.com/earth-engine/guides/apps. Accessed 27 Feb 2025
17. Zhang, Y., et al.: UrbanWatch: a 1-meter resolution land cover and land use database for 22 major cities in the United States. Remote Sens. Environ. **278**, 113106 (2022)
18. Souza, C.M., et al.: Reconstructing three decades of land use and land cover changes in brazilian biomes with landsat archive and earth engine. Remote Sens. **12**(17), 2735 (2020)
19. Copernicus Land Monitoring Service – High Resolution Layer: Imperviousness. https://land.copernicus.eu/pan-european/high-resolution-layers/imperviousness. Accessed 5 May 2025
20. High Resolution Layer Imperviousness. https://land.copernicus.eu/en/products/high-resolution-layer-imperviousness. Accessed 5 May 2025
21. Capolupo, A., et al.: An interactive WebGIS framework for coastal erosion risk management. J. Marine Sci. Eng. **9**(6), 567 (2021)
22. Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., Moore, R.: Google Earth Engine: planetary-scale geospatial analysis for everyone. Remote Sens. Environ. **202**(1), 18–27 (2017)
23. Zhao, Q., Yu, L., Li, X., Peng, D., Zhang, Y., Gong, P.: Progress and trends in the application of google earth and google earth engine. Remote Sens. **13**(18), 3778 (2021)
24. Yang, L., Driscol, J., Sarigai, S., Wu, Q., Chen, H., Lippitt, C.D.: Google earth engine and Artificial Intelligence (AI): a comprehensive review. Remote Sens. **14**(14), 3253 (2022)
25. Asad, A., Capolupo, A., Tarantino, E.: Machine learning and index-based classification approaches for accurately evaluating sealed surfaces from medium-resolution Sentinel 2 data. In: 4th International Proceedings of Mediterranean Geosciences Union. Scopus, Barcelona (2025)
26. Asad, A., Capolupo, A., Tarantino, E.: Assessing soil sealing dynamics in the metropolitan area of Bari using sentinel-2, sealed surface index, and machine learning approaches. In: Computational Science and Its Applications–ICCSA 2025: 25th International Conference Proceedings, Springer International Publishing (2025)
27. Sola, I., et al.: Assessment of atmospheric correction methods for Sentinel-2 images in Mediterranean landscapes. Int. J. Appl. Earth Obs. Geoinf. **73**, 63–76 (2018)

28. Shamsujjoha, M., Grundy, J., Li, L., Khalajzadeh, H., Lu, Q.: Developing mobile applications via model driven development: a systematic literature review. Inf. Softw. Technol. **140**, 106693 (2021)
29. Gudoniene, D., Staneviciene, E., Buksnaitis, V., Daley, N.: The scenarios of artificial intelligence and wireframes implementation in engineering education. Sustainability **15**(8), 6850 (2023)
30. Gutierrez, C., Lara, R., Subauste, D.: Cloud application for the generation of static websites through the recognition of wireframes using artificial intelligence. In: Proceedings of the International Congress on Educational and Technology in Sciences 16–18 November 2021, Chiclayo, Peru (2021)
31. Saia, S.M., Nelson, N.G., Young, S.N., Parham, S., Vandegrift, M.: Ten simple rules for researchers who want to develop web apps. PLoS Comput. Biol. **18**(1), 1009663 (2022)
32. Setiyani, L., Tjandra, E.: UI/UX design model for student complaint handling application using design thinking method (Case Study: STMIK Rosma Karawang). Int. J. Sci. Technol. Manage. **18**(3), 690–702 (2022)
33. Capolupo, A., et al.: A WebGIS prototype for visualizing and monitoring the spatio-temporal changes in seawater quality. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C. (eds) Computational Science and Its Applications – ICCSA 2022 Workshops. ICCSA 2022. LNCS, vol. 13379. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10545-6_24
34. Conti, G., Mercante, E., de Souza, G.E., Sobjak, R., Bazzi, L.C.: AGDATABOX-RS computational application: remote sensing data management. Software X. **23**, 101435 (2023)
35. Processing Environments. https://developers.google.com/earth-engine/guides/processing_environments. Accessed 5 March 2025
36. Capolupo, A., Monterisi, C., Tarantino, E.: Development of an open-source 3D webgis framework to promote cultural heritage dissemination. In: De Paolis, L.T., Arpaia, P., Sacco, M. (eds.) Extended Reality. XR Salento 2022. LNCS, vol. 13446. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15553-6_19
37. Mondejar, E.M., et al.: Digitalization to achieve sustainable development goals: steps towards a Smart Green Planet. Sci. Total Environ. **794**, 148539 (2021)