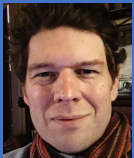




# Low Carbon Regional Energy Planning

*a system to release the value of big data in energy*



Dan Leighton, CEO  
dan@agilis.ai



AGILIS Ai  
solving for x

Low resolution data leads to ineffective action and misdirected investment  
Current carbon accounting **miscalculates emissions by up to 30%**



THE PROBLEM

# Technical Scope & Challenge

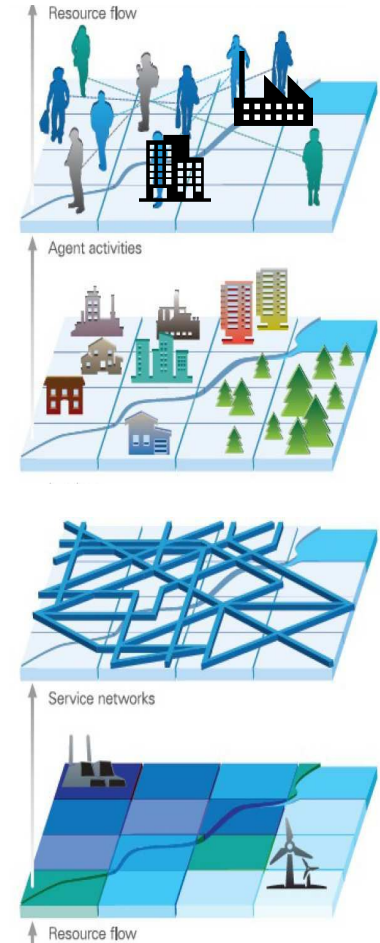
**Regional energy systems need to integrate and analyse many very large data sources to support the zero carbon energy transition by regional grids**

*Electric, hydrogen based transport, heating  
renewable generation, storage and many more*

**This is key to meeting climate and Green Deal goals by 2050**

Investment is being deployed too slowly at only €200B of required €400B each year

**The problem is not the availability of capital but  
the excessively long time it takes to plan Regional Energy Systems**



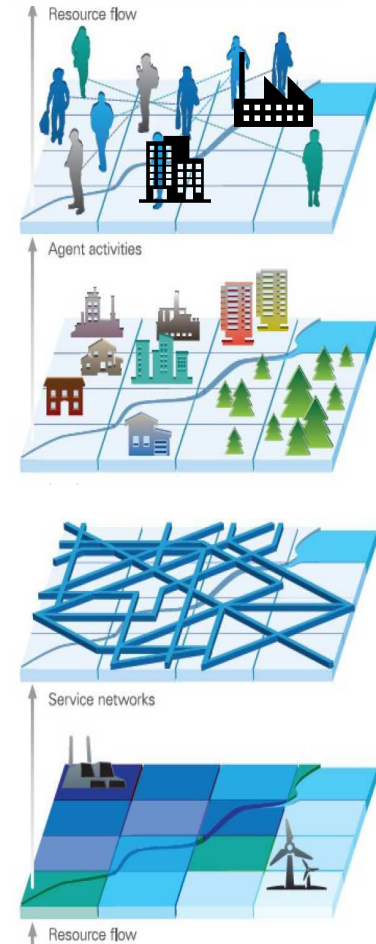
# Technical Scope & Challenge

## We need to de-silo the energy Data Value Chain

*Only then can Regional Low Carbon Energy Systems be built and deployed fast enough to achieve net zero targets*

### This means we need:

- Cleaning and ingesting a very large amount of data both energy (network, demand, generation) and non-energy (socio-demographic, transport, future trends)
- A common analysis framework to access and analyze data - promote best practice in modelling energy transitions
- Visualization interface for stakeholders to collaborate securely and share common data models



# Technical Scope & Challenge

## Service Layer:

Develop a robust application layer for regional assessment of carbon emissions to build a more transparent and accountable energy system

## Geospatial Layer:

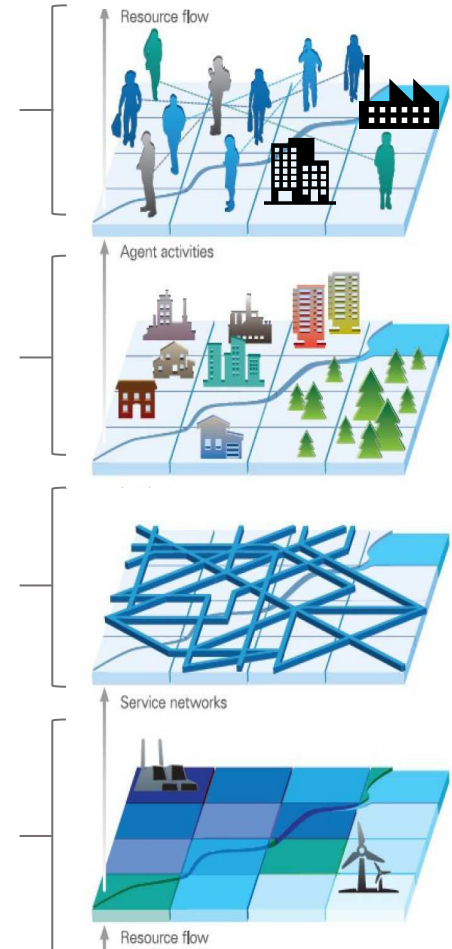
Enrich datasets with high-resolution locational and temporal data to allow planning of low carbon technology rollouts

## Energy Model Layer:

Build a common information model and analysis framework to allow energy models to utilise data in a common format: de-silo the energy Data Value Chain

## Data Layer

Bring together pre-existing datasets to provide a comprehensive model of the key energy domains





**AS PART OF REACH EXPLORE PHASE WE HAVE CONDUCTED INTERVIEWS  
WITH MORE THAN 50 INDUSTRY EXPERTS.**

**THESE ARE THE HARDEST TECHNICAL CHALLENGES THAT THEY IDENTIFIED**

**1. HIGHLY DIVERSE DATA**

APIs, file downloads, CSV, Shapefiles, GeoJson, XML, JSON, Excel

Different coordinates systems and Geospatial references: UK Grid Ref, XY, Lat Lon, WGS84, DHDN

**2. VERY LARGE DATA**

Energy Datasets are hundreds of GBs or TBs

Aggregating, cleaning and querying is very hard requiring experienced software engineers

**3. SLOW DATA**

Data on legacy energy systems often delivered on slow APIs

Often data cannot be queried at the resolution required for energy planning

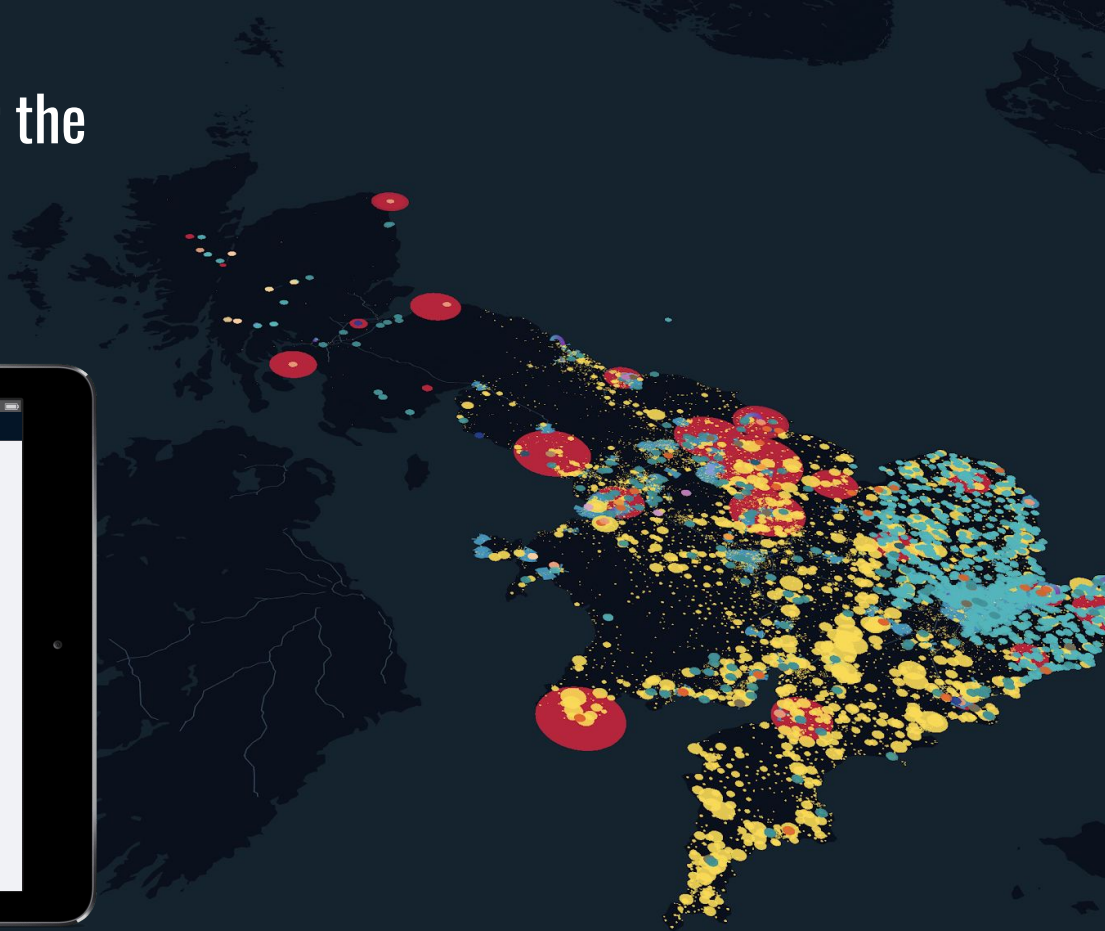
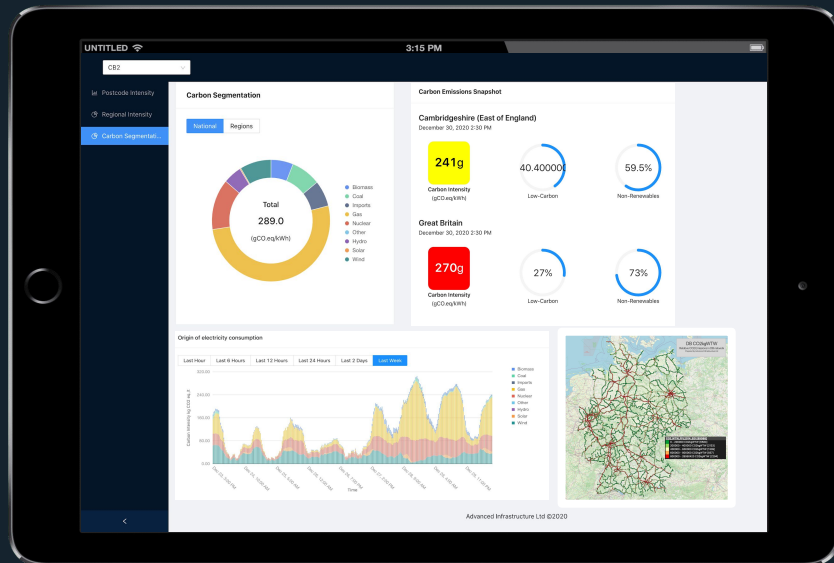
**4. UGLY AND UNAVAILABLE DATA**

Current energy visualisation tools are often not modern or sufficiently flexible to allow experimenting or combining of datasets

Browser limits mean mapping and visualising the data on the web is very challenging



# A big data platform optimised for the energy data value chain



## 2. Selection of algorithms and tools

- We will only use standard well-established open source tools to ensure no bottlenecks of performance or available developer expertise and no licence restrictions
- Our platform will allow integration of any appropriate network and energy flow model algorithms on any modelling platform

### Data Cleanse and Aggregation

#### Data acquisition

Python functions integrated into Apache Airflow and NiFi to acquire energy data from data sources and retain licence and privacy meta data

#### Data transform and load

Python functions integrated into Apache Airflow/NiFi/FiWare Draco to document and perform data ingest, cleanse, provenance and version

#### Data aggregation and query

Storage of aggregated data for fast access e.g. scalable NoSQL, Hadoop/Hive cluster, or optimised SQL for defined datasets

#### Data update

Modularised microservices to update aggregated data from existing and new APIs

#### Data Output

OpenAPI / Swagger definition

React dashboard

Visuals platform: Deck.gl, Kepler.gl, Geomesa

### Energy flow modelling

#### Calculate Generation & CO2 Emissions

At each grid network node

#### Calculate power imbalance

Between exporting and importing regions

#### Calculate inter-regional power flows

Using Three - phase Newton Raphson iteration to calculate AC power flow

#### Calculate power flow carbon intensity

Matrix of carbon intensity balance at bus level

#### Calculate end node carbon intensity

Balanced carbon intensity of power consumed at each node

### Machine Learning

#### Explore and Train

Principal Component Analysis, TSNE Visualisation and Neural Network Analysis on historical data

#### Predictive modelling

Nowcasting of immediate future trends

Historic data training and modelling at hyper local level

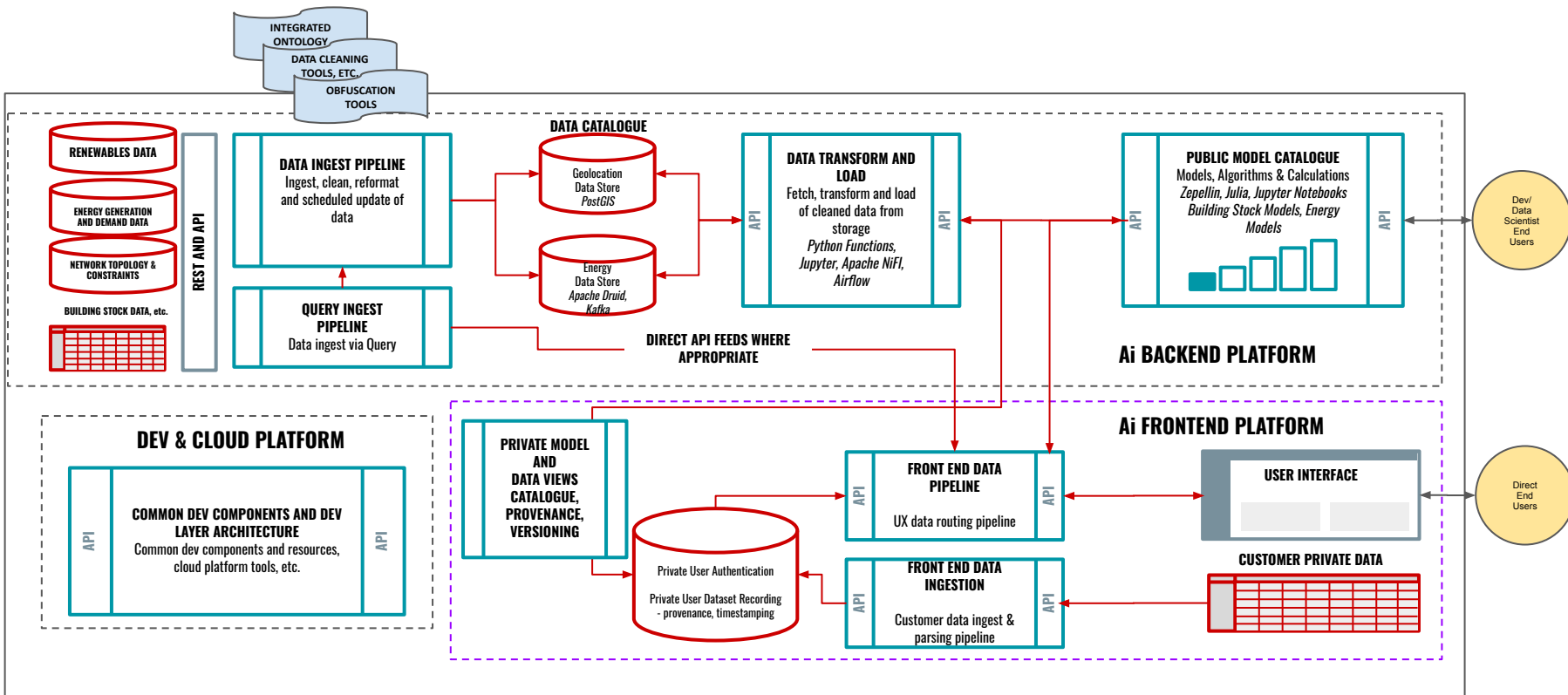




### 3. Scalability & flexibility of the solution: overview

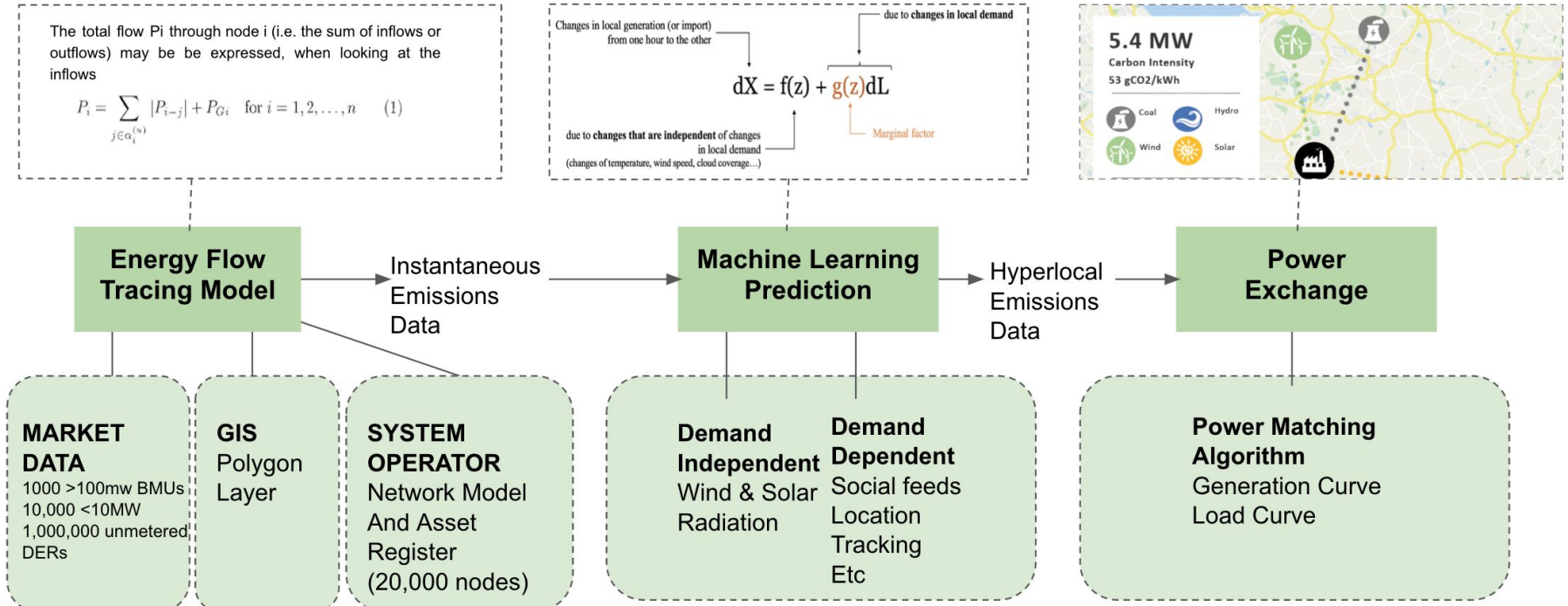
## REACH: CURRENT EXPLORE PHASE PLATFORM ARCHITECTURE AND DATA FLOW

THIS API FIRST ARCHITECTURE HAS BEEN SUCCESSFULLY TESTED AS PART OF THE EXPLORE PHASE TO PROTOTYPE DATA INGESTION PRINCIPLES, MODULAR MODELLING APPROACH AND SCALABILITY OF ARCHITECTURE



# Any algorithm or model can be implemented

- These are a sample of the algorithms that have been tested in POCs



## COMPLETED REACH EXPLORE PHASE TECHNICAL PROTOTYPES

### HIGHLY DIVERSE AND LARGE DATA CHALLENGE ✓

- Successfully tested **Big Energy Data pipeline**
- Successfully **ingested, normalised and cleaned >900GB** of open energy related data in addition to the 5GB+ Solar Energy dataset from our Data Provider
- Successfully **automated cleaning** of data from all target data formats
- Successfully parsed and **converted formats to a common format**
- Identified further toolsets to successfully ingest and **clean any future datasets**

### SLOW, UGLY AND UNAVAILABLE DATA CHALLENGE ✓

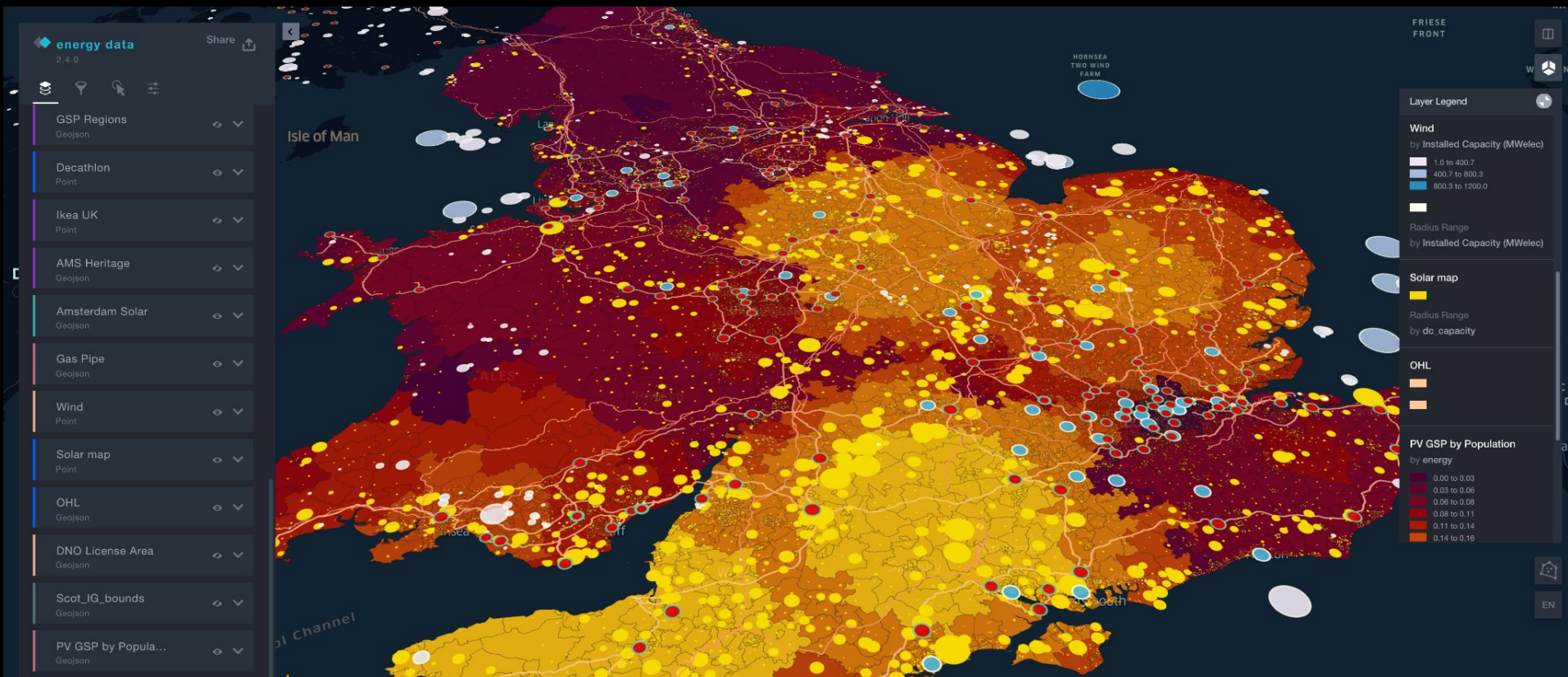
- Successfully allowed **sub-second query and display** of big energy data
- Access speed to Regional and National carbon intensity data **300% faster**
- Successfully created initial **energy models for Solar Power distribution**
- Successfully user tested **visualisation platform** with end users





# STATUS OF PROTOTYPE IN REACH EXPLORE PHASE

We have successfully demonstrated a proof of concept distribution function of our data partner's National Solar Generation Data against population level data to 5000 person census groupings. Our proof of concept architecture queries via our API to extract 100 energy objects from a >5GB dataset to calculate and return the mean distribution of energy for >5300 regions to the visualisation in less than 1 second. We aim to use the lessons learnt to conduct queries of up to 18000 items at similar speed.



# END USER MOCKUPS



# Mockup - End User UI design: USER WORKSPACE

Dashboard

Layers

Projects

Impacts

Tools

Baseline

EPC

Heat demand

Fuel poverty

DFES

SP

CR

DFES

Projects

Project 1

Project 2

...

Carbon

Date Slider

Latest uploads: xxxxxxxxxxxx (link)

Project 1  
Report  
View | status

Project 2  
Report  
View | status

Baseline  
Report  
View | status

Carbon  
Report  
View

Regional view  
Report  
View

National view  
Report  
View

Saved view 1

Saved view 2

Saved view 3

# Mockup - End User UI design: AVAILABLE DATA LAYERS VIEW

Dashboard

Layers

Projects

Impacts

Tools

Housing

EPC

Heat demand

Fuel poverty

Network

Topology

Capacity

DFES

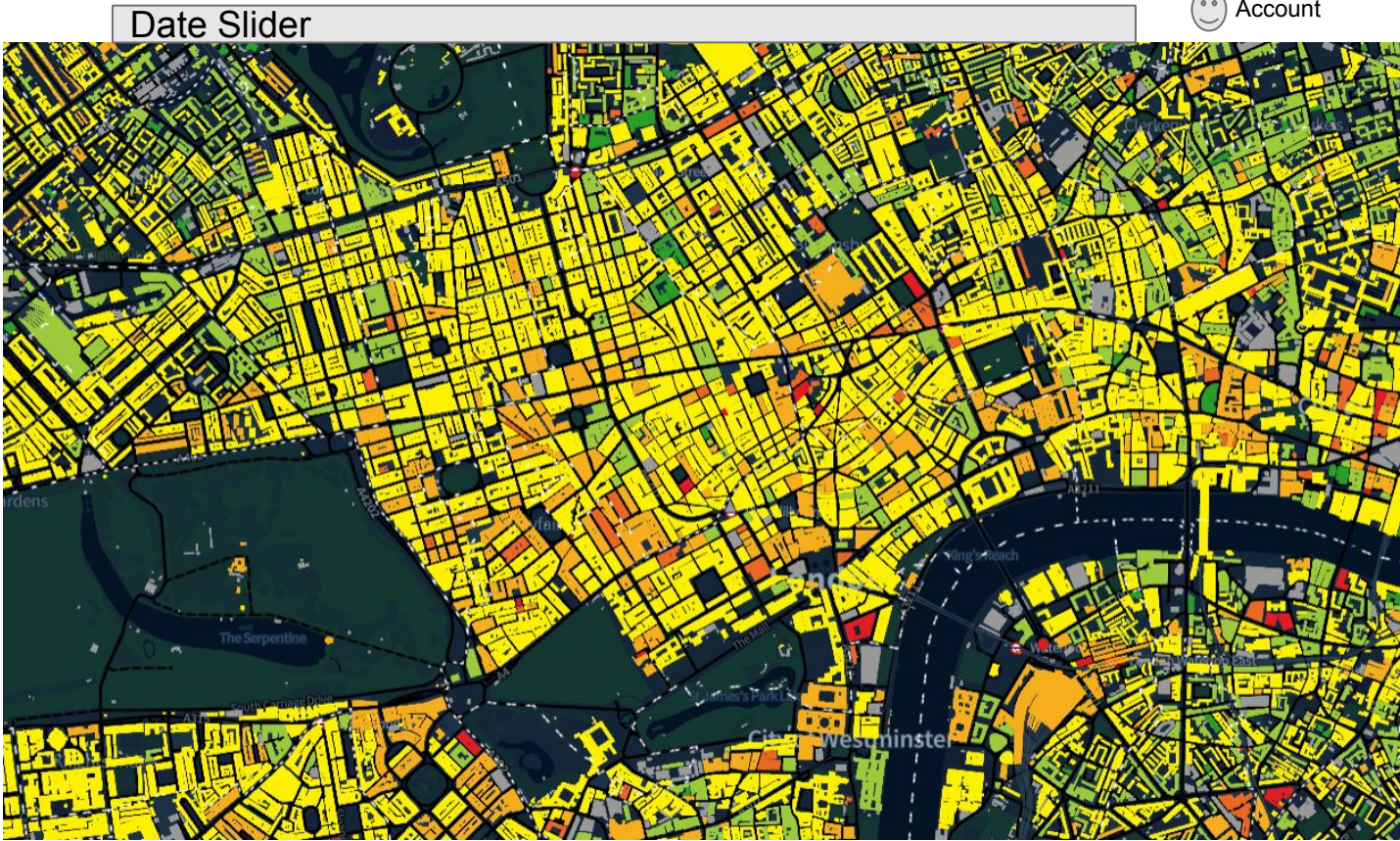
Low carbon technologies

Solar PV

Heat Pumps

DSR

Gas networks





# Mockup - End User UI design: PROJECTS VIEW

Dashboard

Layers

Projects

Impacts

Tools

Project 1

Export

Rooftop PV

Batteries

Night storage heaters

Project 2

...

...

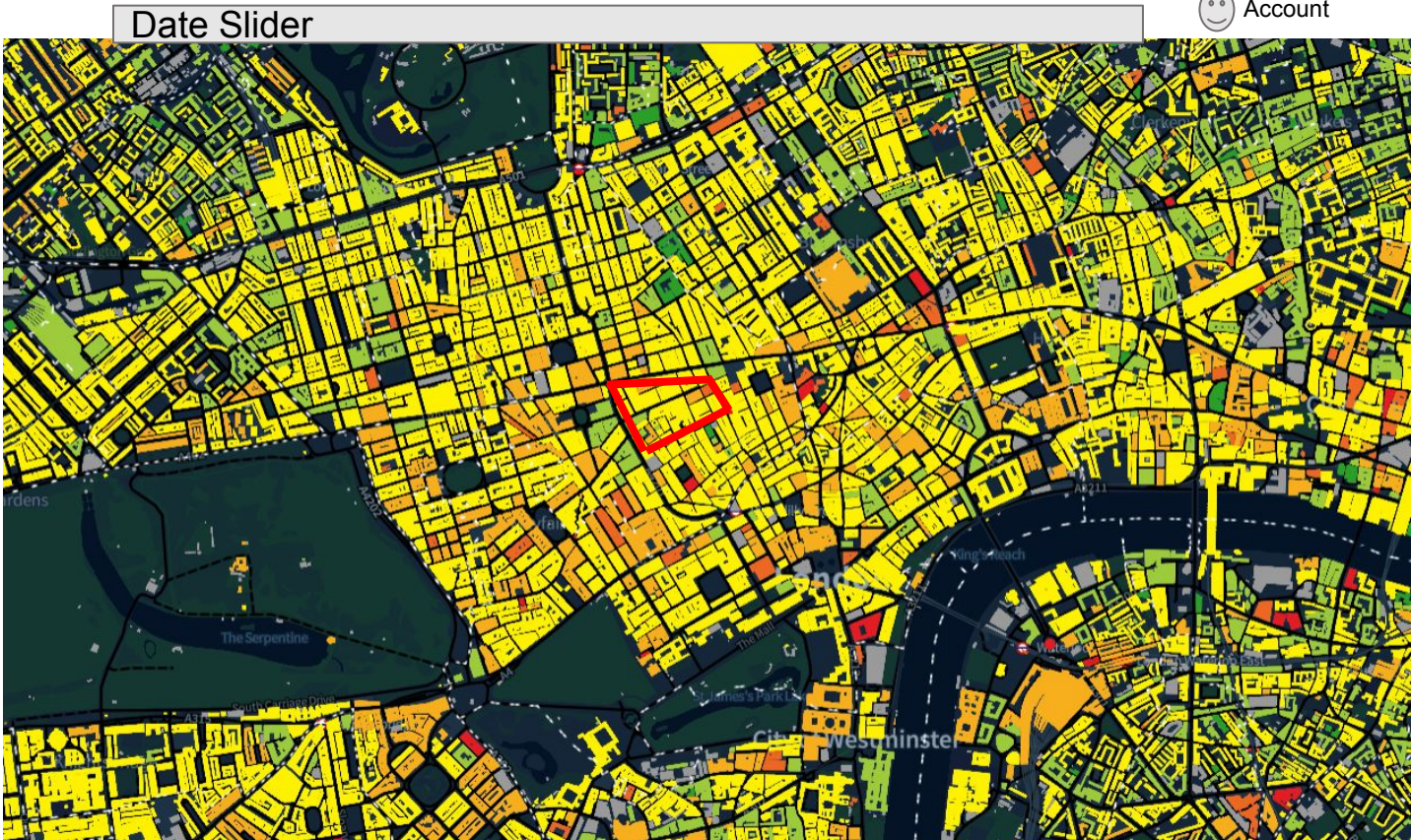
...

Project 3

...

...

....





# Mockup - End User UI design: BASIC MODEL ACCESS

Dashboard

Layers

Projects

Impacts

Tools

Project 1

Show

Rooftop PV

Batteries

Night storage heaters

Project 2

Show

...

...

...

Project 3

Show

...

...

Date Slider

Project 1 Settings

Rooftop PV

Show on map

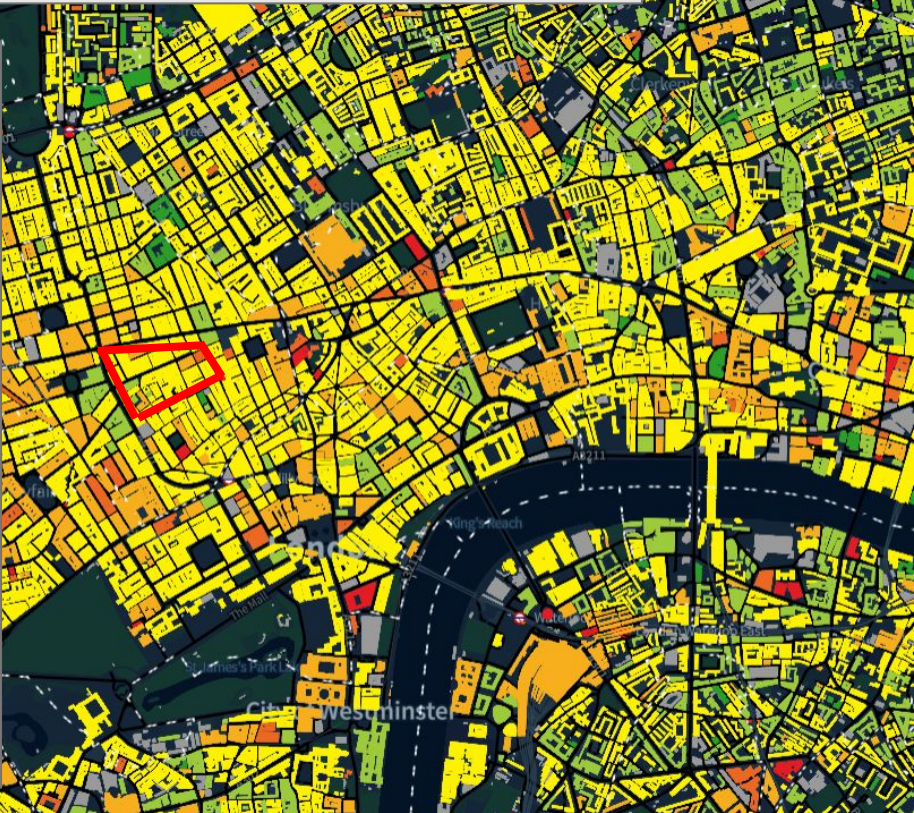
Apply rooftop PV to:

Logic 1

Logic 2

Logic 3

Show impacted network path





# Mockup - End User UI design: IMPACT/OUTCOMES VIEW

Dashboard

Layers

Projects

Impacts

Tools

Baseline

EPC

Heat demand


Fuel poverty

My Projects

Projects 1

Project 2

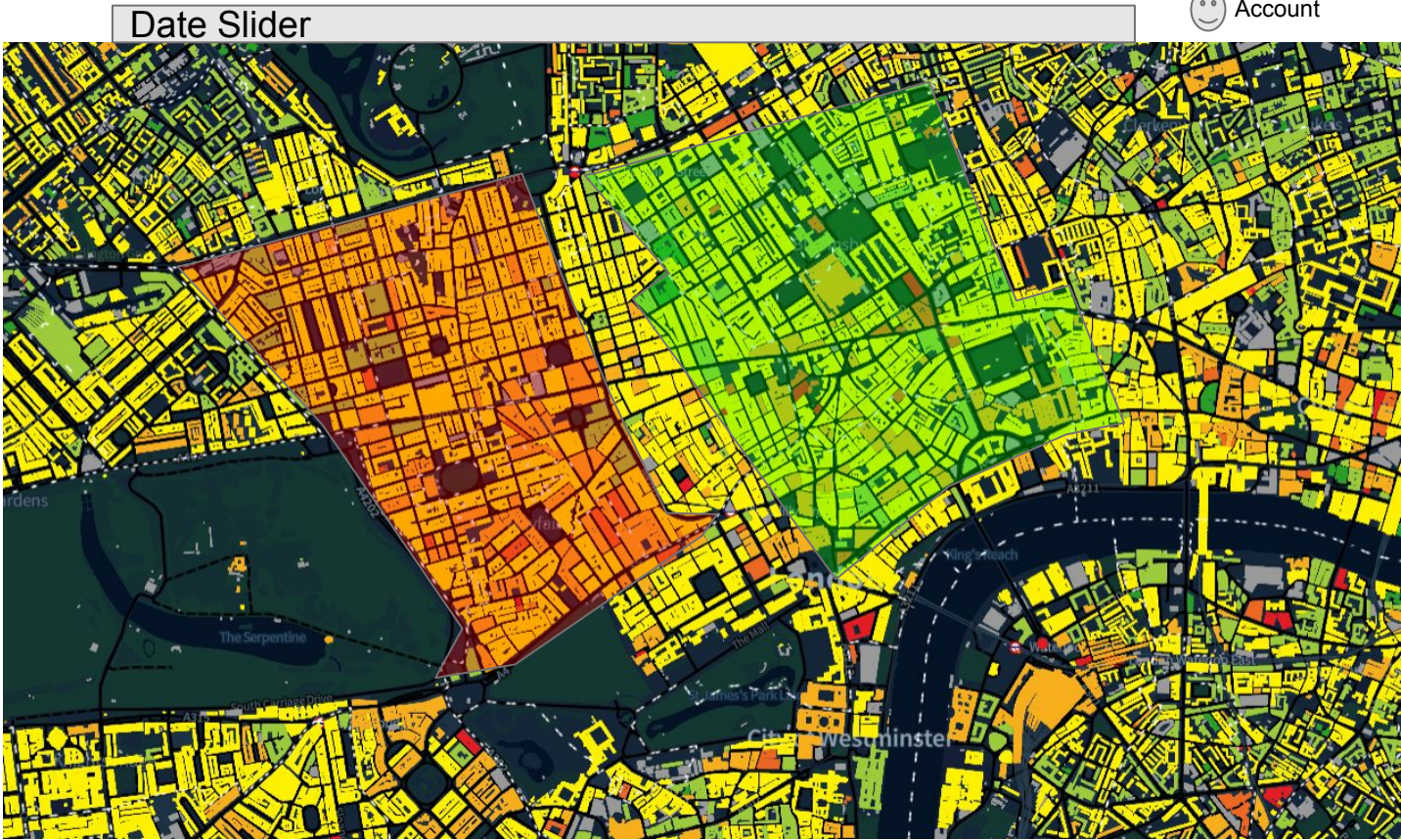
....

Show All Projects

Project 2

Outcomes

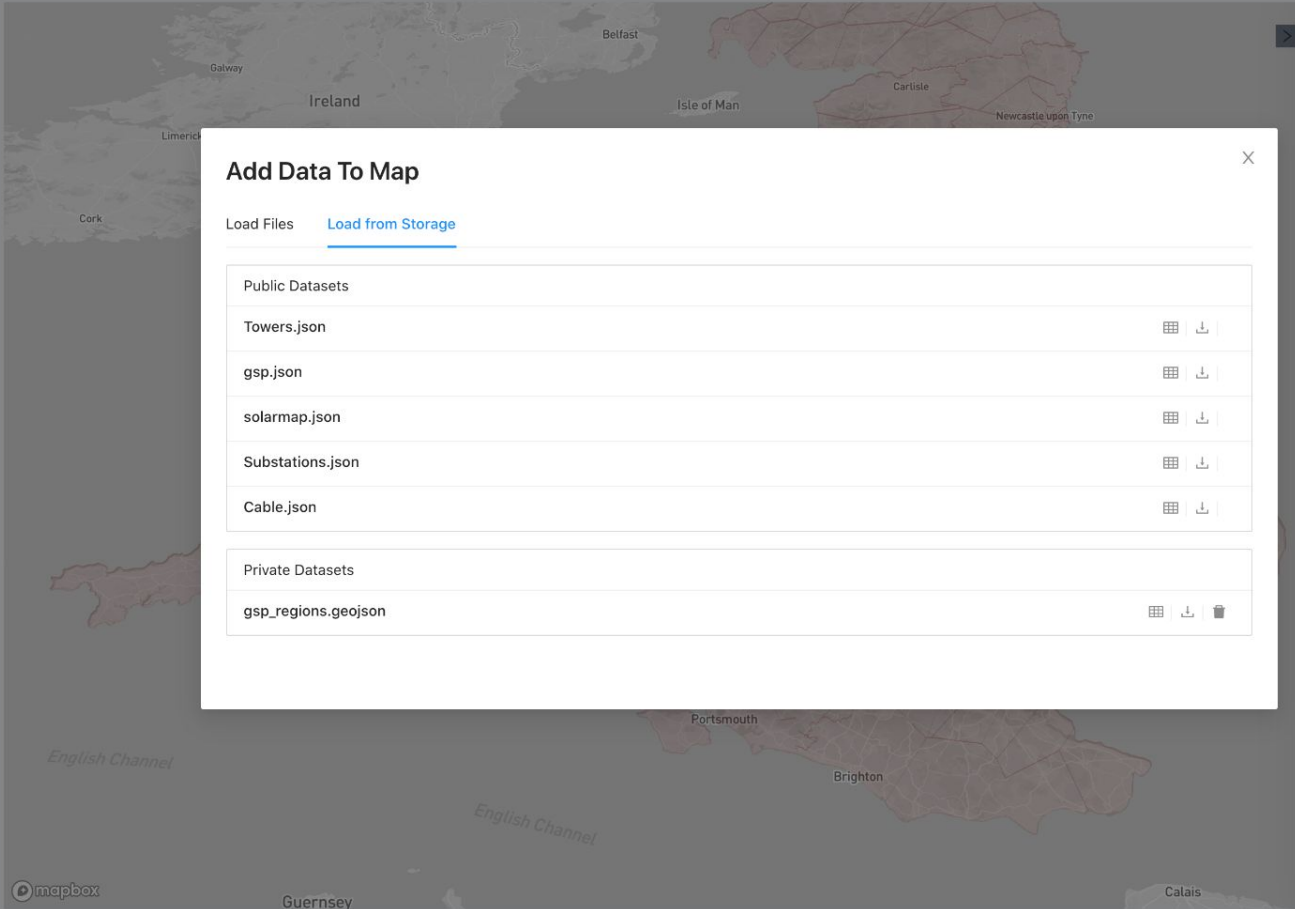
Carbon





# **INTEGRATED DATA LAYER AND MODELLING SYSTEM PROOF OF CONCEPT**
















- Dashboard
- Analysis
- Monitor
- Map
- DeckGL
- Workplace
- Form
- List
- Profile
- Result
- Exception
- Account
- Graphic Editor



## Add Data To Map


Load Files [Load from Storage](#)

### Public Datasets


Towers.json			
gsp.json			
solarmap.json			
Substations.json			
Cable.json			

### Private Datasets

gsp_regions.geojson			
---------------------	---	---	---

- 
- 
- 
- 

## Layers

gsp\_regions.geojson private 

779 rows

+ Add Data

gsp\_regions.geojson

+ Add Layer

Dashboard

Analysis

Monitor

Map

DeckGL

Workplace

Form

List

Profile

Result

## Marketplace



**Jupyter Notebook** 0.0.1

Advanced Infrastructure

This extension allows you to link your layers to a jupyter notebook.

Remove



**Julia Notebook** 0.0.1

Sheffield University

This extension uses the Anymod Energy Modelling system to predict building stock energy usage

Get



**Obfuscation of Data**

Advanced Infrastructure

This extension allows any dataset to be anonymised to allow it to be published publicly

Get



## Layers

gsp\_regions.geojson

private



779 rows

+ Add Data



gsp\_regions.geojson



Filter this dataset where:



Pseudocode: GSP Region = GSP is North of the Watford Gap AND GSP AVG Demand > 23000 MW

Operate on this dataset:

Pseudocode: Adjusted GSP AVG Demand = (Current GSP AVG Demand \* 95%)

Apply Filter and Operation to Map

Save this Filter and Operation Privately

Publish this Filter and Operation For others to use

Connect this dataset to a notebook/code/api



Export/Add this Dataset/Map to a Report



Expose this dataset as an API

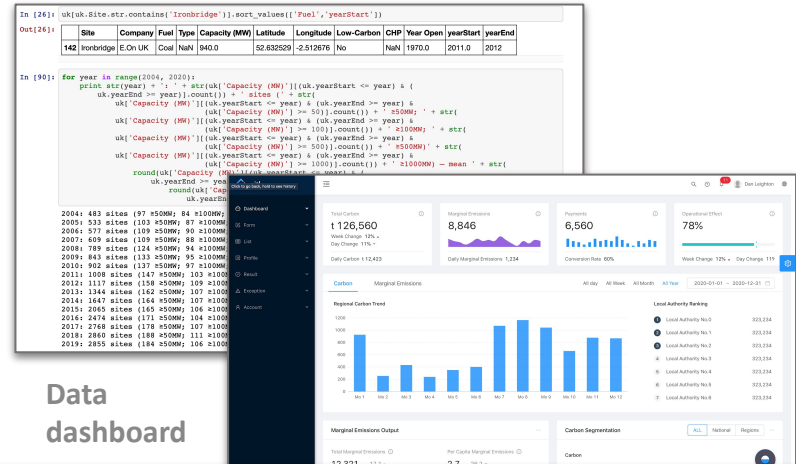


# Status of Prototype and Mockups

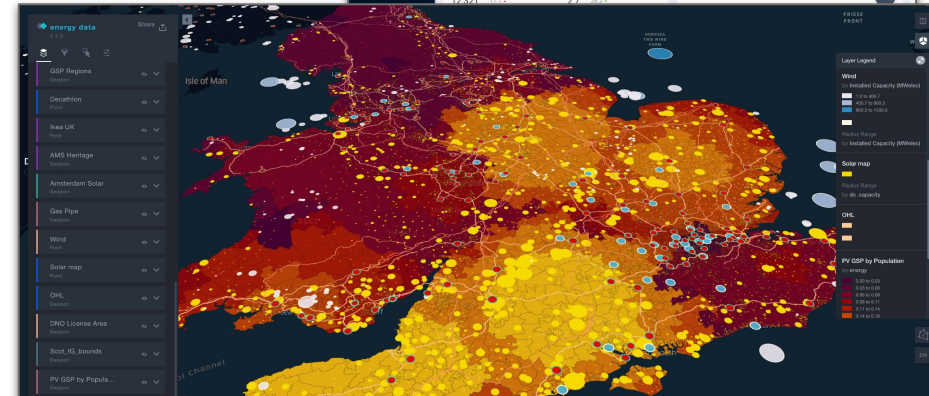
## Completed stages

- Critical data sources for aggregated big data set identified and initial data transform and load pipelines built
- Prototype Data Partner Solar Energy distribution model built
- MVP outcomes identified through extensive user research
- Mockups built and tested for feedback with both Planning level End Users and Data Scientists
- First customers identified
- First iterations completed of:
  - Visualisation platform
  - Data transform pipeline
  - R statistical tool deployed as a scalable API
- Clustering software built and deployed
  - Software deployment platform
  - API deployed Kubernetes clusters
  - Unlimited server scaling
  - Allows own custom software to be deployed at scale
- FiWare, GCP and AWS platforms assessed
- Prototype of integrated data model and visualisation system built

## Data aggregation



## Data dashboard



## Visualisation platform



# TO COMPLETE IN REACH EXPERIMENT PHASE

- Assess *scalability* potential of **energy data modelling system** - *test* with energy data scientists
- *Identify* testable **energy data ontologies** - integrate and *test* with energy planners
- *Integrate* **time-series data optimisation** - test speed and throughput requirements with end users
- *Integrate* **streaming of geolocation data** - *test* visualisation of geolocation data with end users

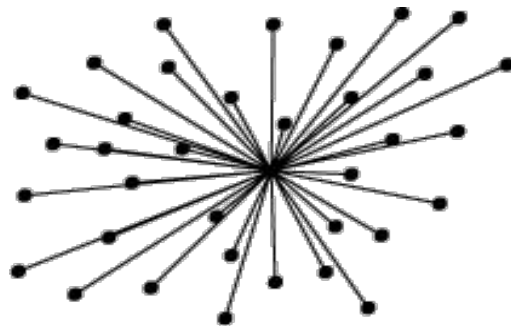


### 3. Scalability & flexibility of the solution

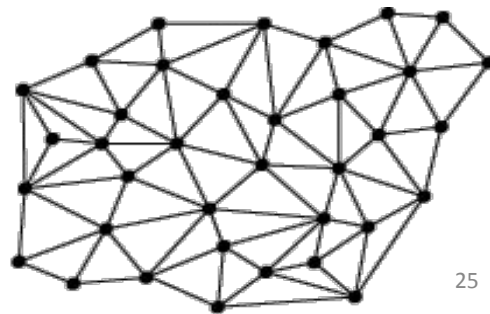
*What are we planning that is different?*

- 1. Planning for scale from the start using modularised microservices architecture*
- 2. Optimising data search and aggregation of against appropriate data structures*
- 3. Building in data streaming and messaging capabilities as early as possible to solve SQL bottlenecks*
- 4. Building for commercial viability and utility from day one*

**Moving from monolithic data structures  
difficult to scale**



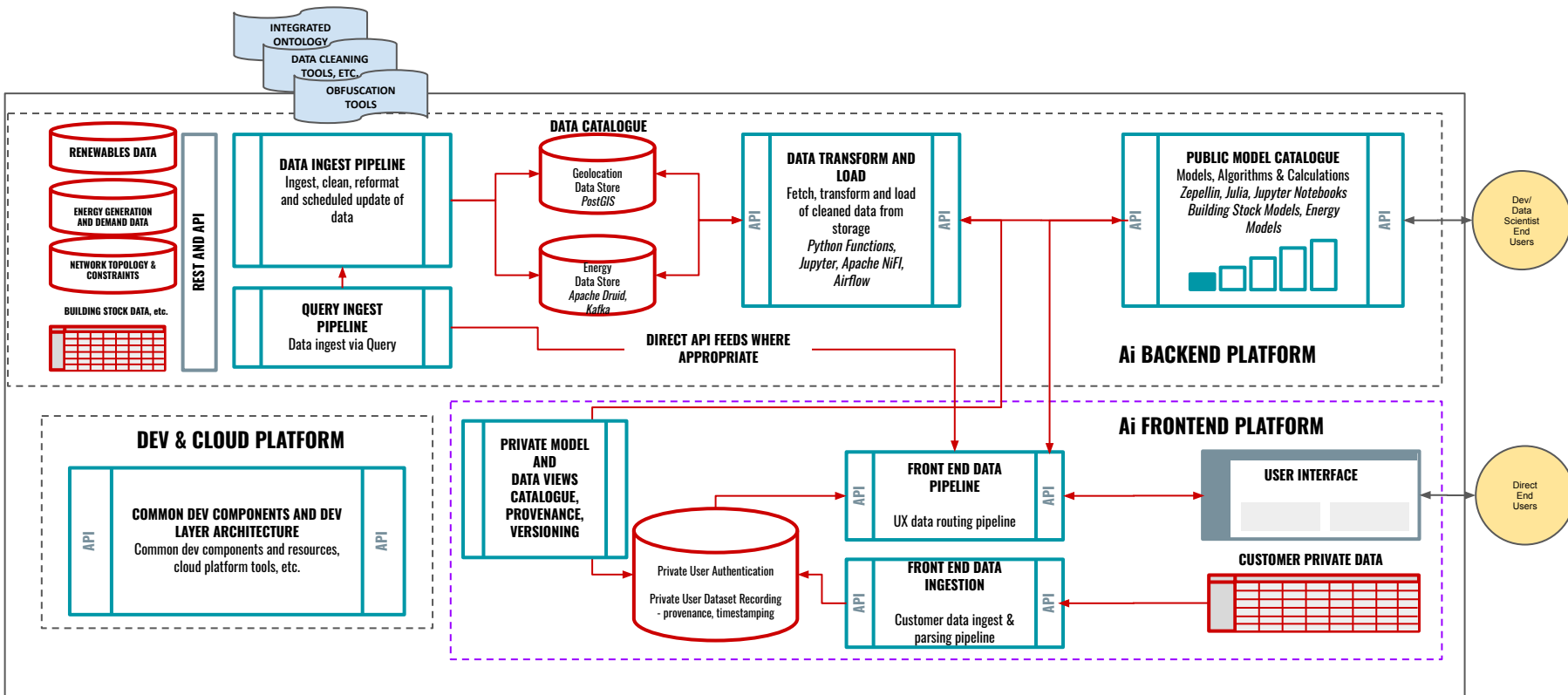
**To distributed data structures that scale  
on demand**



### 3. Scalability & flexibility of the solution: overview

## REACH: CURRENT EXPLORE PHASE PLATFORM ARCHITECTURE AND DATA FLOW

THIS API FIRST ARCHITECTURE HAS BEEN SUCCESSFULLY TESTED AS PART OF THE EXPLORE PHASE TO PROTOTYPE DATA INGESTION PRINCIPLES, MODULAR MODELLING APPROACH AND SCALABILITY OF ARCHITECTURE



## 4. Data security and legal compliance



### **Authentication and authorisation:**

All users of the system will be required to use two factor authentication systems to access data and will be required to provide KYC identification to allow access to sensitive data.

Developers are not given access to core data.



### **Compliance:**

We recognise that whilst our data sets do not hold personally identifiable information that would be subject to GDPR regulation, we do store strategically sensitive information that we have a duty of care to protect. If mishandled, data of this nature could have lasting ramifications.



### **Encryption and Separation of Concerns:**

All data on our systems is encrypted by default both in transit and at rest. Our unique Kubernetes deployment platform separates application layer from customer data and customer configurations to ensure total sandboxing of all customer data with no access by engineers to private customer datasets beyond what is needed to provide the service.



### **Ethics:**

The responsible use of algorithms and data is paramount for the sustainable development of machine intelligence applications. We adhere to the UK's Digital Catapult Ethics Framework and the European Ethics Guidelines for Trustworthy Ai

## 5. Quality assurance and Risk management

**ISO 31000 Risk Management** practices will be used throughout this project

**A dynamic risk register** utilising risk screening will be used to monitor the project, adhering to ISO 31000 principles.

**Risks and critical path analyses** have been conducted and will be reviewed at fortnightly meetings, led by our risk manager to ensure active risk monitoring, mitigation measures, root cause analysis as necessary and proactive minimisation of knock-on delays. The top five technical risks are shown (right)

RISK DESCRIPTION				MITIGATION			
Unsuitability of proposed Data Architecture due to inability to meet user needs leading to low user adoption	3	3	9	RETENTION: Risk of unsuitability is the purpose of the project REDUCE: Explore a wide range of architectures during the feasibility phase	2	2	4
Compute solution requires unjustified power resources and impact on carbon emissions. Environmental impact precludes wider adoption beyond trial.	3	3	9	REDUCE: Use optimised compute on massively scalable clusters to reduce compute time. Optimise algorithms and cloud storage to reduce requirements of data storage, machine learning training and caching.	1	3	3
Software bugs cause system interoperability	2	3	6	MITIGATION: Weekly code reviews, extensive testing, patient/client bug reporting functionality, quality assurance mechanisms.	1	1	1
Poor UX experience causing users to disengage or misuse software. Causing lower revenue and reputational risk.	2	3	6	REDUCE: Deploy extensive user research and testing. Employ UI/UX principles.	1	3	3
Scaling leading to decreasing architecture resilience with increasing users.	2	3	6	REDUCE: Extensive real time testing as users increase. Rollback ability.	1	3	3
Incomplete data sets result in reduced ability to fully test architectural options ready for Alpha Phase, resulting in incomplete design.	3	2	6	REDUCE / SHARE: Use stakeholders or private data providers to secure more data sources, prioritise data analysis further, to focus on most critical architectural components, build in flexibility to Alpha stage to plug gaps in data analysis and test other versions in prototyping stage	2	2	4

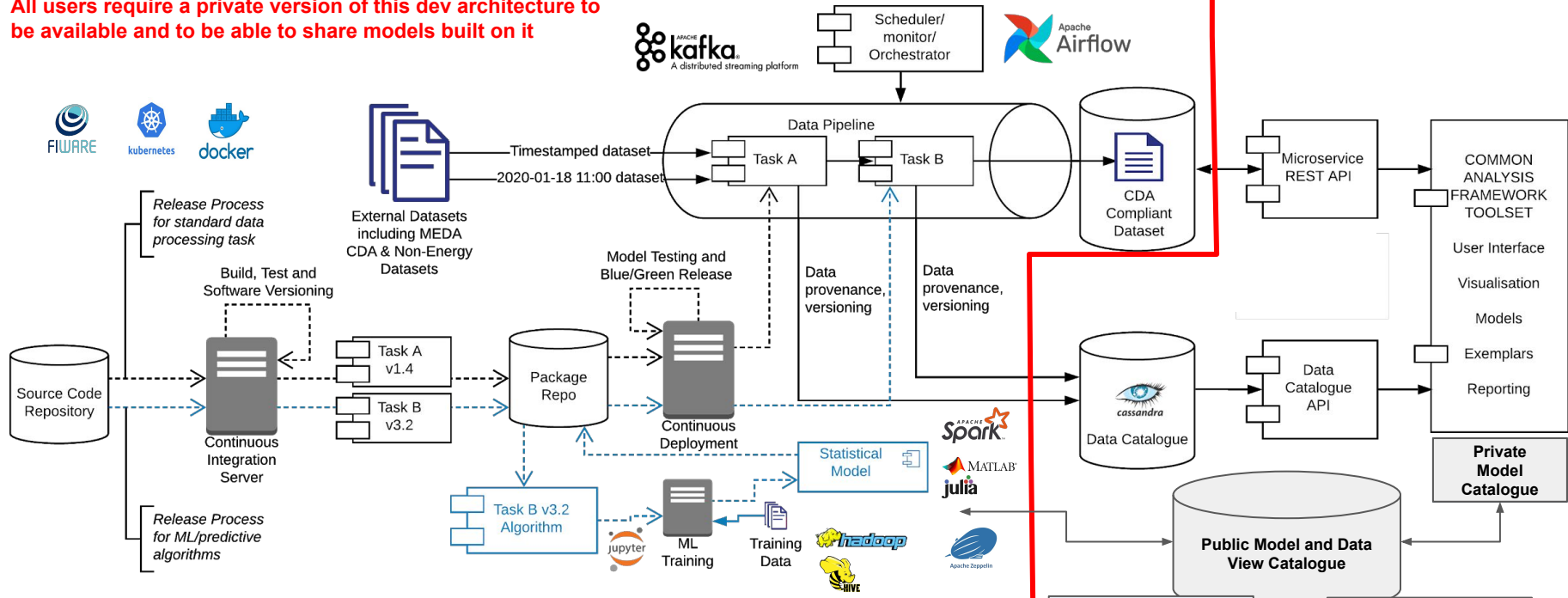


# QUESTIONS

# APPENDIX

### 3. Scalability & flexibility of the solution: detail

All users require a private version of this dev architecture to be available and to be able to share models built on it



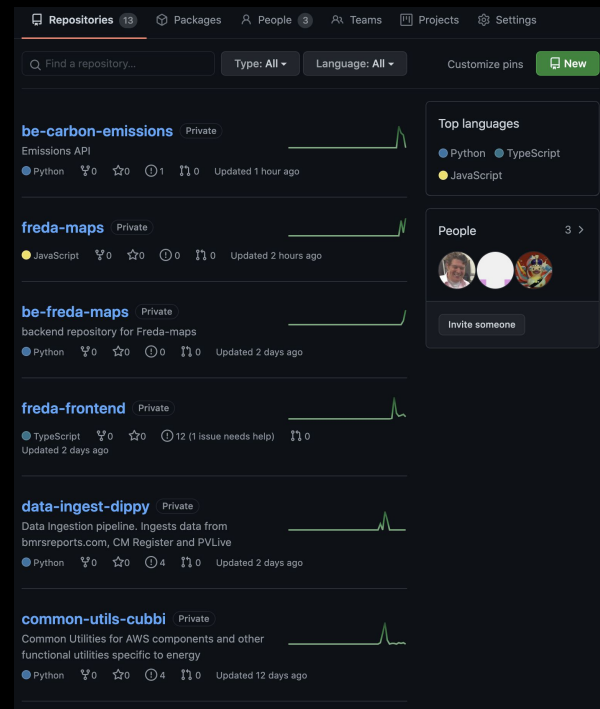
#### Notes on CI/CD Pipeline and Software Development Practices

We use Git Repos (currently Github) to store all code. All code is deployed with a Dev flag initially. This flag is recognised and triggers both unit tests and a code review by senior devs. Once unit tests are passed, code review is conducted. Since all code is developed as isolated modular functions, this allows code to be deployed once test and functionality is confirmed. See the following two slides for further details.

# Software Development

## *Our Professional Principles and Practice: Overview*

- Everything is an API
  - Each module has its own repository
  - Each module has its own API
  - All code can be developed completely independently to allow iterative improvement of isolated chunks
- Separate the concerns
  - Make improvement easy. Model and compute via API: calculate at the backend, display at the frontend
  - Separate code into controllers, utilities and handlers
- Type is nice
  - Use TypeScript and heavy software linting to ensure good quality code
- No-one gets to push their own code to production
  - Code cannot be pushed to production without review by another senior team member - no exceptions

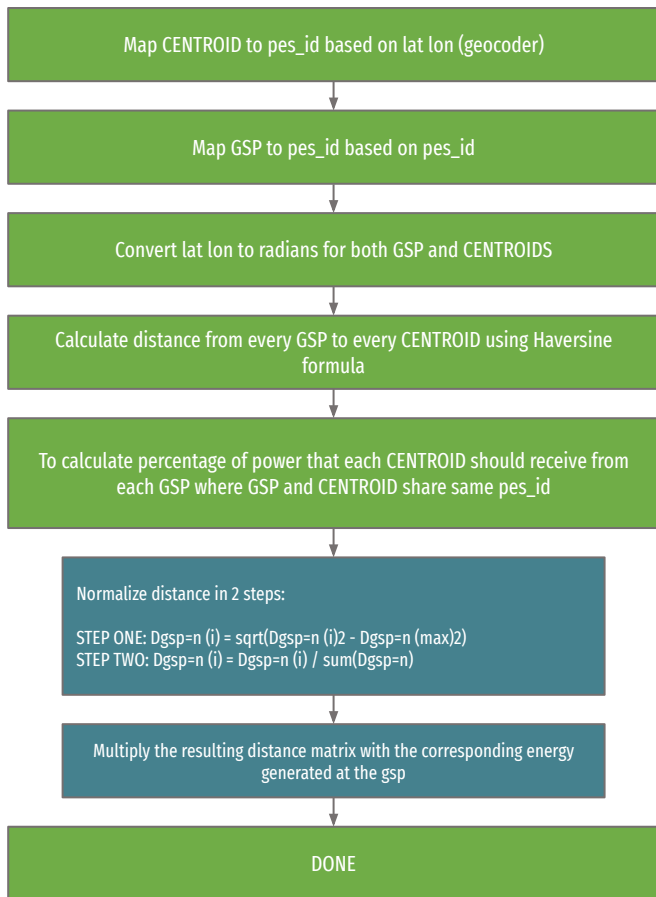


# Software Development Principles: Detail

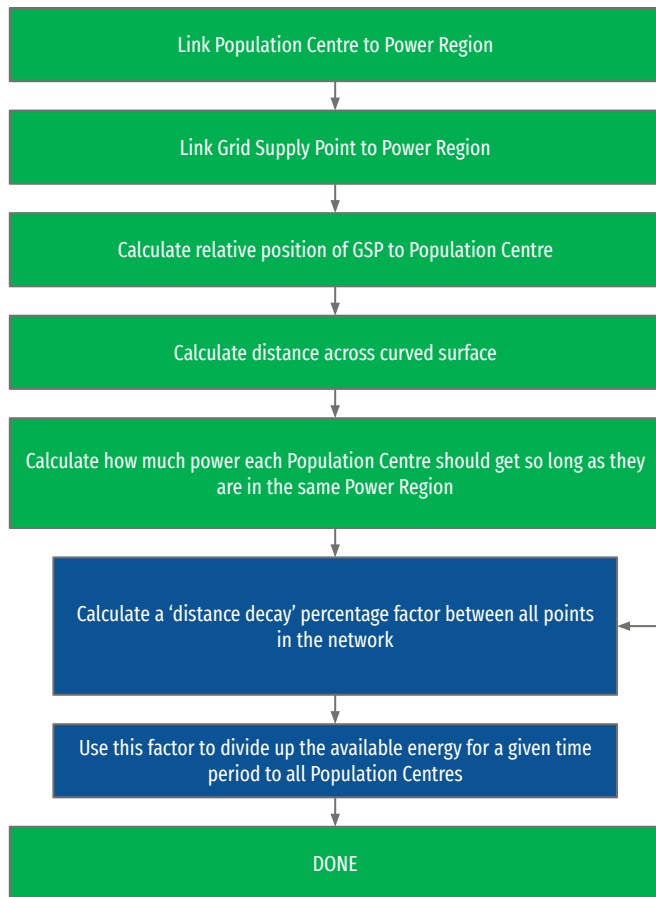
1. **Requirements gathering** - Project managers gather requirements from the team understanding the user story - which is the end goal, not a feature, expressed from the software user's perspective. A user story is an informal, general explanation of a software feature written from the perspective of the end-user or customer.
2. **Architecting the requirement** - A solutions architect from the team writes down a "technical specification" of the user story. In this process, the architect assesses potential changes to existing components, databases and interfaces. The architect ensures that there is either data or code redundancy. The goal of this process is to break down the user story into granular tasks that are maintained as Github Issues and are coded according to priority.
3. **Development Lifecycle** - The developers refer to issues assigned to them and estimate a time for completion. This data is aggregated by the project managers helping them properly plan the next set of requirements or manage releases to clients. All code is maintained on github and pushed to a branch periodically. The developers not only write the code for the module but also write the unit tests and the integration test. When the code is ready to be deployed, developers create a "pull-request" requesting a senior developer to review their code. Pull-requests are usually linked to an issue so the reviewer has all the information they need to conduct a thorough review. All backend and front end modules follow the principle of "separation of concerns" that distributes the program to distinct sections such that each section addresses a separate concern. These are the steps followed by the developers while coding:
  - a. **Code structure** - The developers are responsible to maintain their code in
    - i. **Controllers** - Controllers define the main intent of the feature. The "core-logic" is defined in controllers.
    - ii. **Utilities** - Utilities are functions that are required throughout the feature and may be used by one or more controllers. This reduces code redundancy.
    - iii. **Handler** - A handler is an abstraction wrapper that processes a request, executes an appropriate controller and returns a response. Handlers are mostly used in writing APIs
  - b. **Modularity and reusability**- The developers are required to maintain their code in a way where it can be visualized as a class or a function. A function that is used more in more than one place is generally placed under "Utils" but if the particular function is used in more than one API, it is moved to "common-utilities". Common utilities are a set of classes and functions that are used by almost all frontend or backend APIs. This ensures that developers do not re-write code that should be standardized.
  - c. **Linting** - We use ESLint and pyLint that are integrated into the IDEs. This helps developers spot errors and fix them before deploying code. The same tools are setup in the automated pipeline. When developers push (commit) code to their branch there is a lint check that is performed. The same tests are run when the reviewer approves code to be merged into the master branch. This ensures that when the code is merged into master, there are no errors or unused definitions.
  - d. **Tests and coverage** -Developers maintain two tests for their module (Unit Test - a test for one functionality or module and an Integration Test - which is a test for the entire functionality) In these tests, developers write not just positive or happy cases but also show proof that the code is written can handle exceptions and errors without halting execution through writing negative cases. During the review, the reviewer checks the tests to ensure that all cases are covered. The automated pipeline is designed to conduct tests everytime code is merged to master. A summary of the report is created that shows the "coverage" of the tests written for the functionality. If the coverage is low (this happens if the developer did not address all cases on their tests) code cannot be merged by the reviewer.
  - e. **Updating issues** - The last step of the development process after a PR is merged is to update the issue back on Github where the developers are required to put down the commit that solved the respective issue. This ensures documentation is consistent and all issues can be directly linked to a commit id
4. **Dev Deployment** - When code is merged to master, an automated pipeline is triggered that runs tests, checks for lint issues and packs all requirements and deploys it to the serverless function. At this point, the feature is available for the product team to test.
5. **Production Deployment** - After the product team and the testing team has tested the feature built on development thoroughly the code progresses through other respective environments (Dev → UAT → SIT and finally Production)



## ALGORITHM



## EXPLANATION



## METHODOLOGIES

# EXAMPLE PROTOTYPE SOLAR POWER DISTRIBUTION MODEL

POC: CAN BE REPLACED BY ITERATIVELY  
BETTER MODULAR MODELS SUCH AS A  
KRIGING FUNCTION

# Emissions API Definition Prototype

## Created for REACH Explore

### Carbon Emission Controllers

- 1. Marginal Intensity Controller
- 2. Regional Intensity Controller
- 3. Plant Ranking Controller
- 4. Regional Ranking Controller

#### Marginal Intensity Controller

```
"datetime" : {
  "from" : "2014-01-01",
  "to" : "2014-12-31"},
"controller":"marginal_intensity",
"metric" : "mean",
"aggregate_by" : "month",
"data":{
  "region_id":"1"
}
}
```

##### Input Parameters

```
datetime : { "from" : DateTimeRange, "to" : DateTimeRange }
controller : "marginal_intensity"
metric : avg or sum or min or max or std or var
aggregate_by : month or year or day
data : { "region_id" : int (range - 0,17)}
```

#### Plant Ranking Controller

```
{"datetime": {
  "from" : "",
  "to" : ""},
"controller" : "most_common_plant"
}
```

##### Input Parameters

```
datetime : { "from" : DateTimeRange, "to" : DateTimeRange }
controller : "most_common_plant"
```

#### Regional Intensity Controller

```
"datetime" : {
  "from" : "2014-01-01",
  "to" : "2014-12-31"},
"controller":"regional_intensity",
"metric" : "mean",
"aggregate_by" : "month",
"data":{
  "region_id":"1"
}
}
```

##### Input Parameters

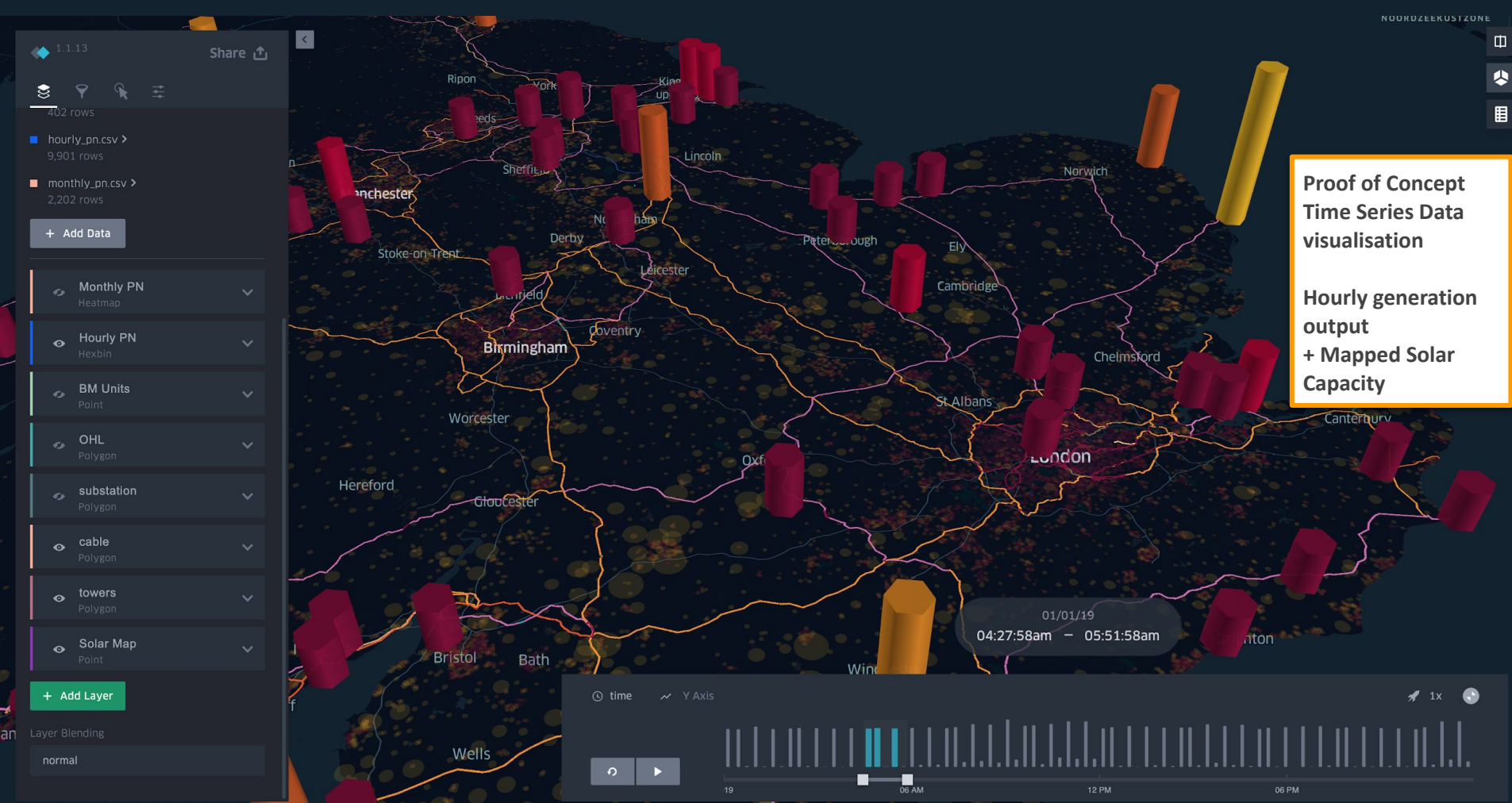
```
datetime : { "from" : DateTimeRange, "to" : DateTimeRange }
controller : "regional_intensity"
metric : avg or sum or min or max or std or var
aggregate_by : month or year or day
data : { "region_id" : int (range - 0,17)}
```

#### Regional Ranking Controller

```
{
  "datetime" : {
    "from" : "2014-01-01",
    "to" : "2019-01-01"
  },
  "controller" : "regional_ranking"
}
```

##### Input Parameters

```
datetime : { "from" : DateTimeRange, "to" : DateTimeRange }
controller : "regional_ranking"
```



substation

Polygon

OHL

Polygon

Time Series

Point

Basic

Point

Fill Color

Outline

Radius

Label

+ Add Layer

Layer Blending

normal

02/23/18

10:25:26am - 11:21:19am

time

Y Axis

23

12 PM

Sat 24

12 PM

Fet

