

Development of the digital observatory; Prototype and final version of the Observatory



**Eco
Ready**

Lead partner: AUTH



The ECO-READY project has received funding from the European Union's Horizon Europe Research and Innovation Programme under grant agreement n° 101084201

Technical References

Project Acronym	ECO-READY
Project Title	Achieving Ecological Resilient Dynamism for the European food system through consumer - driven policies, socio-ecological challenges, biodiversity, data-driven policy, sustainable futures
Project Coordinator	Czech University of Life Sciences (CZU)
Project Duration	48 months

Deliverable No.	D4.2 Early
Dissemination level ¹	PU
Work Package	WP4
Task	4.2
Lead beneficiary	AUTH
Contributing beneficiary(ies)	WR, CUT, WHITE
Due date of deliverable	31 January 2025
Actual submission date	31 January 2025

¹ PU = Public

PP = Restricted to other programme participants (including the Commission Services)

RE = Restricted to a group specified by the consortium (including the Commission Services)

CO = Confidential, only for members of the consortium (including the Commission Services)



Document history

Version	Date	Beneficiary	Author
0.1	5 December 2024	AUTH	Themistoklis Diamantopoulos
0.2	20 December 2024	AUTH	Themistoklis Diamantopoulos, Thanasis Michailoudis, Dimitrios-Nikitas Nastos
0.3	3 January 2025	AUTH	Themistoklis Diamantopoulos, Thanasis Michailoudis, Dimitrios-Nikitas Nastos
0.4	10 January 2025	AUTH	Themistoklis Diamantopoulos, Thanasis Michailoudis, Dimitrios-Nikitas Nastos
0.5	16 January 2025	AUTH	Themistoklis Diamantopoulos, Thanasis Michailoudis, Dimitrios-Nikitas Nastos
0.6	20 January 2025	AUTH	Themistoklis Diamantopoulos, Thanasis Michailoudis, Dimitrios-Nikitas Nastos
0.7	23 January 2025	AUTH	Themistoklis Diamantopoulos, Thanasis Michailoudis, Dimitrios-Nikitas Nastos
0.8	28 January 2025	WR	Ali Hurriyetoglu
0.9	29 January 2025	CZU	Ioannis Manikas
1.0	31 January 2025	AUTH	Themistoklis Diamantopoulos, Thanasis Michailoudis, Dimitrios-Nikitas Nastos, Andreas Symeonidis



Summary

Task 4.2 includes the development of the ECO-READY Observatory. The deliverable is a prototype, split into an early version (this version, 4.2.1) and a late version (4.2.2). The deliverable includes three modules.

The first module, described in Section 2, is the big data management infrastructure. It is available at repositories <https://github.com/AuthEceSoftEng/ecoready-observatory> and <https://github.com/AuthEceSoftEng/ecoready-services> and is described in terms of its usage (API available online at <https://ecoready-services.issel.ee.auth.gr/docs>).

The second module, described in Section 3, is the dashboard, which further includes the interactive map. The source code of the dashboard is available at two repositories (for the backend: <https://github.com/AuthEceSoftEng/ecoready-dashboard-backend> and for the frontend: <https://github.com/AuthEceSoftEng/ecoready-dashboard-frontend>), along with API documentation (<https://ecoready-dashboard.issel.ee.auth.gr/backend/api-docs/>). The dashboard itself, including the different screens is available online at the address <https://ecoready-dashboard.issel.ee.auth.gr/> and is illustrated using screenshots in this deliverable.

The third module, described in Section 4, is the mobile app, for which the source code is available at repository <https://github.com/AuthEceSoftEng/ecoready-mobile-app> that includes instructions for building the app. The apk is available at the address <https://github.com/AuthEceSoftEng/ecoready-mobile-app/releases/tag/latest>, while the section also involves screenshots and details about using the app.



List of Abbreviations

Abbreviation	Description
API	Application Programming Interface
APK	Android Application Package
EU	European Union
HTTP	HyperText Transfer Protocol
NPM	Node Package Manager
ORM	Object-Relational Mapping
RSS	Really Simple Syndication
SQL	Structured Query Language
UI	User Interface
UX	User Experience



Disclaimer

This publication has been produced within the ECO-READY project which has been funded by the European Union Horizon Europe Research and Innovation Programme under grant agreement n° 101084201. This publication reflects only the views of the authors. The European Commission and Research Executive Agency cannot be held responsible for any use which may be made of the information contained therein.



The ECO-READY project has received funding from the European Union's Horizon Europe Research and Innovation Programme under grant agreement n° 101084201

Table of contents

1. Introduction.....	9
1.1. Objectives and Scope.....	9
1.2. Document Structure	10
2. ECO-READY Infrastructure.....	11
2.1. Architecture and Implementation	11
2.1.1. Architecture and Implementation.....	11
2.1.2. Infrastructure Components and Connectivity	11
2.1.3. Replication and Fault Tolerance	13
2.2. Infrastructure Usage	14
2.2.1. Overview	15
2.2.2. Key Functionality of the API	15
2.2.3. API Integration with Infrastructure	20
2.2.4. Fault Tolerance and Scalability.....	21
2.2.5. API Security	21
2.2.6. Integration Data Sources and Living Labs	22
3. ECO-READY Dashboard and Map	23
3.1. Architecture and Implementation	23
3.2. Dashboard Usage.....	24
3.2.1. Dashboard API	24
3.2.2. Dashboard Screens.....	26
4. ECO-READY Mobile App.....	34
4.1. Architecture and Implementation	34
4.2. Mobile App Usage.....	35
Key Functionality	36
References	41



1. Introduction

1.1. Objectives and Scope

WP4 includes the design and development of the ECO-READY observatory, which involves a tool-set of different modules, i.e. an infrastructure capable of handling diverse types of evolving data and streams, a dashboard and interactive map that provide data visualizations and actionable insights around food security, and a mobile app meant to engage and inform the users. The user categories identified include: farmers-breeders, citizens, policy makers, and industry users. Using the observatory can benefit these user categories in multiple ways; when developing new policies, when applying mitigation actions or adapting to evolving climate situations or even when making decisions about everyday life.

In this context, this deliverable provides the first version of the aforementioned modules, a version that is functional and allows users to have a clear view of the data and scenarios of ECO-READY. Given its robust architecture, designed in Deliverable 4.1 of ECO-READY, the observatory can process, store and analyse different datasets and even evolving data streams. To demonstrate this, we have consumed different data from different sources (EU Agri-Food Markets Database, FAOSTAT Food and Agriculture Data) as well as from the Living Labs of the project. Other databases can also be integrated according to the needs of the project (e.g. about climate, remote sensing data, etc.), yet the Living Labs offer a major source of information for populating the ECO-READY observatory, as they produce large amounts of data targeting regional food products, which allow developing and quantifying evidence-based models, policies and climate mitigation scenarios.

This deliverable (4.2.1) is the early version of the ECO-READY observatory prototype, and thus it provides links to the repositories where the different systems are developed, as well as demo apps (in the form of online URLs that include web applications or installation files in the case of the mobile app). Specifically, the modules include first the **big data management infrastructure** that is used as a scalable data store and processing unit that can handle data analysis tasks. We also develop the **dashboard** that is a web application capable of visualizing different data and providing recommendations in the food security domain. Moreover, the dashboard allows metadata searching (searchable depository) for data present in the infrastructure, and further integrates an **interactive map**, which allows

the users to focus on different areas and bioclimatic regions. Finally, a **mobile app** is also developed, which is used as means of engaging users to inform them and raise awareness about the challenges of food security with respect to climate change and biodiversity.

Finally, we note that our implementation is functional (i.e. the main components have been developed and the tools can be executed), which is also tested using the data outlined above. The modules have been implemented according to the technical specifications of Task 4.1 (Deliverable 4.1), so they have authorization, authentication and resilience.

1.2. Document Structure

In this section, we outline how the work is presented into different sections. In specific, the rest of this deliverable is organized as follows:

- **Section 2** comprises the development of the infrastructure of ECO-READY. It involves links to the source code and describes the technologies used and the setup for the different components. Moreover, it provides information and details about using the infrastructure API.
- **Section 0** includes details about the dashboard. It involves the links to the implementation of the dashboard components, including backend, frontend, and database. It further provides details about using the dashboard including the interactive map, and, finally, provides screenshots of the different pages (EU product data, data from Living Labs, interactive map).
- **Section 4** presents the mobile app. It includes details about the implementation and execution of the app, and further provides screenshots of the different pages, including all the different components. The screens include news feeds, tips, calculators, and interactive games.



2. ECO-READY Infrastructure

2.1. Architecture and Implementation

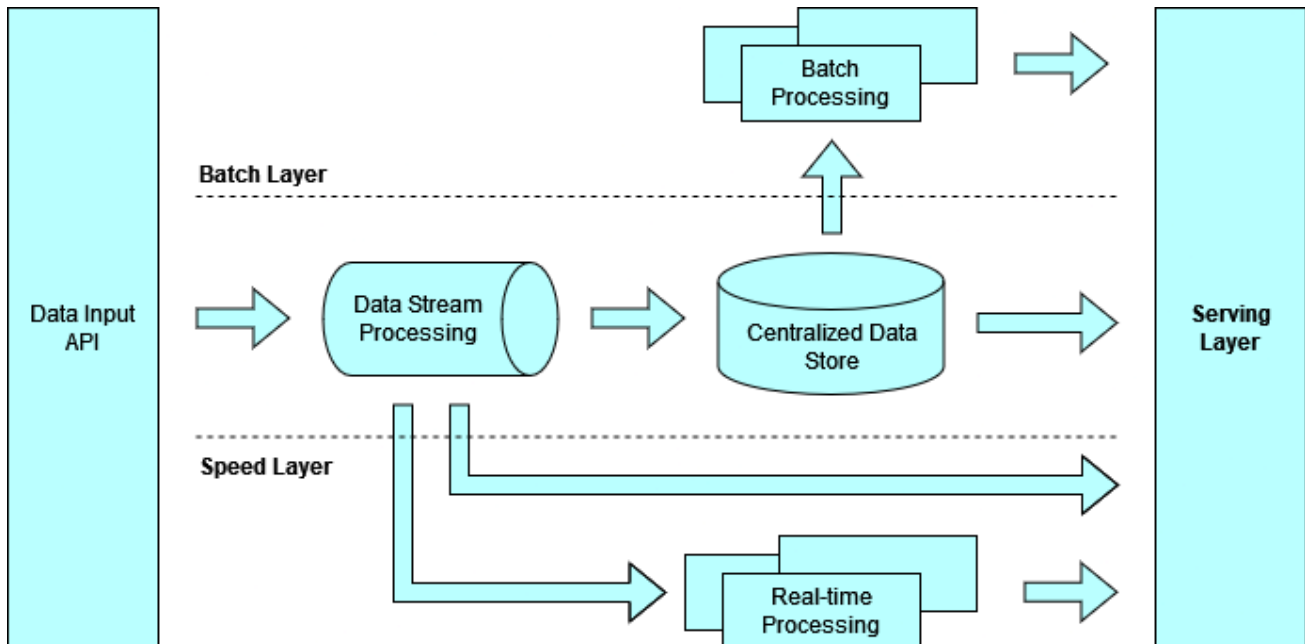


Figure 1 Architecture of ECO-READY Infrastructure

2.1.1. Architecture and Implementation

The ECO-READY infrastructure is a distributed system designed for real-time data processing, storage, and analytics. It uses open-source technologies like Kafka, Cassandra, Flink, Zookeeper, and ksqlDB to deliver a robust and fault-tolerant architecture. The system operates across **two nodes**, with services distributed to ensure high availability, scalability, and resilience. The infrastructure's GitHub repository is available at:

<https://github.com/AuthEceSoftEng/ecoready-observatory>

2.1.2. Infrastructure Components and Connectivity

The main components of the ECO-READY infrastructure are:

1. Kafka:

- a. A distributed event-streaming platform deployed as two brokers across the nodes.
- b. Kafka ensures durability and fault tolerance using topic replication and leader election.

- c. A cluster ID, generated during setup, is used to initialize and identify the cluster.

2. Cassandra:

- a. A distributed NoSQL database deployed as two nodes for high availability and fault tolerance.
- b. Data replication is configured via a SimpleStrategy with a replication factor of 2, ensuring data redundancy.

3. Flink:

- a. A stream-processing framework with distributed **JobManager** and **TaskManager** components deployed on both nodes.
- b. High availability is achieved using Zookeeper for coordination and state checkpoints stored in shared recovery directories.

4. Zookeeper:

- a. A distributed coordination service deployed as three nodes. Two Zookeeper nodes run on the second node for fault tolerance.
- b. It provides leader election and synchronization for Flink.

5. ksqlDB:

- a. A SQL-based stream-processing platform deployed in two instances .
- b. ksqlDB interacts directly with Kafka to perform real-time queries and transformations on data streams.
- c. Each instance uses a unique service ID for isolation and operates independently, enabling distributed SQL query execution.

This repository contains the full setup for the ECO-READY infrastructure, including configuration files, Docker Compose files, and automation scripts.

1. Docker Compose Configuration:

- a. A comprehensive docker-compose.yml file defines all services and their interconnections.
- b. Environment variables (.env) allow for flexible and customizable configurations, simplifying deployment across multiple nodes.



2. Custom Docker Images:

- a. A **custom Flink image** is built with Kafka connectors and Python support, enabling advanced stream processing capabilities.
- b. Consumer images (kafka-to-cassandra and flink-to-cassandra) facilitate the transfer and transformation of real-time data streams.

3. Automation Scripts:

- a. `generate-cluster-id.py`: Creates a unique Kafka cluster ID.
- b. `create_cassandra_tables.py`: Sets up Cassandra keyspaces and tables programmatically.

2.1.3. Replication and Fault Tolerance

The ECO-READY infrastructure is designed for high availability and resilience. Below is how each component contributes to fault tolerance:

1. Kafka:

- a. Topics are replicated across the two brokers, ensuring no data loss during broker failures.
- b. Consumer groups allow distributed processing and automatic failover if a consumer crashes.

2. Cassandra:

- a. Data replication across nodes ensures read/write availability even if one node fails.
- b. Failed nodes synchronize with the active cluster upon recovery, ensuring consistent data.

3. Flink:

- a. Flink jobs are stateful, with regular checkpoints stored in shared recovery directories. These checkpoints enable the system to resume processing from the last consistent state after a failure.
- b. JobManager redundancy ensures that if one JobManager fails, the other automatically takes over.

4. Zookeeper:

- a. Deployed as a three-node ensemble, Zookeeper tolerates the failure of one node while maintaining quorum for coordination and leader election.

5. ksqlDB:

- a. Two ksqlDB instances are deployed to ensure high availability. These instances can operate independently or distribute the workload, allowing for uninterrupted SQL-based stream processing.
- b. If one ksqlDB instance fails, the other can handle queries without disruption.

The ECO-READY infrastructure minimizes downtime and ensures data integrity during system or power failures:

1. Data Persistence:

- a. Kafka, Cassandra, and ksqlDB use persistent storage volumes, ensuring no data is lost during outages.
- b. Flink checkpoints stored in shared directories allow jobs to resume from their last consistent state.

2. Automatic Recovery:

- a. Kafka brokers re-synchronize with the cluster upon recovery, ensuring no messages are lost.
- b. Cassandra nodes replay missed writes and queries from the active nodes to maintain consistency.
- c. Flink restores jobs from checkpoints stored in the shared recovery directory.

3. High Availability:

- a. Distributing services across two nodes ensures no single point of failure.
- b. Zookeeper quorum and Kafka replication enable the system to function even if a single node or service is down.

2.2. Infrastructure Usage

The API serves as the core interface for interacting with the **ECO-READY** infrastructure, which is designed to provide a scalable, distributed, and fault-tolerant platform for managing organizations, projects, collections, and real-time data.

The API's GitHub repository can be found at:

<https://github.com/AuthEceSoftEng/ecoready-services>

Also, the detailed API documentation is available at:

<https://ecoready-services.issel.ee.auth.gr/docs>



2.2.1. Overview

The API facilitates seamless communication between users and the backend infrastructure, allowing them to:

1. Manage organizational and project structures.
2. Create, update, and delete collections tied to specific Kafka topics and Cassandra tables.
3. Enable real-time and batch data ingestion through Kafka.
4. Query and aggregate data stored in Cassandra.
5. Deploy and manage live data consumers and data pipelines using Flink.

This API abstracts the complexity of managing a distributed system, providing developers with a straightforward and intuitive way to interact with the infrastructure.

2.2.2. Key Functionality of the API

In this chapter, we will explore some key endpoints of the API, showcasing how they can be used to manage organizations, users, projects, and data effectively. These endpoints form the backbone of the system and allow for seamless interaction with its various components. Each endpoint is designed with specific functionality in mind, making it easier for users to organize, store, and retrieve data efficiently.

2.2.2.1. Organization Management

The API provides endpoints to create, update, delete, and retrieve information about organizations. These endpoints can be used only by the administrator of the infrastructure. An organization represents a group, such as a farming cooperative or an agricultural research institute, where projects and data will be managed. Each organization acts as a logical boundary for projects and collections. For example, to create an organization named “ecofarms” the admin will use the following endpoint.

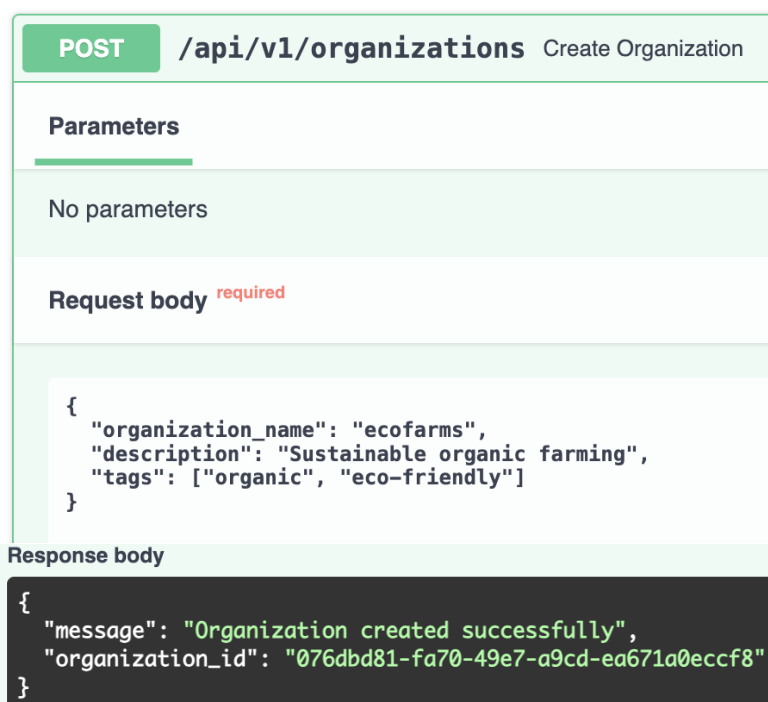


Figure 2 Screenshot of the API for Organization Management

2.2.2.2. User Management

The API provides endpoints that are responsible for the management of the users that belong to organizations. The admin is able to manage accounts for users who will manage and interact with the organization's projects. These could include farm managers, data analysts, or researchers. For example, to create a user for an organization will use the following endpoint.



Figure 3 Screenshot of the API for User Management

2.2.2.3. Project Management

The organization users can manage the various projects of the organization. Projects are created for distinct activities of the organization. For example, to create a project for the ecofarms organization the organization user can use the following endpoint.

Project Management

POST /api/v1/organizations/{organization_id}/projects Create Project	
Parameters	
Name	Description
organization_id * required string(\$uuid) (path)	076dbd81-fa70-49e7-a9cd-ea671a0eccf8
Request body required	
<pre>{ "project_name": "tomatoes", "description": "A greenhouse for organic tomatoes", "tags": ["greenhouse", "organic"] }</pre>	

Figure 4 Screenshot of the API for Project Management

2.2.2.4. Project Keys Management

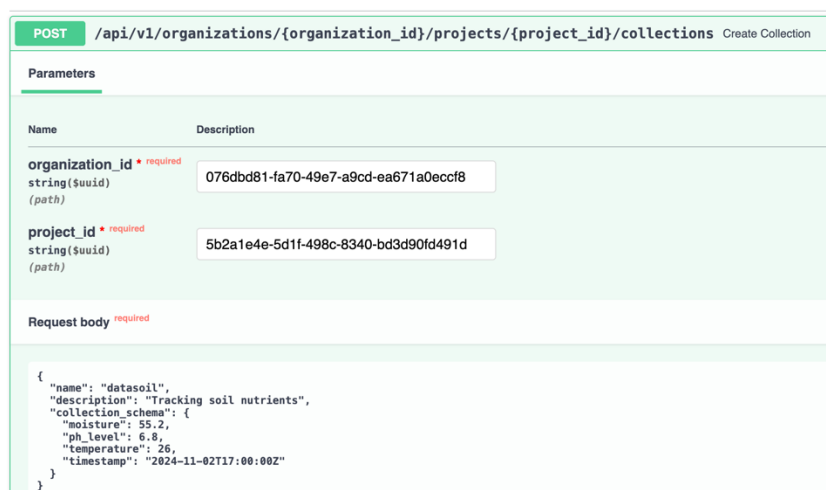
Organization users can generate keys to control access to their projects. Different key types (read, write, or master) provide varying levels of permissions to ensure secure and controlled data management.

POST /api/v1/api/organizations/{organization_id}/projects/{project_id}/keys Create Project Key	
Parameters	
Name	Description
organization_id * required string(\$uuid) (path)	076dbd81-fa70-49e7-a9cd-ea671a0eccf8
project_id * required string(\$uuid) (path)	5b2a1e4e-5d1f-498c-8340-bd3d90fd491d
Request body required	
<pre>{ "key_type": "master" }</pre>	

Figure 5 Screenshot of the API for Project Keys Management

2.2.2.5. Collection Management

Collections are created within projects to organize and store specific types of data, such as soil samples, crop statistics, or weather conditions. Each collection has a schema defining the structure of the data. The schema is auto generated by providing an instance of the data of the collection. The API supports creating, updating, deleting, and querying collections. To create a collection the following endpoint can be used.



POST /api/v1/organizations/{organization_id}/projects/{project_id}/collections Create Collection

Parameters

Name	Description
organization_id * required string(\$uuid) (path)	076dbd81-fa70-49e7-a9cd-ea671a0eccf8
project_id * required string(\$uuid) (path)	5b2a1e4e-5d1f-498c-8340-bd3d90fd491d

Request body * required

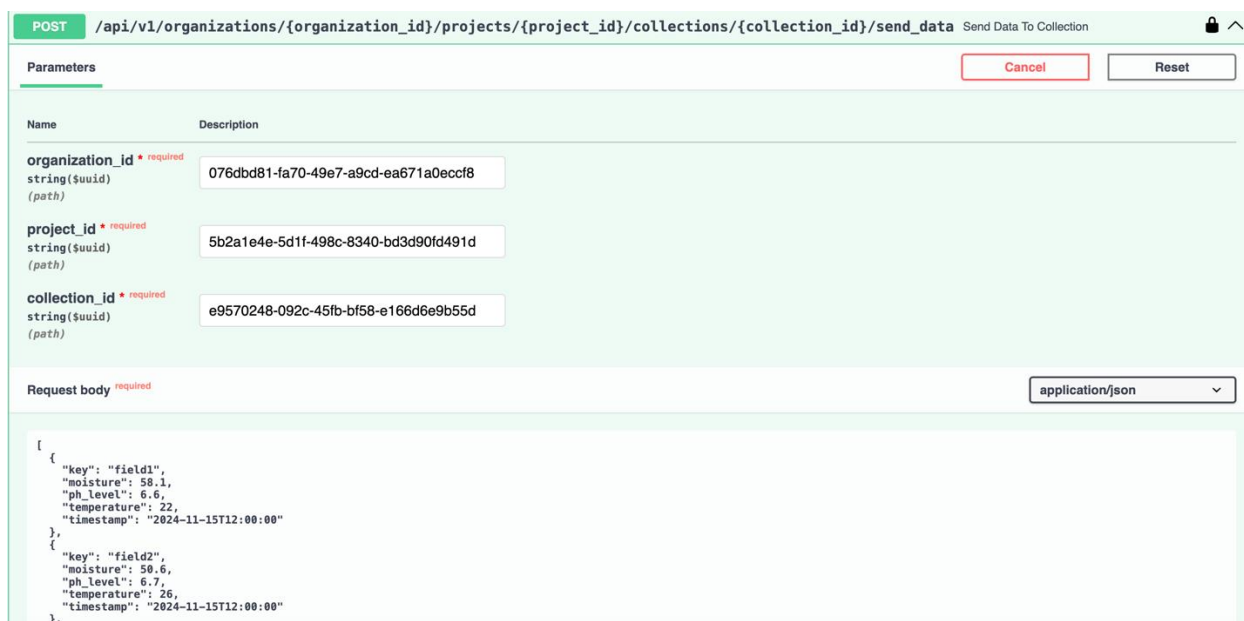
```
{
  "name": "datasoil",
  "description": "Tracking soil nutrients",
  "collection_schema": {
    "moisture": 55.2,
    "ph_level": 6.8,
    "temperature": 26,
    "timestamp": "2024-11-02T17:00:00Z"
  }
}
```

Figure 6 Screenshot of the API for Collection Management

This endpoint creates the corresponding Kafka Topic and Cassandra Table for the collection.

2.2.2.6. Data Ingestion

Through the appropriate endpoint the ones who have access through the suitable key can send data to the corresponding collection. The data pass through the Apache Kafka and are stored to Apache Cassandra.



POST /api/v1/organizations/{organization_id}/projects/{project_id}/collections/{collection_id}/send_data Send Data To Collection

Parameters

Name	Description
organization_id * required string(\$uuid) (path)	076dbd81-fa70-49e7-a9cd-ea671a0eccf8
project_id * required string(\$uuid) (path)	5b2a1e4e-5d1f-498c-8340-bd3d90fd491d
collection_id * required string(\$uuid) (path)	e9570248-092c-45fb-bf58-e166d6e9b55d

Request body required

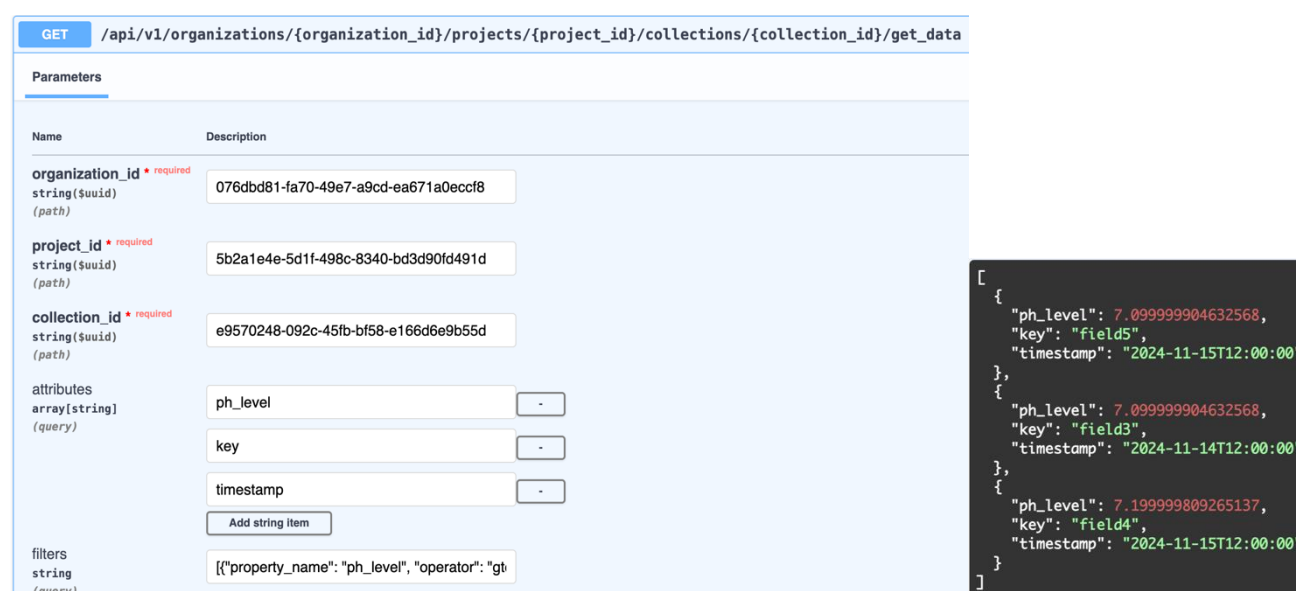
application/json

```
[
  {
    "key": "field1",
    "moisture": 58.1,
    "ph_level": 6.6,
    "temperature": 22,
    "timestamp": "2024-11-15T12:00:00"
  },
  {
    "key": "field2",
    "moisture": 50.6,
    "ph_level": 6.7,
    "temperature": 26,
    "timestamp": "2024-11-15T12:00:00"
  }
]
```

Figure 7 Screenshot of the API for Data Ingestion

2.2.2.7. Data Retrieval

The API allows querying data stored in the infrastructure. Users fetch stored data for analysis, visualization, or reporting. Filtering, selecting attributes, and ordering the data are supported for efficient data retrieval.



GET /api/v1/organizations/{organization_id}/projects/{project_id}/collections/{collection_id}/get_data

Parameters

Name	Description
organization_id * required string(\$uuid) (path)	076dbd81-fa70-49e7-a9cd-ea671a0eccf8
project_id * required string(\$uuid) (path)	5b2a1e4e-5d1f-498c-8340-bd3d90fd491d
collection_id * required string(\$uuid) (path)	e9570248-092c-45fb-bf58-e166d6e9b55d
attributes array(string) (query)	ph_level - key - timestamp - Add string Item
filters string (query)	[{"property_name": "ph_level", "operator": "gt"}]

```
[
  {
    "ph_level": 7.099999904632568,
    "key": "field5",
    "timestamp": "2024-11-15T12:00:00"
  },
  {
    "ph_level": 7.099999904632568,
    "key": "field3",
    "timestamp": "2024-11-14T12:00:00"
  },
  {
    "ph_level": 7.199999809265137,
    "key": "field4",
    "timestamp": "2024-11-15T12:00:00"
  }
]
```

Figure 8 Screenshot of the API for Data Retrieval

In this case the user asked for the ph_level, timestamp and key attributes only for the records that have ph_level over 7.

2.2.2.8. Statistics

Users are able perform statistical operations (e.g., averages, maximum values) on the stored data to derive insights using the following endpoint.

Get Data Statistics

GET /api/v1/organizations/{organization_id}/projects/{project_id}/collections/{collection_id}/statistics

Parameters

Name	Description
organization_id * required string(\$uuid) (path)	076dbd81-fa70-49e7-a9cd-ea671a0eccf8
project_id * required string(\$uuid) (path)	5b2a1e4e-5d1f-498c-8340-bd3d90fd491d
collection_id * required string(\$uuid) (path)	e9570248-092c-45fb-bf58-e166d6e9b55d
attribute * required string (query)	temperature
stat string (query)	Statistical operation to perform avg
interval string (query)	every_2_days

Figure 9 Screenshot of the API for Statistics

In this case the user requested the average temperature on a 2-day basis for each region.

2.2.3. API Integration with Infrastructure

1. Kafka Integration:

- Kafka topics are created for each collection.
- Producers ingest data into these topics, and consumers process and persist data into Cassandra.

2. Cassandra Integration:

- Metadata tables (e.g., organization, project, collection) are stored in a centralized keyspace (metadata).
- Each organization's data is isolated within its own keyspace, ensuring logical separation and scalability.

3. Flink Integration:

- a. Flink jobs are deployed to process and aggregate Kafka streams in real-time, with results written to new Kafka topics.

4. Docker Integration:

- a. The API uses Docker to manage consumer containers dynamically.
- b. Each collection has a corresponding Kafka-to-Cassandra consumer deployed in a container.

2.2.4. Fault Tolerance and Scalability

1. Replication:

- a. Kafka and Cassandra are configured with replication to ensure data durability and fault tolerance.
- b. Cassandra tables use a SimpleStrategy with a replication factor of 2, enabling resilience against node failures.

2. Dynamic Deployment:

- a. The API dynamically deploys and manages Docker containers for consumers, allowing for easy scaling and recovery.

3. Resilient Flink Pipelines:

- a. Flink jobs are designed with high availability, leveraging Zookeeper for coordination.

4. Consistency:

- a. The API enforces strict schema validation and consistency across components, ensuring data integrity.

2.2.5. API Security

The ECO-READY API enhances security by utilizing a **multi-layered authentication approach**, incorporating **OAuth2, JWT tokens, and API keys** to control access. OAuth2 and JWT tokens handle **user authentication and authorization**, ensuring that only authenticated users can access resources. JWT tokens provide **stateless session management**, allowing secure API interactions without continuously querying a database for session validation.

For **collection-level access**, the system employs **API tokens** to regulate read and write permissions. These API keys are generated per project and can be assigned different permission levels, such as **read, write, or master access**, ensuring that only authorized external users can interact with the data. Moreover, instead of relying on collection or project names in API calls, **UUIDs (unique identifiers)** are used, making endpoints more secure by preventing predictable access patterns. This **combination of factors** ensures a **secure access control mechanism** for managing system's data efficiently.

2.2.6. Integration Data Sources and Living Labs

To develop and enrich the ECO-READY platform with data aligned to the project's objectives, we integrated publicly available datasets alongside real-time data streams from Living Labs. The European Commission's **Agri-Food Markets Database** and the **FAOSTAT Food and Agriculture Data** were vital contributors, offering key market information across various agricultural products. By utilizing the API, we systematically incorporated this data into our infrastructure, ensuring it remained accessible and analyzable within the ECO-READY platform.

Additionally, the Living Labs provided localized observational data from environmental and agricultural experiments. These data inputs, ingested through the API, bridged the gap between macro-level datasets and granular, community-driven insights. This dual integration ensures the platform remains adaptable, responsive to real-world conditions, and capable of addressing both large-scale and local-level challenges effectively.



3. ECO-READY Dashboard and Map

3.1. Architecture and Implementation

The ECO-READY Dashboard consists of several interconnected components that work in synergy to retrieve, process and display desired data.

The user can interact with the dashboard through the frontend which is based on the React framework and styled with the use of Materia-UI library. Subsequently, it communicates with the backend via HTTP requests to a RESTful API exposed by the Node.js server, which essentially serves as a bridge between the frontend and the databases. It works by processing incoming requests from the frontend, and fetching data from external data systems, through their respective services. The ECO-READY infrastructure, is responsible for all calls regarding the project's data, while the MongoDB database handles user data.

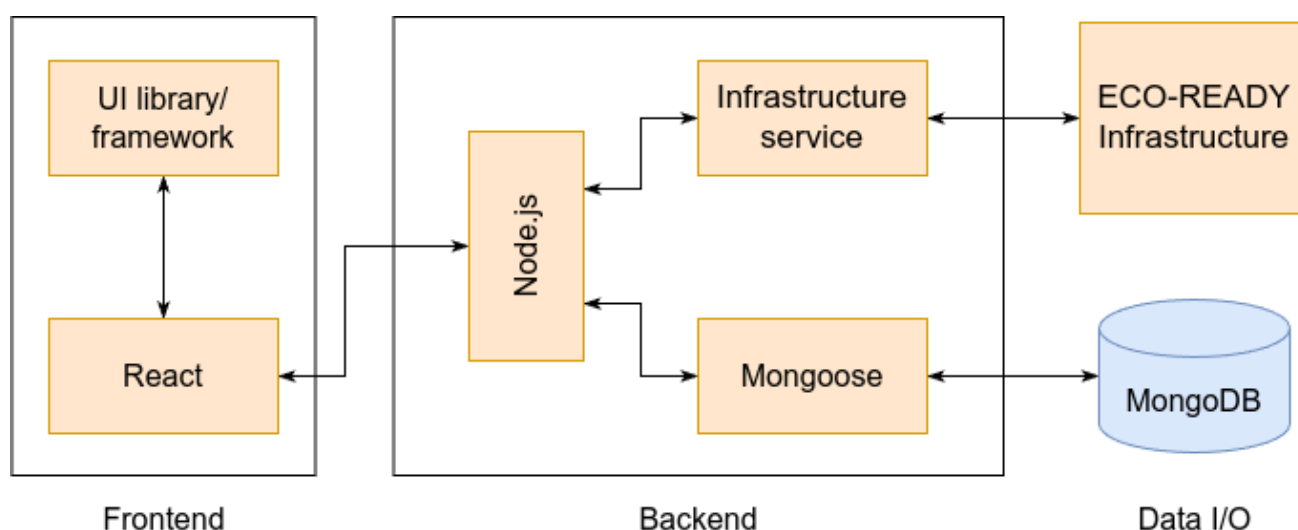


Figure 10 Dashboard Technical Architecture

All code regarding frontend and backend development can be found in the following GitHub repositories.

Frontend: <https://github.com/AuthEceSoftEng/ecoready-dashboard-frontend>

The frontend contains all the files needed to create the UI and UX of the dashboard. These include the images and icons that appear on the screens, as well as the styling of each screen. Furthermore, we have also built a set of reusable React components, so that we can easily and effectively display desired information. The components range from graphs and plots, to buttons, date pickers, sliders and forms, among others, that all contribute

towards a pleasant user experience. Of course, the repo also contains all the screens that one can see on the Dashboard (further explained below) and finally, calls and configurations for effective communication and data retrieval from the backend.

Add here the links to the GitHub repositories and provide instructions (~1 paragraph per repo) explaining how they are implemented.

Backend: <https://github.com/AuthEceSoftEng/ecoready-dashboard-backend>

The backend contains the API endpoints and services that are needed for interaction with the frontend, as well as with the ECO-READY infrastructure. These endpoints handle requests for file uploads, data retrieval, and other functionalities required by the frontend. It also contains application configurations for PM2 (Process Manager 2), such as environment variables and port for production.

To ensure a seamless and reliable deployment of the Node.js application, we utilize NPM (Node Package Manager) for managing dependencies and PM2 for process management. NPM is used to initialize the project and install all necessary packages, ensuring the application is fully equipped with required modules, while PM2 is employed to handle the runtime of the application, providing features such as automated restarts in case of crashes and robust process monitoring. To enhance reliability, we also include a startup script is configured using PM2's built-in startup command, that enables the application to restart automatically after a system reboot or power failure. Finally, the configuration includes *pm2 save* to preserve the process list for future restoration. These practices ensure that the application remains operational with minimal manual intervention, offering an efficient and resilient production environment.

3.2. Dashboard Usage

3.2.1. Dashboard API

A prototype API can be found at the following address:

<https://ecoready-dashboard.issel.ee.auth.gr/backend/api-docs/>

The API calls, that the dashboard supports as of now, are mainly focused on data and file management, along with account creation and authentication methods. The calls can also be seen below, categorized according to their functionality:



ECO-READY Dashboard API 1.0.0 OAS 3.0

This is the ECO-READY dashboard API. It is used to host the actions of the backend of the ECO-READY dashboard, and communicates with the different components (mongo users DB, infrastructure, integrations).

Servers

<http://localhost:4000/api/v1> - Local server

Collections

GET </eco-ready-services/getcollections> Retrieve a list of collections

Data

GET </eco-ready-services/getdata> Retrieve data from a collection

GET </eco-ready-services/getdatastatistics> Retrieve data statistics from a collection

Files

POST </file/delete> Delete a file

POST </file/> Upload a file

PUT </file/> Re-upload a file

Public

GET </> Public route to verify server is working

Users

POST </createUser> Create a new user

POST </createUserInvited> Create a new user with an invitation token

GET </user/> Retrieve a list of all users

POST </user/delete> Delete a user

Authentication

POST </authenticate> Authenticate a user

POST </authenticateGoogle> Authenticate a user using Google OAuth

GET </user/decode> Decode the authenticated user's details

GET </user/attempt-auth> Attempt authentication

Password

POST </forgotpassword> Request password reset

POST </resetpassword> Reset a user's password

Invitations

POST </user/> Send an invitation to a user

Figure 11 Screenshot of the Dashboard Backend API

3.2.2. Dashboard Screens

The prototype is available at the address:

<https://ecoready-dashboard.issel.ee.auth.gr/>

The landing page of the dashboard is shown in Figure 12. New users can sign up, while the dashboard also offers functionality for forgotten passwords (in this case a link is sent to the email of the user). This is achieved using SendGrid.

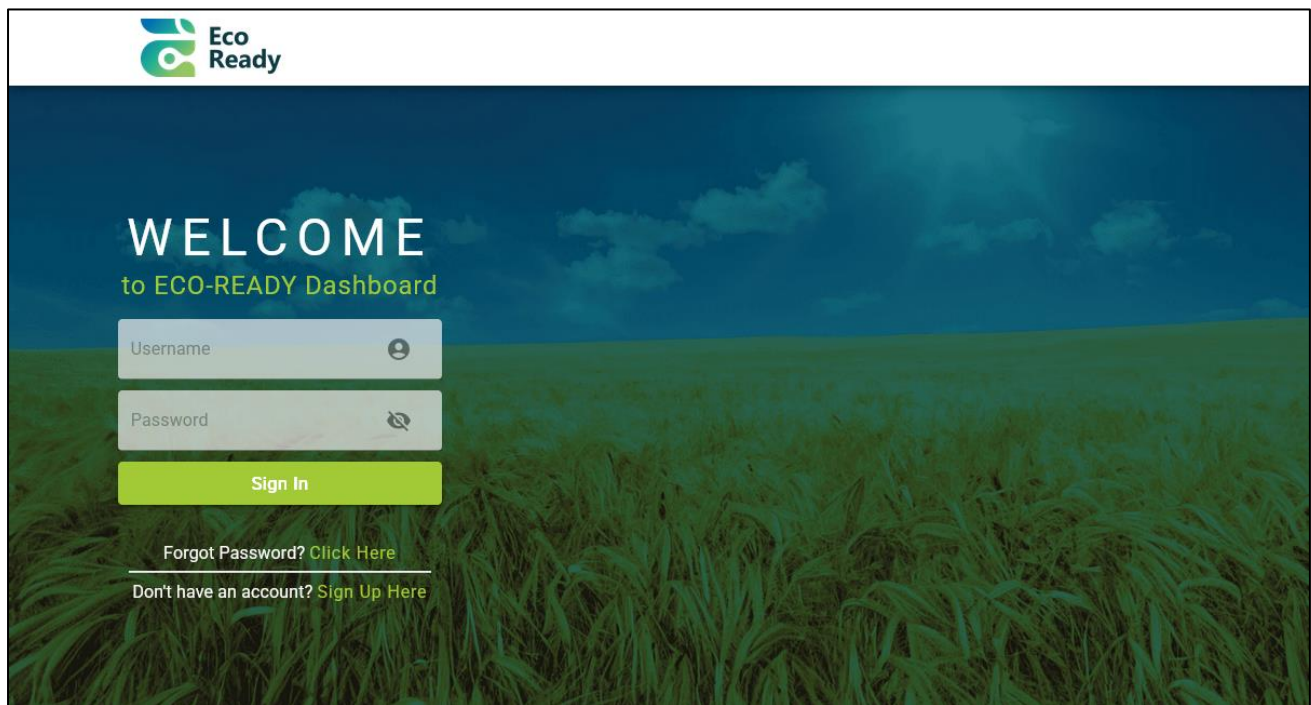


Figure 12 Screenshot of the ECO-READY Dashboard Landing page

Upon signing up to the dashboard, users can log in using their credentials. The first screen they see when entering the dashboard is the overview page (Figure 13). In this page, one can select to view the data and details of a specific Living Lab or to view the details of a product, either on a product page or on the map.

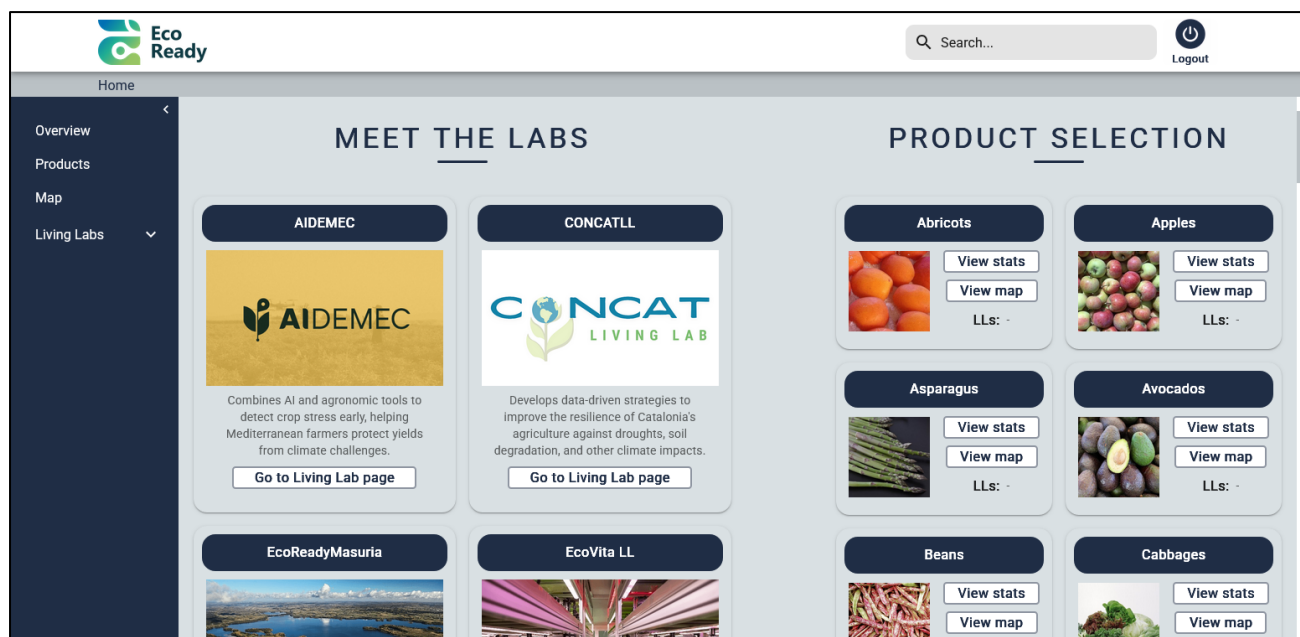


Figure 13 Screenshot of the ECO-READY Dashboard Overview page

For the product page, different metrics are defined for each product, based on its production statistics and its prices (shown in Figure 14 and Figure 15, respectively, which are in the same page - scrolled up and down).

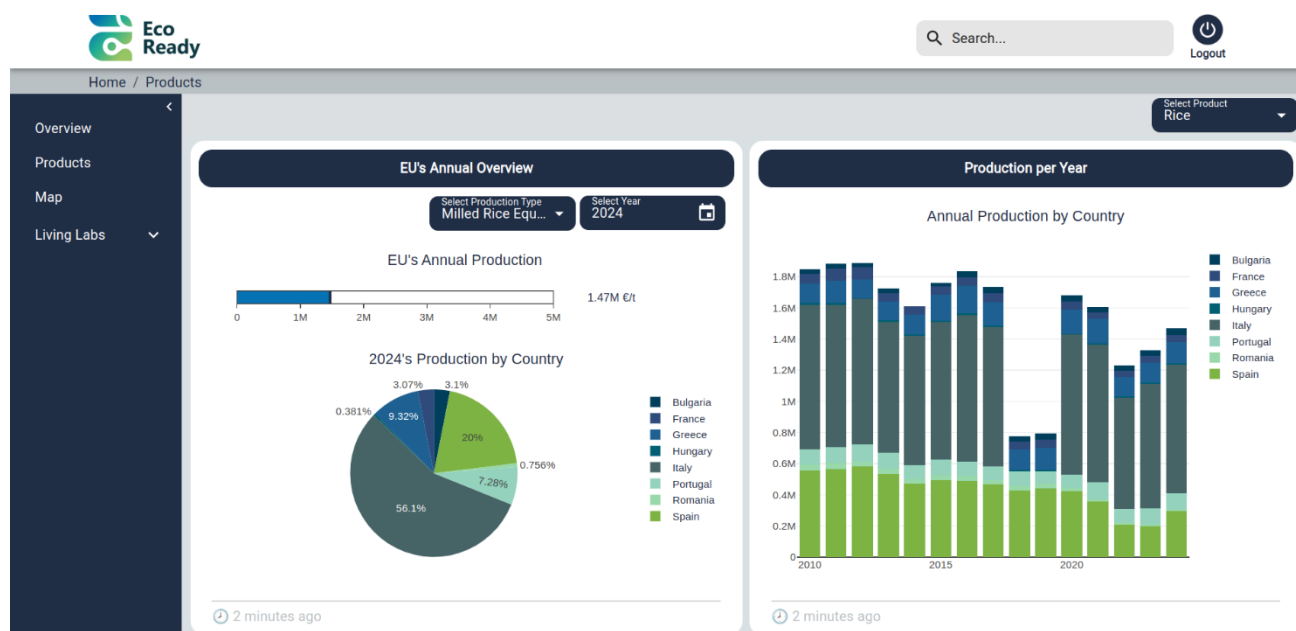


Figure 14 Screenshot of the ECO-READY Product page (scrolled up, production view)

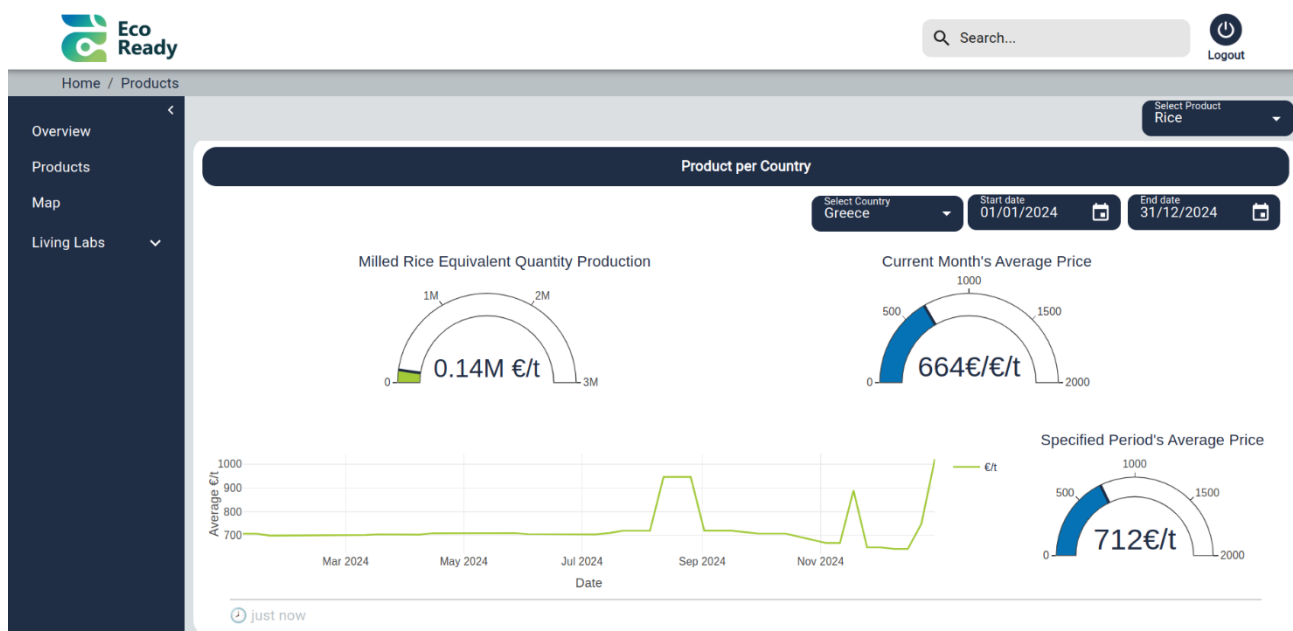


Figure 15 Screenshot of the ECO-READY Product page (scrolled down, pricing view)

Also, the user is given the option to choose the product, as well as the time period of the displayed information (shown in Figure 14figure 7 and Figure 15gure 8, respectively, in similar format).



Figure 16 Screenshot of the ECO-READY Product page (scrolled up, production view)

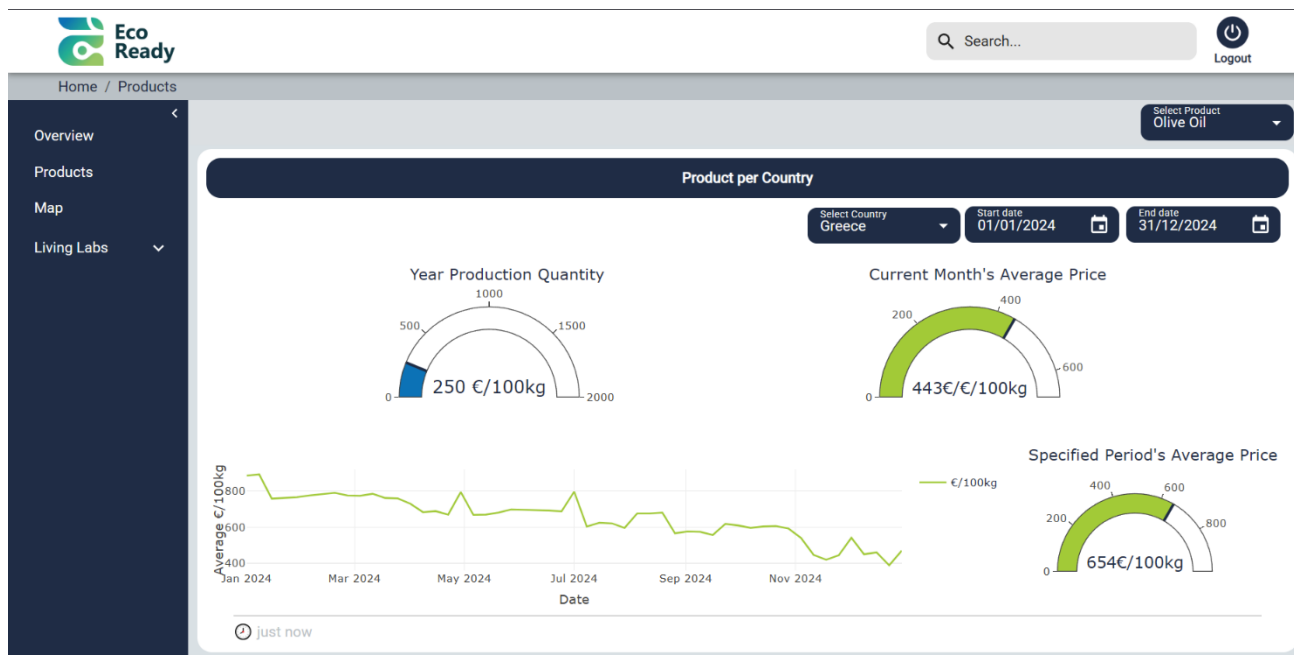


Figure 17 Screenshot of the ECO-READY Product page (scrolled up, production view)

To further examine the details of a product throughout Europe, one can use the interactive map (Figure 9). The map allows filtering based on products on a yearly basis. The map then depicts the average price of the product for each country as a layer, and/or (for the products that include the relevant variables) production stats (e.g. total yield per country).

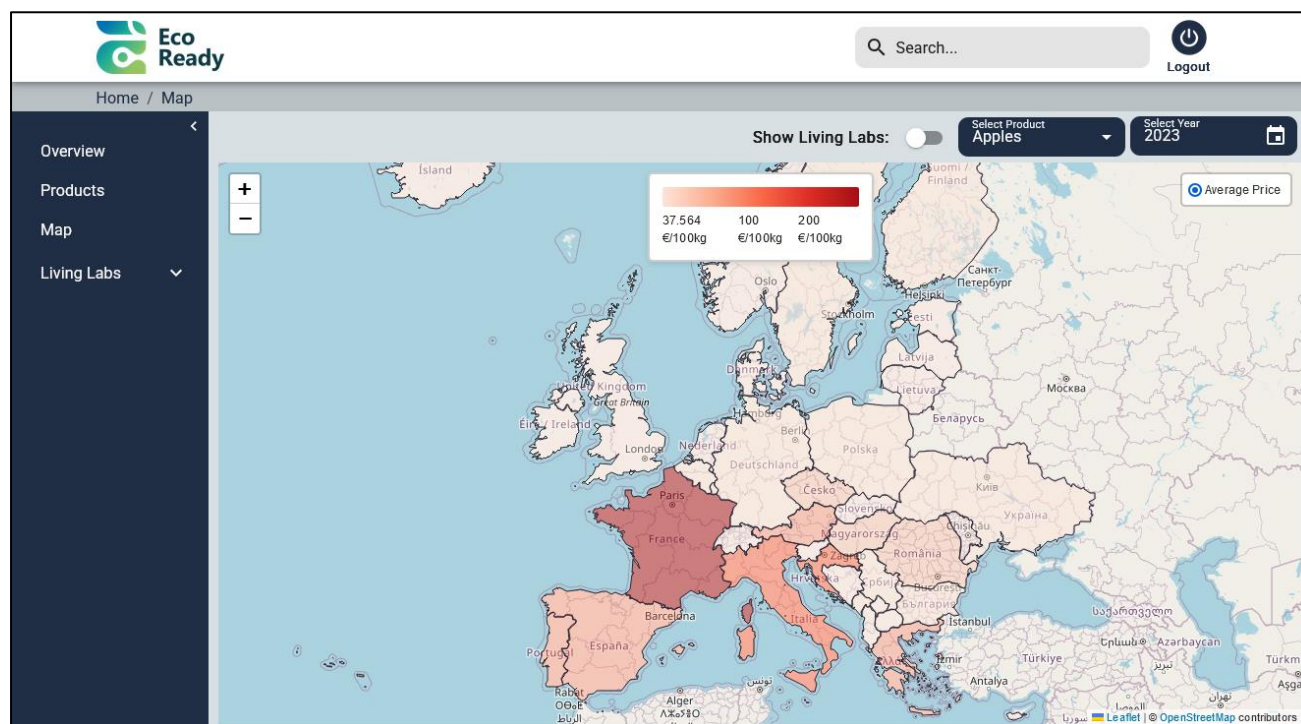


Figure 18 Screenshot of the ECO-READY Map page depicting a product pricing layer

It is also possible to depict the Living Labs in the same map, such as in Figure 19.

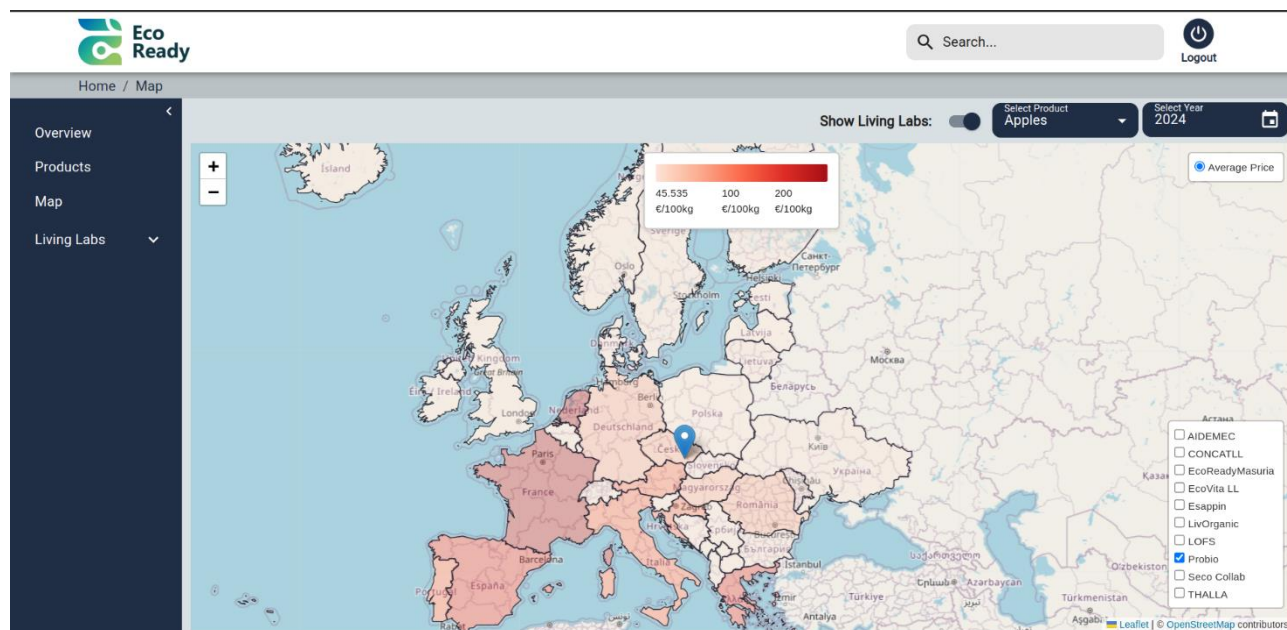


Figure 19 Screenshot of the ECO-READY Map page depicting a Living Lab

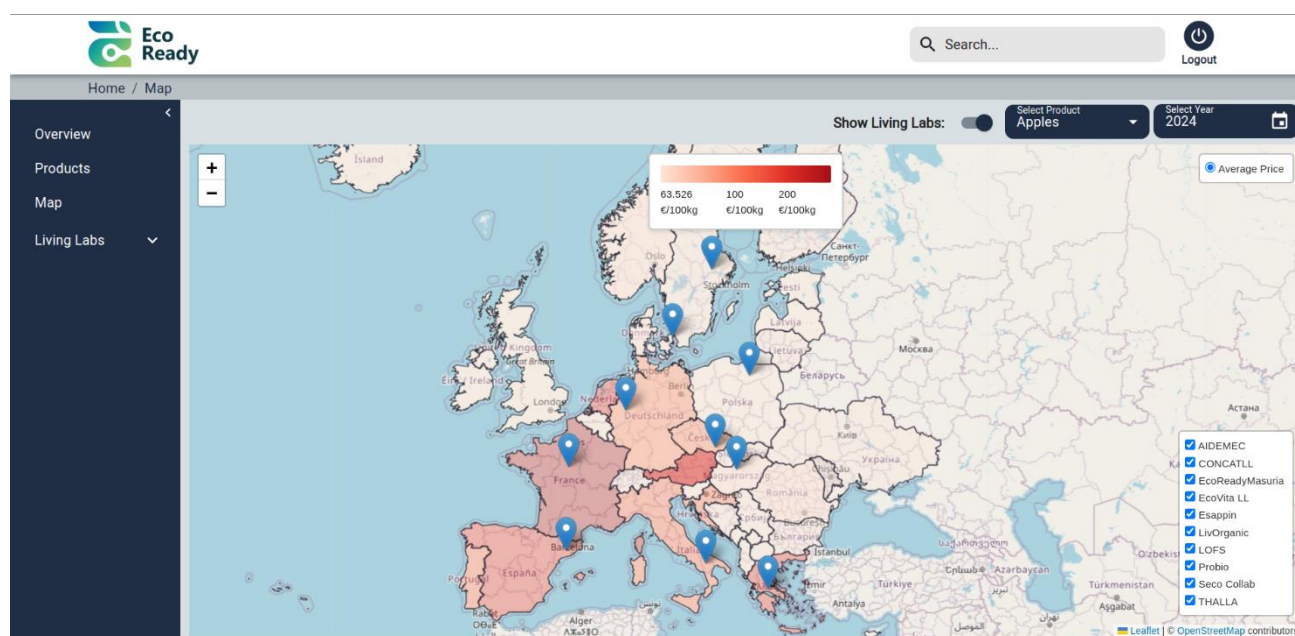


Figure 20 Screenshot of the ECO-READY Map page depicting multiple Living Labs

Also, each Living Lab has each own dedicated page, where information of interest is presented. The information can vary from meteorological data, to production techniques

and metrics. Options regarding the products, farming cites, and time periods are present here as well, depending on the Living Lab. Example screens of two LLs are presented below.

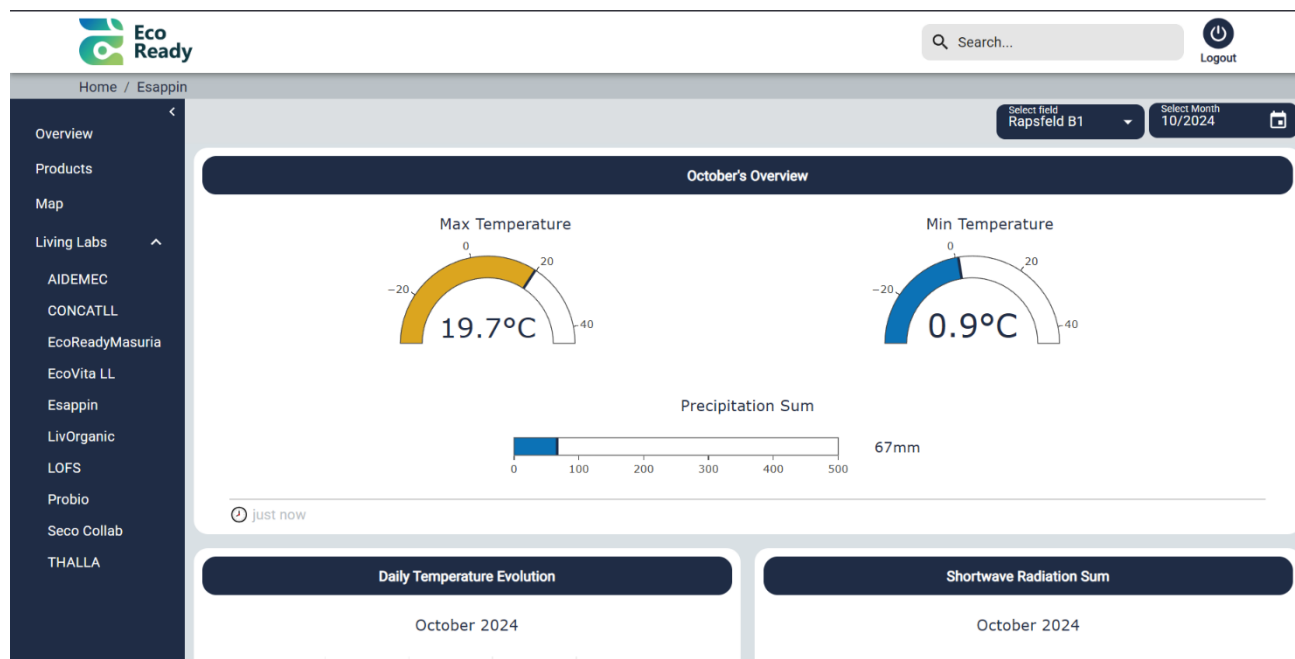


Figure 21 Screenshot of the ESAPPIN LL screen (1/3)

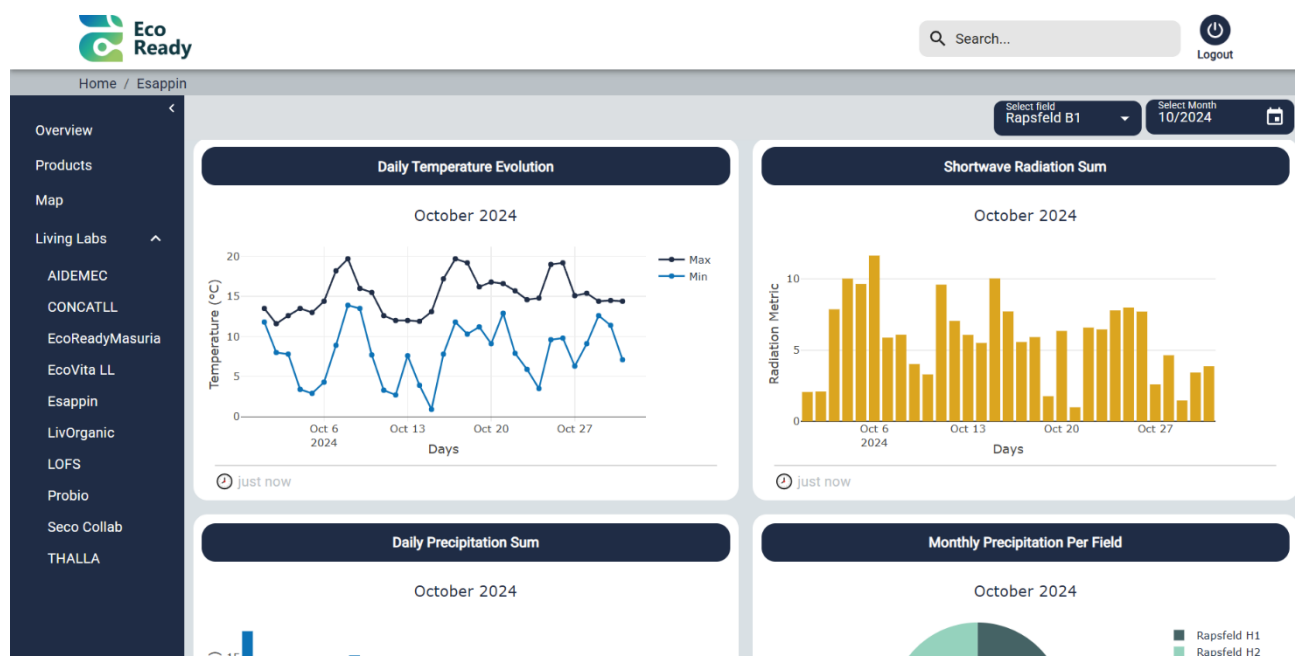


Figure 22 Screenshot of the ESAPPIN LL screen (2/3)

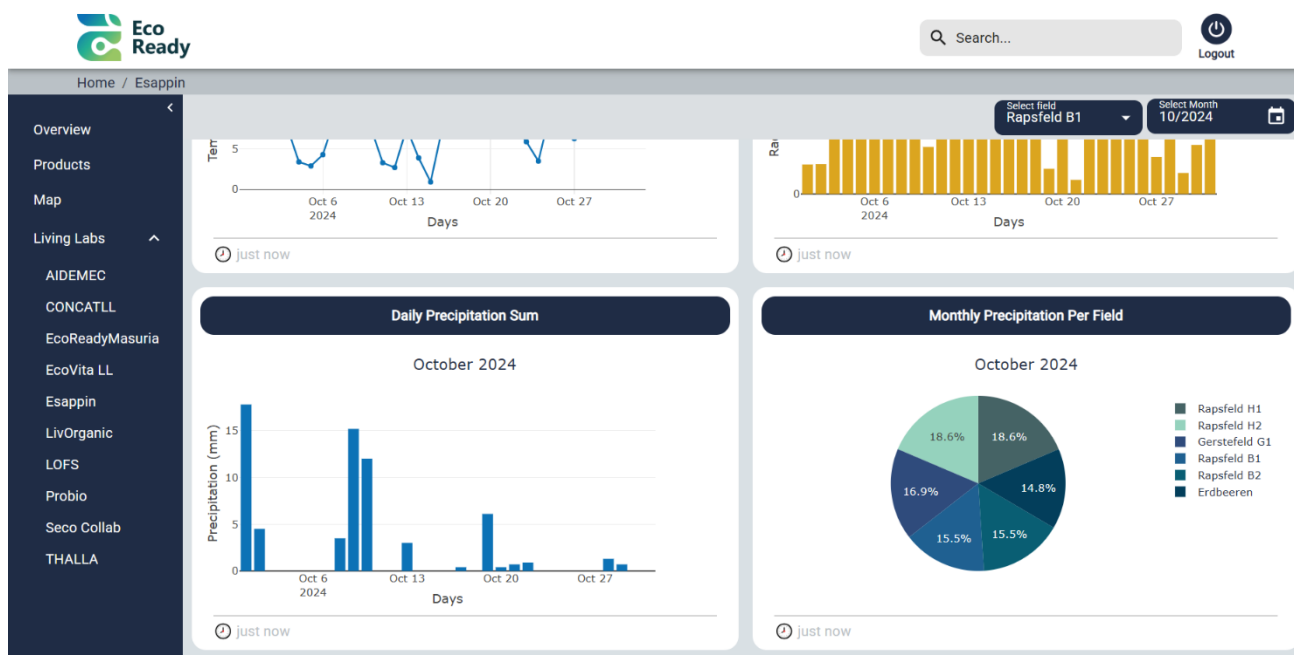


Figure 23 Screenshot of the ESAPPIN LL screen (3/3)

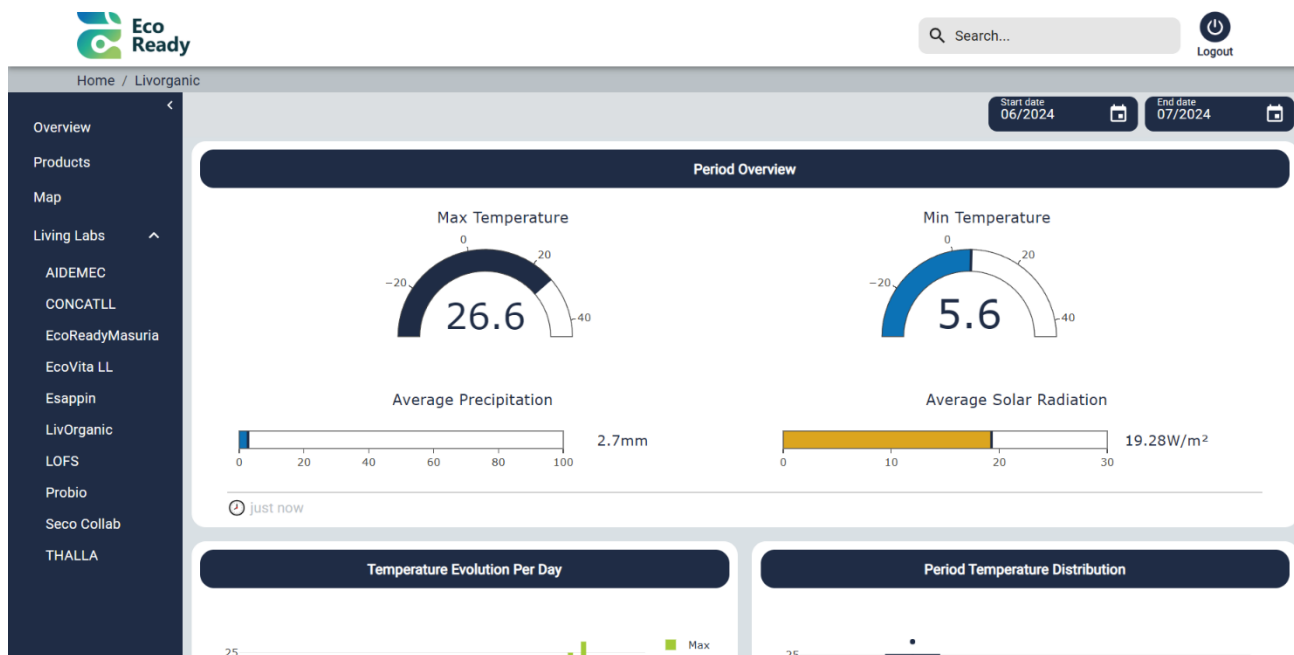


Figure 24 Screenshot of the LivOrganic LL screen (1/3)

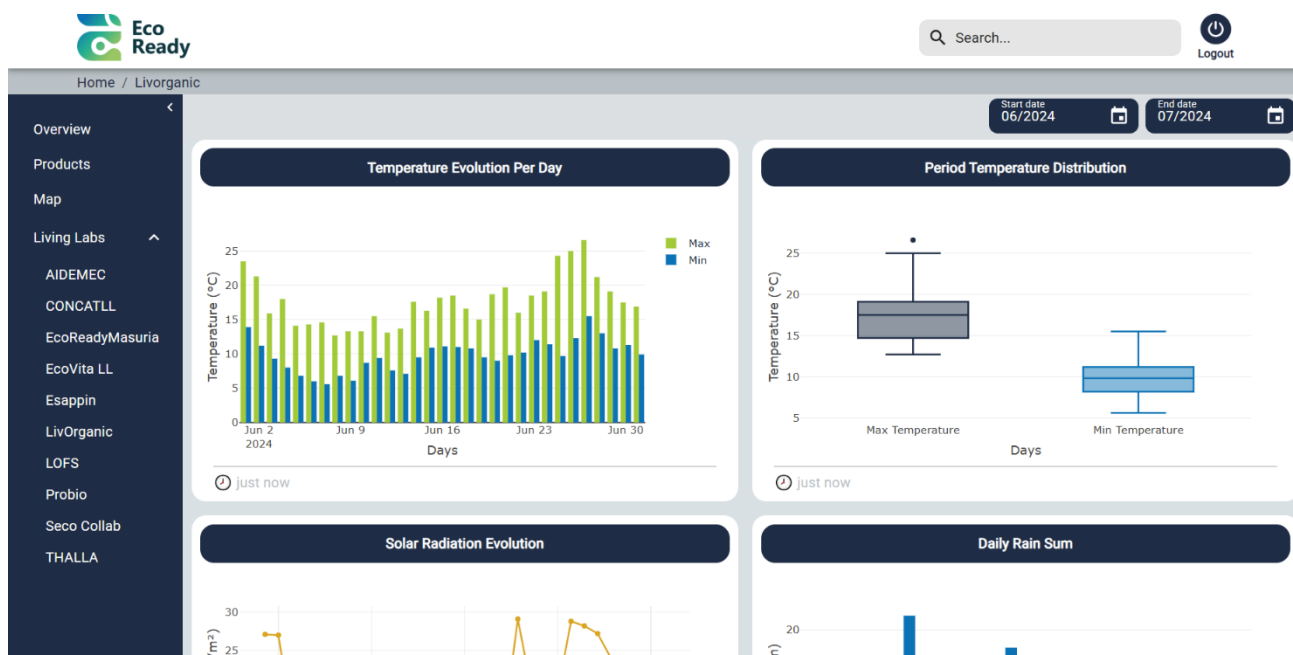


Figure 25 Screenshot of the LivOrganic LL screen (2/3)

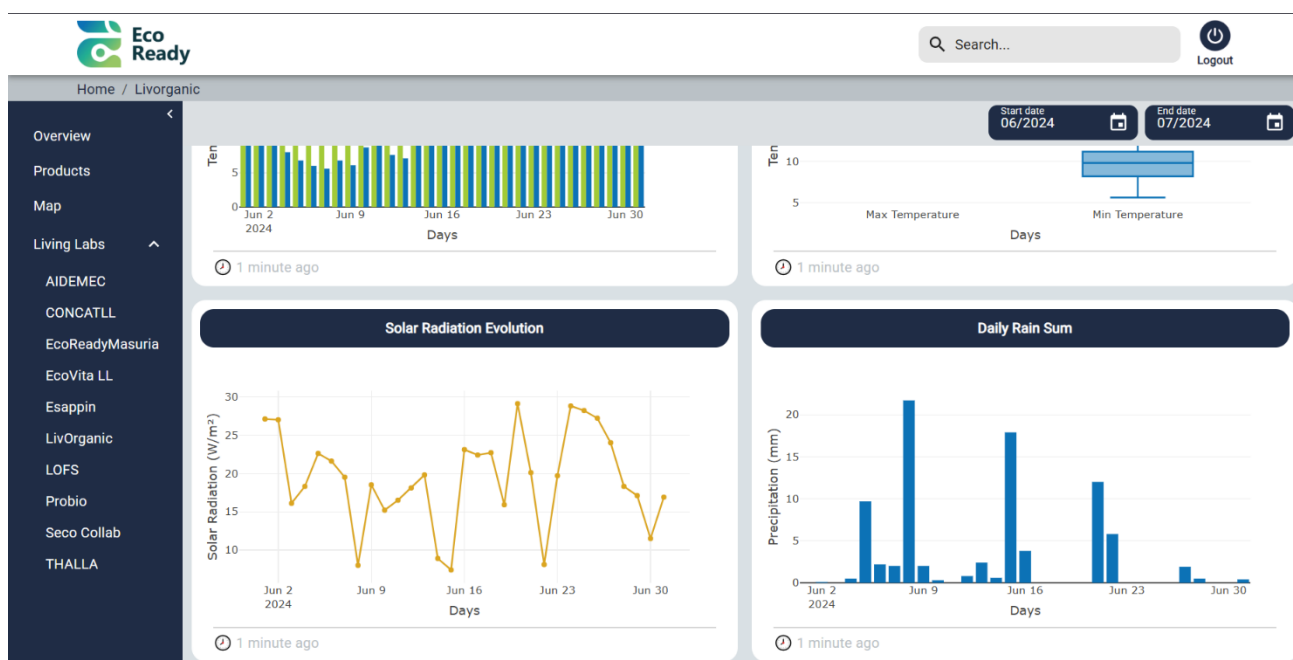


Figure 26 Screenshot of the LivOrganic LL screen (3/3)

4. ECO-READY Mobile App

4.1. Architecture and Implementation

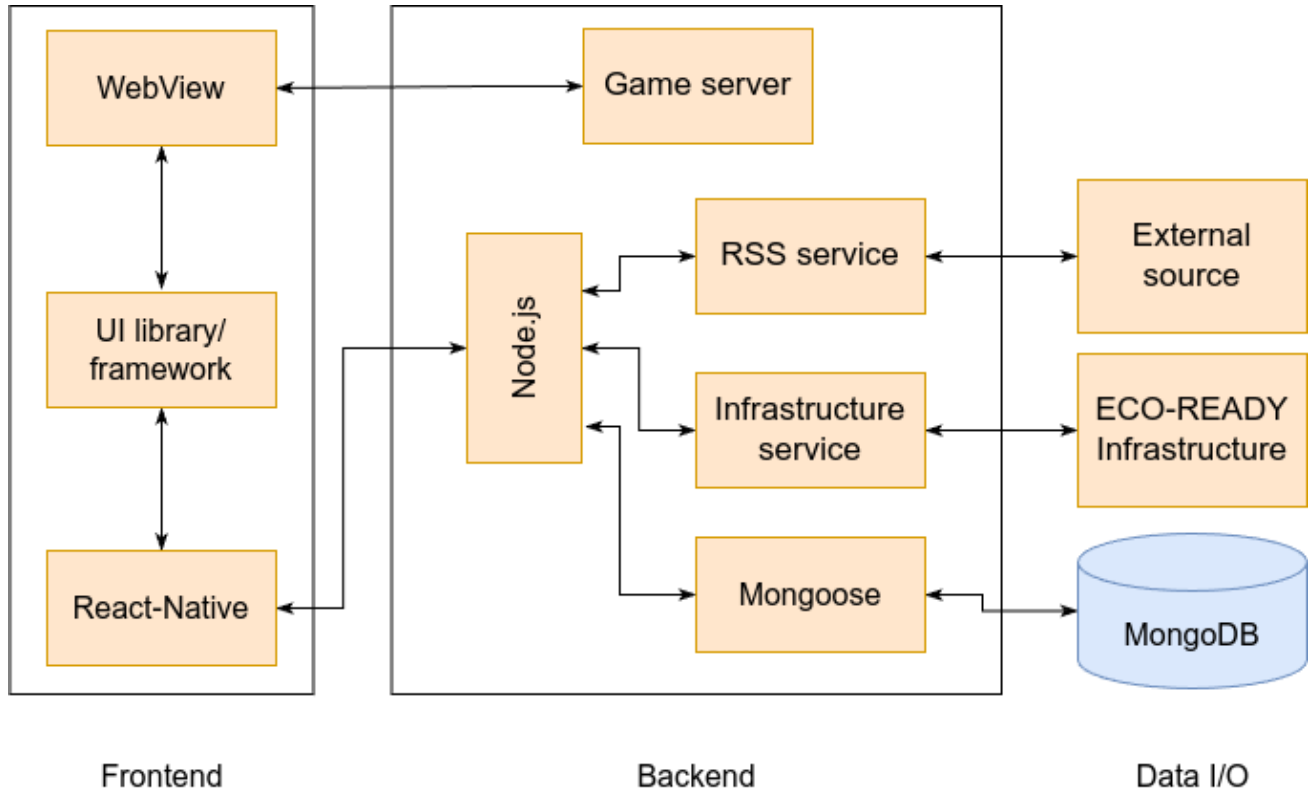


Figure 27 Mobile Application Technical Architecture

ECO-READY mobile application is a comprehensive tool designed to make sustainable living accessible, actionable, and engaging for everyone. By combining education, practical tools, and interactive content, the app empowers users to understand their environmental impact and take meaningful steps toward a greener future. Whether it's tracking your carbon footprint, staying informed with the latest eco-news, or playing games that teach sustainability strategies, ECO-Ready app brings tools you need into one intuitive platform.

The GitHub repository for the mobile app is available at:

<https://github.com/AuthEceSoftEng/ecoready-mobile-app>

The technical architecture of the mobile app is divided into three main components:

1. **Frontend:** Built with **React-Native** for cross-platform compatibility (iOS and Android). The UI is rendered using a combination of a **UI library/framework** and **WebView**.

2. **Backend:** Developed with **Node.js** to handle API requests, manage data, and facilitate communication with various services.
3. **Data I/O:** Powered by **MongoDB**, with the ORM layer handled by **Mongoose**. External services, such as RSS feeds and game APIs, are integrated into the ecosystem.

Each module is designed to work independently, making it easy to add new features such as games, calculators, and RSS feeds without affecting the overall system.

- **React-Native (Frontend):** React-Native provides the main entry point for the app, enabling a modular approach to UI design. Screens and features are organized into components, which can be reused or replaced easily. The WebView is used to display dynamic content such as HTML-based calculators or external resources directly within the app.
- **Node.js (Backend):** The backend, built with Node.js, acts as a mediator between the app's frontend and external services. It is structured with independent microservices for game servers, RSS services, and infrastructure management. This modularity ensures scalability and maintainability.
- **Mongoose and MongoDB (Data I/O):** MongoDB serves as the primary database for storing user data, configurations, and content. Mongoose provides a structured schema and query mechanism for easy data manipulation.
- **External Services:** Integration with external RSS feeds and game servers is handled through dedicated APIs, ensuring each service operates independently. The ECO-READY infrastructure further allows seamless communication with new services.

4.2. Mobile App Usage

The APK of the application is available at the following address:

<https://github.com/AuthEceSoftEng/ecoready-mobile-app/releases/tag/latest>

The mobile app is designed to inspire eco-conscious living through its user-friendly interface and comprehensive tools. By offering actionable insights and engaging features, the app empowers users to reduce their environmental footprint and adopt sustainable practices.

The app greets users with a clean, welcoming dashboard that introduces its core offerings,



all of which are easy to navigate via an intuitive menu bar. The key functionality include access to a news feed, environmental calculators, eco-friendly tips and educational games, all aimed at promoting sustainability in an engaging and practical way.

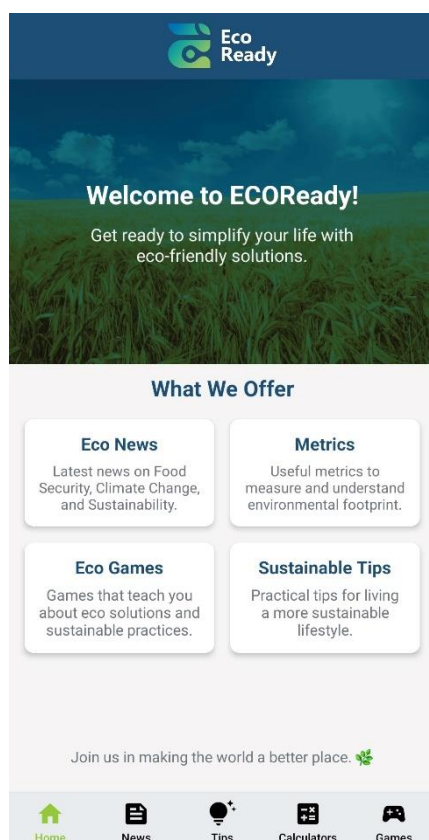


Figure 28 Mobile Application Overview Screen

Key Functionality

Eco News

ECO-Ready keeps you informed with a dedicated news section, featuring the latest updates on global environmental topics like biodiversity, climate change, and food security. Curated from reputable sources, these articles help users stay up-to-date with current issues and solutions. By staying informed, users gain the knowledge needed to make environmentally conscious decisions in their daily lives.

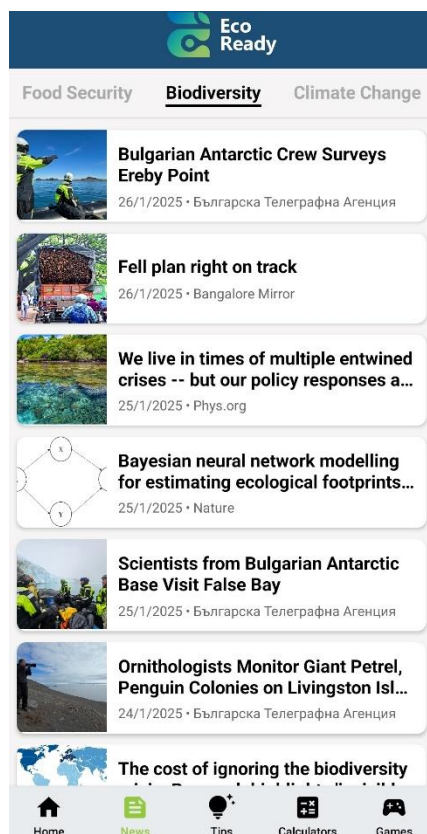


Figure 29 Mobile Application Eco News Screen

Eco Tips

The app provides a wide variety of practical, eco-friendly advice to help users adopt sustainable habits. These tips are organized into categories for easy navigation, including:

- Water Conservation
- Energy Efficiency
- Climate Change
- Biodiversity

Each tip is designed to be actionable, offering clear ways to reduce waste, save energy, and protect the environment in simple, everyday activities.

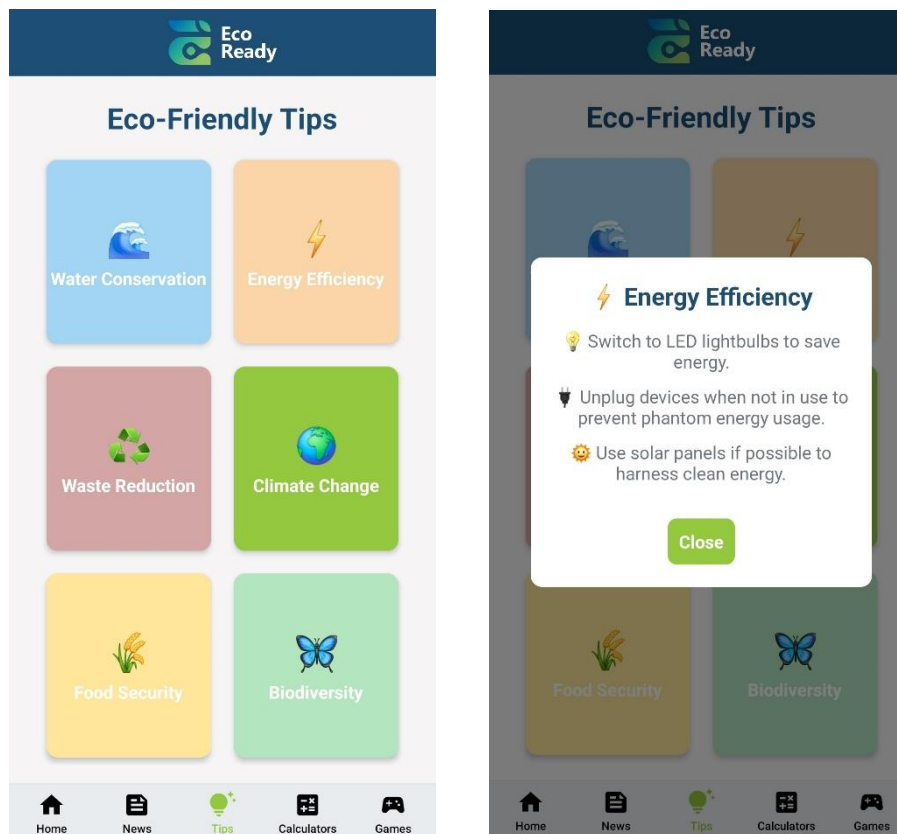


Figure 30 Mobile Application Eco Tips Screenshots

Environmental Calculators

ECO-Ready offers a comprehensive suite of calculators to help users assess their environmental impact and find ways to improve. These include:

- **Transport Impact Calculator:** Enter your travel distance and vehicle type to instantly see your CO₂ emissions.
- **Water Usage:** Understand how much water you consume and identify areas to save.
- **Renewable Energy Savings:** Analyze energy consumption and possible economic benefits of solar panels.
- **Recycling Impact:** Evaluate the environmental impact of how much you recycle.

These tools provide more than just measurements—they offer actionable insights, empowering users to take meaningful steps toward sustainability.

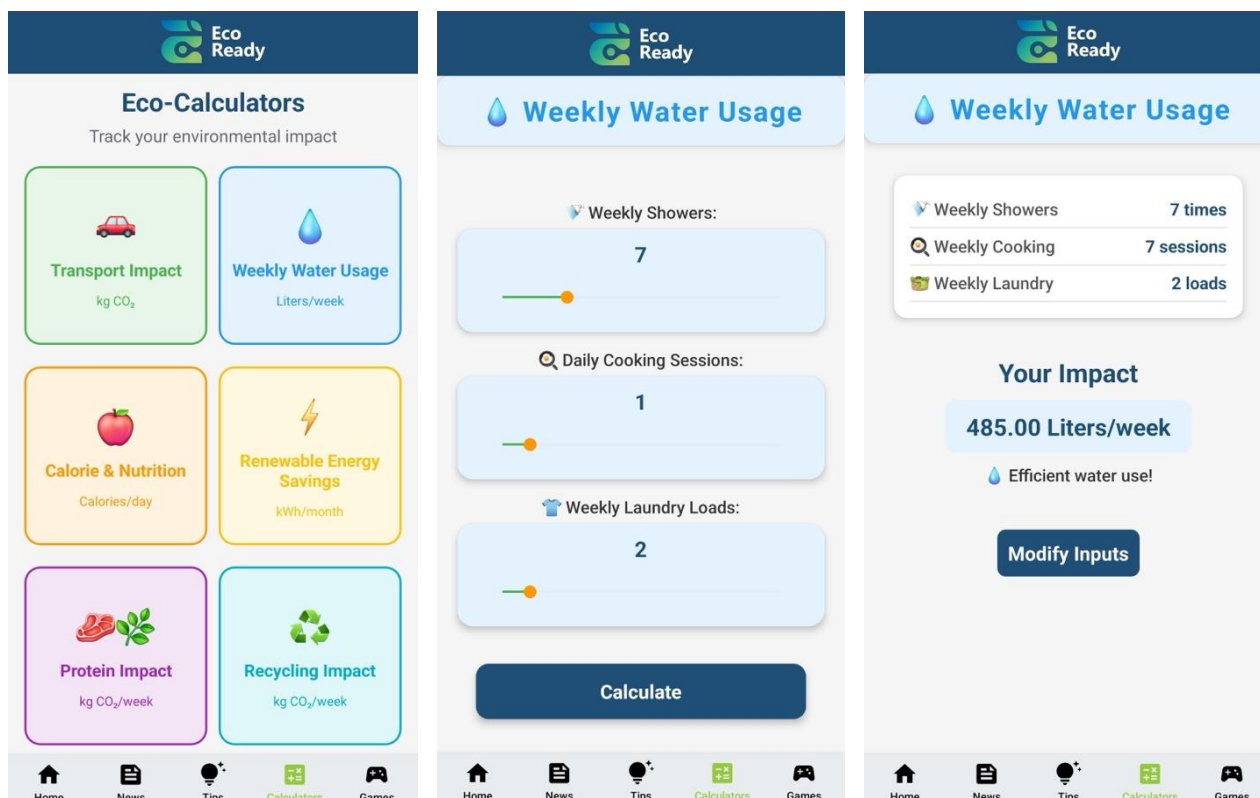


Figure 31 Mobile Application Calculators Screenshots

Interactive Games

The ECO-Ready app makes learning about sustainability engaging and enjoyable with two interactive games:

- **Quiz Game:** Test your knowledge on sustainability topics like food security and climate change.
- **Climate Strategy Game:** Step into realistic scenarios and make strategic decisions that balance environmental impact, food security, and budget constraints.

The games are both fun and educational, making sustainability concepts accessible to users of all ages.

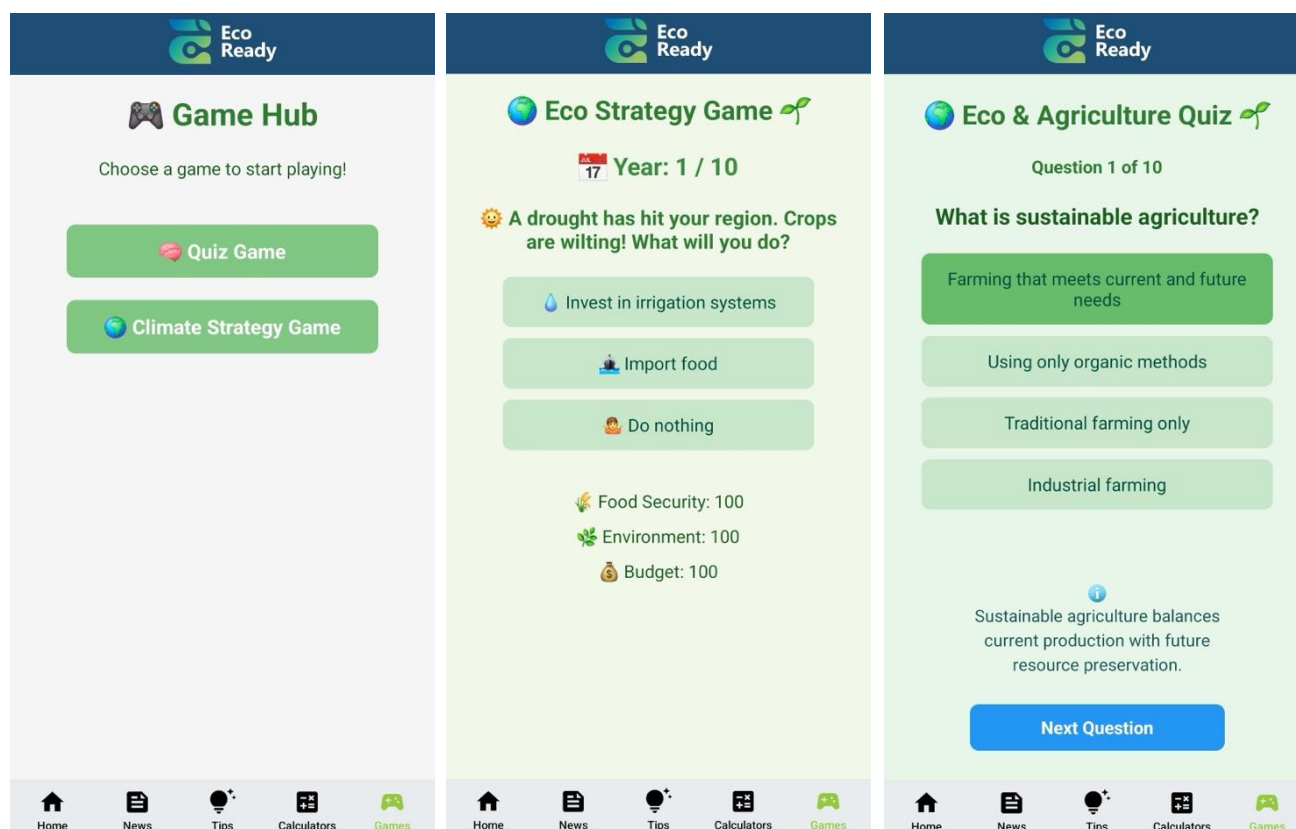


Figure 32 Mobile Application Games Screenshots

References

European Commission's agricultural and rural development department (2024), Agri-Food Markets: Market data on national and European agriculture, available online: https://agridata.ec.europa.eu/extensions/DataPortal/agricultural_markets.html [last accessed: January 30, 2025]

Food and Agriculture Organization of the United Nations - FAO (2024), FAOSTAT Food and Agriculture data, available online: <https://www.fao.org/faostat/en/#home> [last accessed: January 30, 2025]

