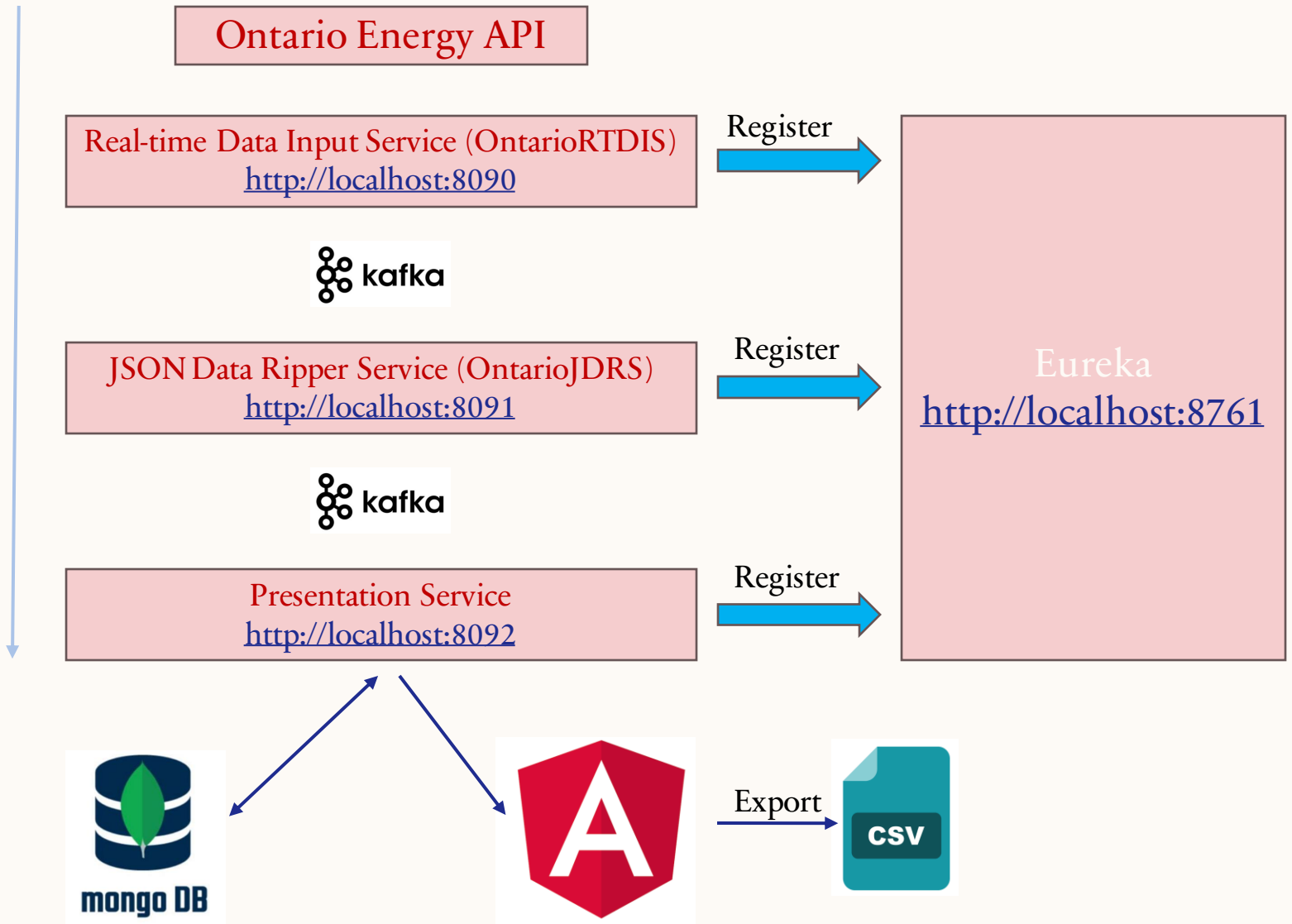
ARCHITECTURE



MICROSERVICE

EUREKA

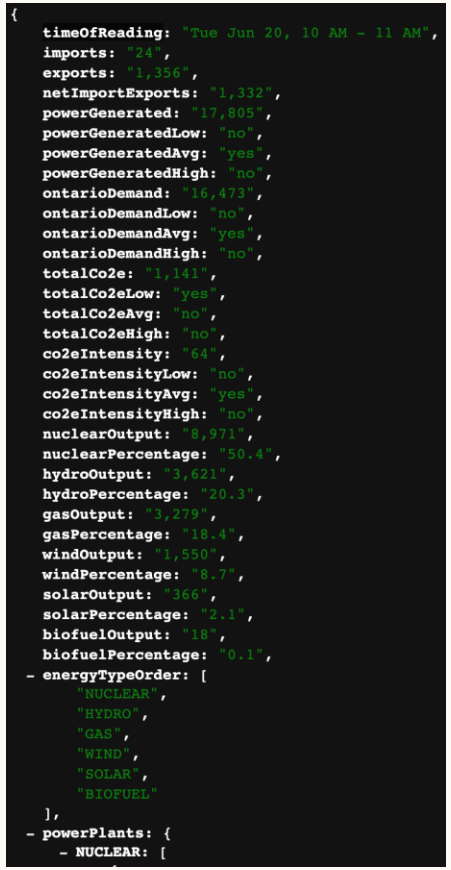

```yaml
spring:
  application:
    name: EurekaServer

server:
  port: 8761

eureka:
  instance:
    hostname: localhost
  client:
    registerWithEureka: false #telling the server not to register himself
    fetchRegistry: false
```

```
server.port=8091
spring.application.name=OntarioJDRS
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
```

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ONTARIOJDRS	n/a (1)	(1)	UP (1) - CS590-202306-28.cs.mum.edu:OntarioJDRS:8091
ONTARIORTDI	n/a (1)	(1)	UP (1) - CS590-202306-28.cs.mum.edu:OntarioRTDI:8090
PRESENTATIONSERVICE	n/a (1)	(1)	UP (1) - CS590-202306-28.cs.mum.edu:PresentationService:8092

REAL-TIME DATA INPUT SERVICE




```java
@Autowired
private KafkaProducer kafkaProducer;

no usages  Tri An Vuong
@Scheduled(cron = "@hourly")
public void getData() {
    RestTemplate restTemplate = new RestTemplate();
    String url = "https://live.gridwatch.ca/WebServices/GridWatchWebApp.asmx/GetHomeViewData_v2";
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    HttpEntity<String> entity = new HttpEntity<~>(headers);
    String answer = restTemplate.getForObject(url, String.class, entity);
    System.out.println(answer);
    //published data to stream using KafkaProducer
    kafkaProducer.sendMessage(answer);
}
```

"stream-input"

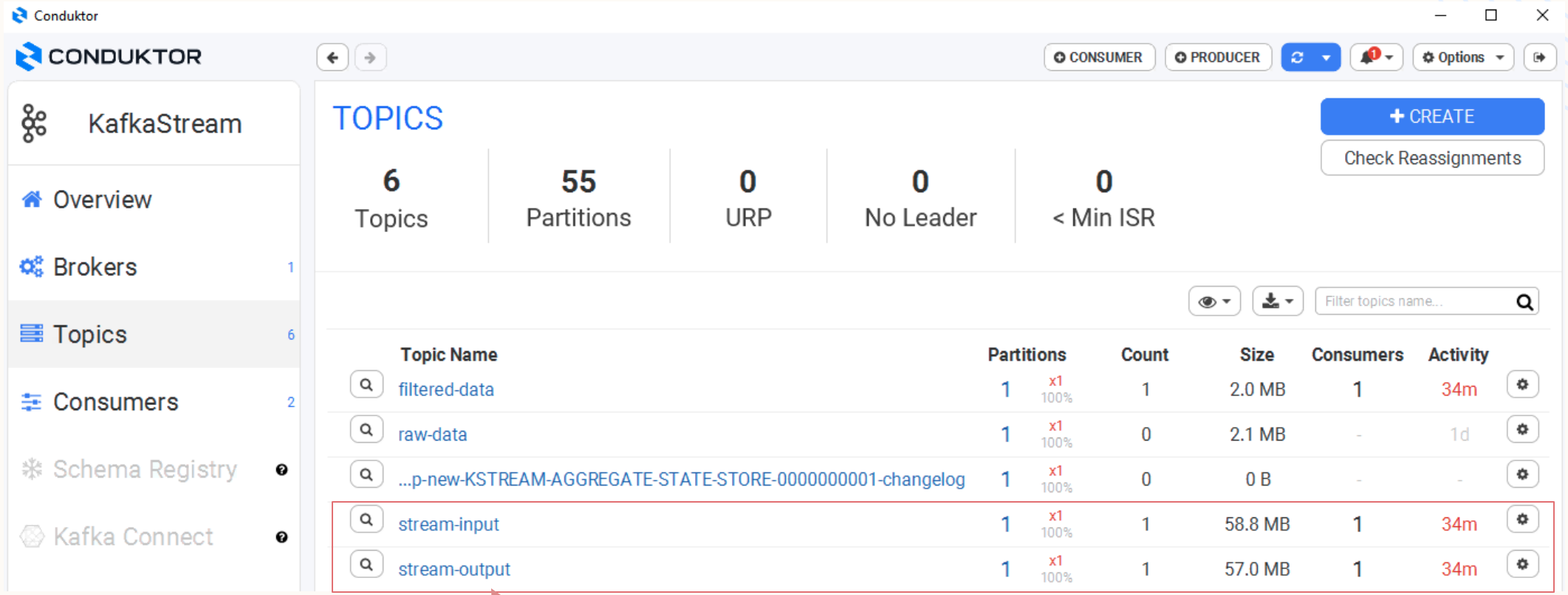
```java
@Autowired
void buildPipeline(StreamsBuilder streamsBuilder) {
    KStream<String, String> messageStream = streamsBuilder
            .stream( topic: "stream-input", Consumed.with(STRING_SERDE, STRING_SERDE));
    messageStream.to( s: "stream-output");
}
```

kafka

"stream-output"

Consumer Service

REAL-TIME DATA INPUT SERVICE



Kafka receive stream-input & stream-output topic

JSON DATA RIPPER SERVICE

kafka

```java
@KafkaListener(topics = "stream-output")
public void listenGroupFoo(String message) {
    System.out.println("Received Message from Kafka stream: " + message);
    OntarioEnergy ontarioWeather = ConvertStringToObject.covertFromJsonToOntario(message);
    String ontarioWeatherString = ConvertObjectToString.convertFromOntarioWeatherToString(ontarioWeather);
    System.out.println("Sending convert object: " + ontarioWeatherString);
    sender.send( topic: "filtered-data", ontarioWeatherString);
}
```

kafka

Consumer Service

JSON DATA RIPPER SERVICE

Filter data

```java
@KafkaListener(topics = "stream-output")
public void listenGroupFoo(String message) {
    System.out.println("Received Message from Kafka stream: " + message);
    OntarioEnergy ontarioWeather = ConvertStringToObject.covertFromJsonToOntario(message);
    String ontarioWeatherString = ConvertObjectToString.convertFromOntarioWeatherToString(ontarioWeather);
    System.out.println("Sending convert object: " + ontarioWeatherString);
    sender.send( topic: "filtered-data", ontarioWeatherString);
}
```


```
{
   timeOfReading: "Tue Jun 20, 10 AM - 11 AM",
   imports: "24",
   exports: "1,356",
   netImportExports: "1,332",
   powerGenerated: "17,805",
   powerGeneratedLow: "no",
   powerGeneratedAvg: "yes",
   powerGeneratedHigh: "no",
   ontarioDemand: "16,473",
   ontarioDemandLow: "no",
   ontarioDemandAvg: "yes",
   ontarioDemandHigh: "no",
   totalCo2e: "1,141",
   totalCo2eLow: "yes",
   totalCo2eAvg: "no",
   totalCo2eHigh: "no",
   co2eIntensity: "64",
   co2eIntensityLow: "no",
   co2eIntensityAvg: "yes",
   co2eIntensityHigh: "no",
   nuclearOutput: "8,971",
   nuclearPercentage: "50.4",
   hydroOutput: "3,621",
   hydroPercentage: "20.3",
   gasOutput: "3,279",
   gasPercentage: "18.4",
   windOutput: "1,550",
   windPercentage: "8.7",
   solarOutput: "366",
   solarPercentage: "2.1",
   biofuelOutput: "18",
   biofuelPercentage: "0.1",
 - energyTypeOrder: [
       "NUCLEAR",
       "HYDRO",
       "GAS",
       "WIND",
       "SOLAR",
       "BIOFUEL"
   ],
 - powerPlants: {
     - NUCLEAR: [
```

```
{
  "date": "2023-06-21T05:18:28.362+00:00",
  "timeOfReading": "Wed Jun 21, 11 PM - 12 AM",
  "powerGenerated": "17,032",
  "ontarioDemand": "15,817",
  "totalCo2e": "926",
  "co2eIntensity": "54",
  "nuclearPercentage": "52.8",
  "nuclearOutput": "8,996",
  "hydroPercentage": "21.2",
  "hydroOutput": "3,611",
  "gasPercentage": "16.3",
  "gasOutput": "2,782",
  "windPercentage": "9.5",
  "windOutput": "1,626",
  "biofuelPercentage": "0.1",
  "biofuelOutput": "17",
  "solarPercentage": "0.0",
  "solarOutput": "0",
  "imports": "9",
  "exports": "1,224",
  "netImportExports": "1,215"
},
```

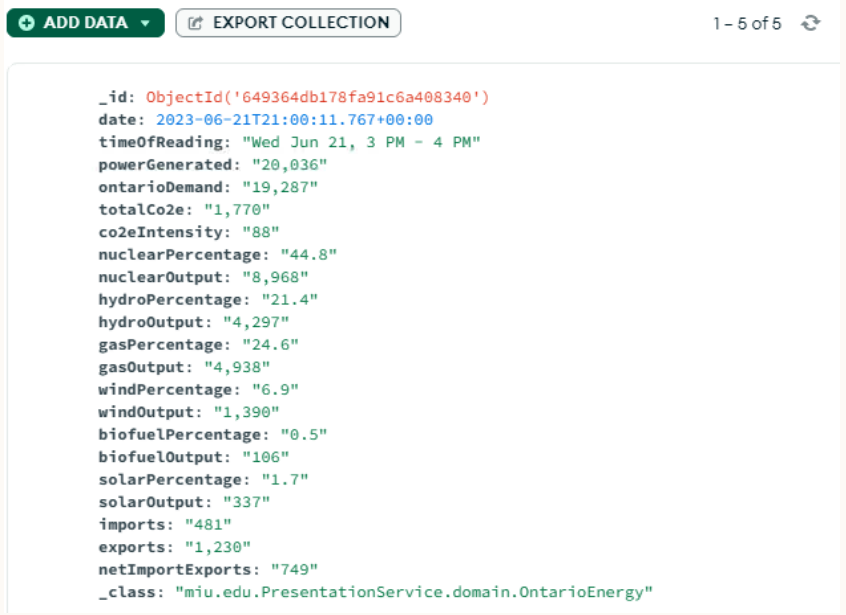
PRESENTATION SERVICE

kafka →

```java
@KafkaListener(topics = "filtered-data")
public void listenGroupFoo(String message) {
    System.out.println("Received Message from ripper service: " + message);
    OntarioEnergyDTO ontarioWeatherDTO = ConvertStringToObject.covertFromJsonToOntario(message);
    OntarioEnergy ontarioWeather = OntarioEnergyAdapter.convertFromDtoToOntarioWeather(ontarioWeatherDTO);
    ontarioWeather.setDate(new Date());
    repository.save(ontarioWeather);
}
```

↓

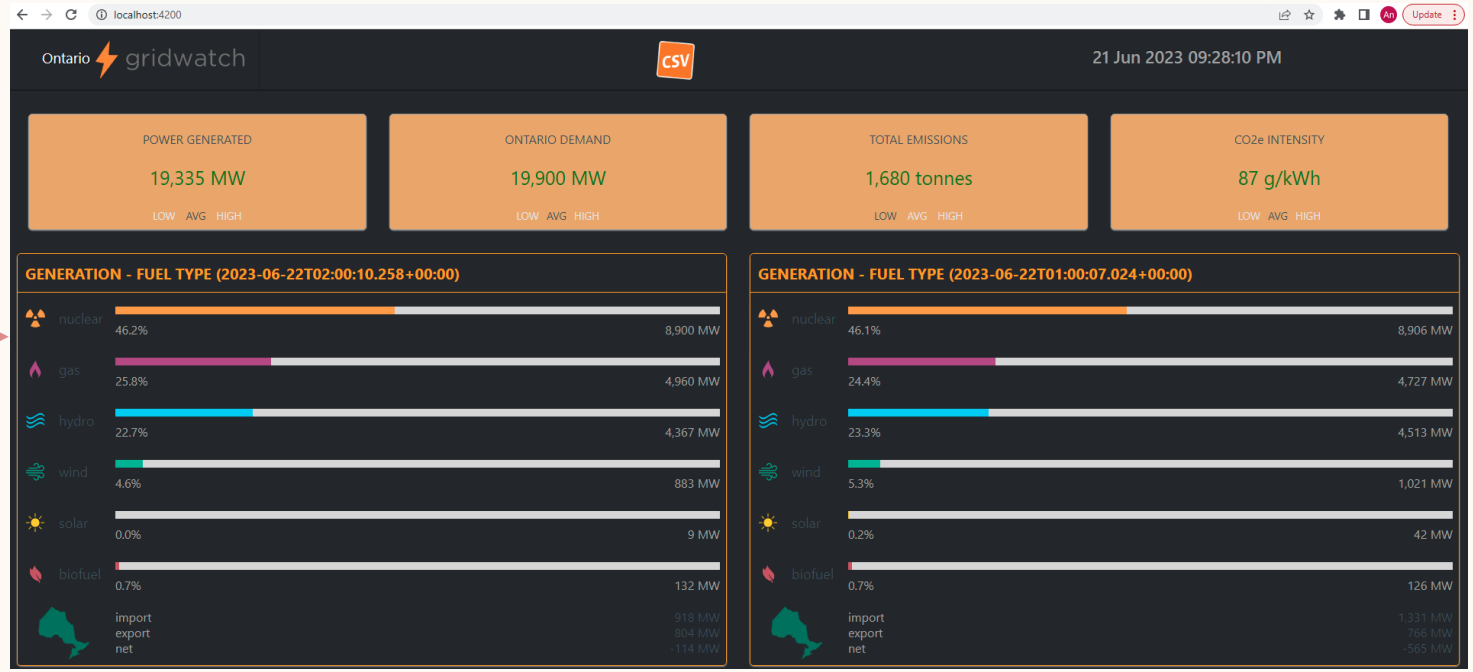
mongo DB



PRESENTATION SERVICE


```java
@RestController
@RequestMapping("api/ontario-energy")
public class OntarioEnergyController {

    2 usages
    @Autowired
    private IOntarioEnergyService service;

    no usages  Tri An Vuong
    @GetMapping("")
    public ResponseEntity<?> findAll () {
        Data data = new Data(service.findTop2());
        return new ResponseEntity<Data>(data, HttpStatus.OK);
    }
```

Controller to provide data for UI



PRESENTATION SERVICE


```java
@GetMapping("/export")
public void exportToCSV (HttpServletResponse response) throws IOException {
    response.setContentType("text/csv");
    String fileName = "ontario-energy.csv";
    String headerKey = "Content-Disposition";
    String headerValue = "attachment; filename=" + fileName;
    response.setHeader(headerKey, headerValue);

    Data data = new Data(service.findAll());
    ICsvBeanWriter csvWriter = new CsvBeanWriter(response.getWriter(), CsvPreference.STANDARD_PREFERENCE);

    String[] csvHeader = {"Date & Time", "Time Of Reading", "Power Generated",
            "Ontario Demand", "Total Co2e", "Co2e Intensity", "Nuclear Percentage",
            "Nuclear Output", "Hydro Percentage", "Hydro Output", "Gas Percentage",
            "Gas Output", "Wind Percentage", "Wind Output", "Biofuel Percentage",
            "Biofuel Output", "Solar Percentage", "Solar Output", "Imports", "Exports", "Net Import Exports"};
    String[] nameMapping = {"date", "timeOfReading", "powerGenerated",
            "ontarioDemand", "totalCo2e", "co2eIntensity",
            "nuclearPercentage", "nuclearOutput", "hydroPercentage",
            "hydroOutput", "gasPercentage", "gasOutput",
            "windPercentage", "windOutput", "biofuelPercentage",
            "biofuelOutput", "solarPercentage", "solarOutput",
            "imports", "exports", "netImportExports"};

    csvWriter.writeHeader(csvHeader);
    for (OntarioEnergy d: data.getData()) {
        csvWriter.write(d, nameMapping);
    }
    csvWriter.close();
}
```

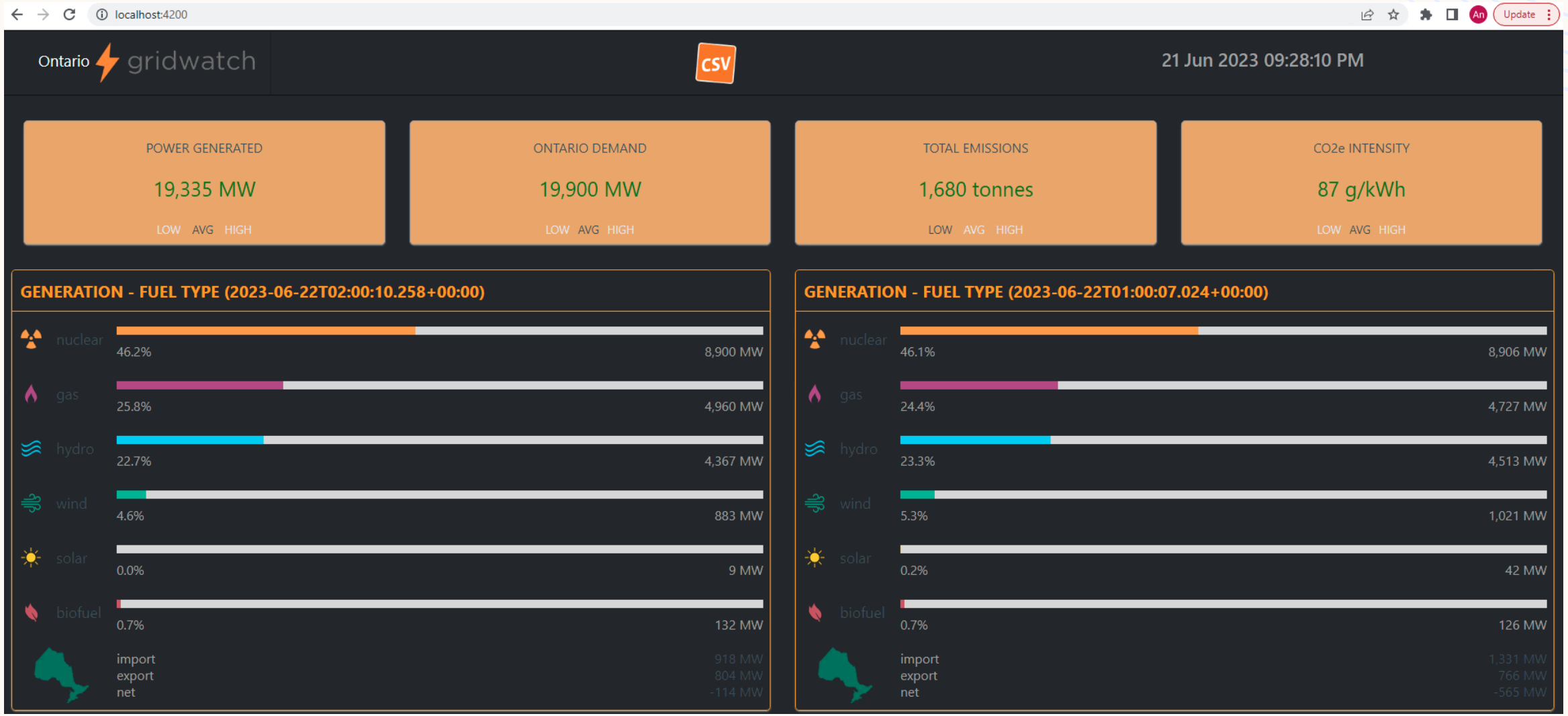
Controller to export CSV file

ontario-energy (7) - Notepad

File Edit Format View Help

```
Date & Time,Time Of Reading,Power Generated,Ontario Demand,Total Co2e,Co2e Intensity,Nuclear Percentage,Nuclear Output,Hydro Percentage,Hydro Output,Gas Percentage,Gas Outpu
Wed Jun 21 16:00:11 CDT 2023,"Wed Jun 21, 3 PM - 4 PM","20,036","19,287","1,770",88,44.8,"8,968",21.4,"4,297",24.6,"4,938",6.9,"1,390",0.5,106,1.7,337,481,"1,230",749
Wed Jun 21 17:00:06 CDT 2023,"Wed Jun 21, 4 PM - 5 PM","20,278","19,752","1,792",88,44.1,"8,943",22.4,"4,549",24.6,"4,979",6.8,"1,371",0.7,138,1.5,298,675,"1,201",526
Wed Jun 21 18:00:07 CDT 2023,"Wed Jun 21, 5 PM - 6 PM","20,139","20,198","1,775",88,44.2,"8,910",23.6,"4,758",24.6,"4,951",5.7,"1,152",0.8,165,1.0,203,987,928,-59
Wed Jun 21 19:00:11 CDT 2023,"Wed Jun 21, 6 PM - 7 PM","19,759","20,045","1,697",86,45.1,"8,907",24.4,"4,828",24.1,"4,756",5.1,"1,016",0.7,132,0.6,120,"1,352","1,066",-286
Wed Jun 21 20:00:07 CDT 2023,"Wed Jun 21, 7 PM - 8 PM","19,335","19,900","1,680",87,46.1,"8,906",23.3,"4,513",24.4,"4,727",5.3,"1,021",0.7,126,0.2,42,"1,331",766,-565
Wed Jun 21 21:00:10 CDT 2023,"Wed Jun 21, 8 PM - 9 PM","19,251","19,365","1,780",92,46.2,"8,900",22.7,"4,367",25.8,"4,960",4.6,883,0.7,132,0.0,9,918,804,-114
```

FRONT END


localhost:4200
Update
Ontario gridwatch
CSV
21 Jun 2023 09:28:10 PM
POWER GENERATED
19,335 MW
LOW AVG HIGH
ONTARIO DEMAND
19,900 MW
LOW AVG HIGH
TOTAL EMISSIONS
1,680 tonnes
LOW AVG HIGH
CO2e INTENSITY
87 g/kWh
LOW AVG HIGH
GENERATION - FUEL TYPE (2023-06-22T02:00:10.258+00:00)
nuclear 46.2% 8,900 MW
gas 25.8% 4,960 MW
hydro 22.7% 4,367 MW
wind 4.6% 883 MW
solar 0.0% 9 MW
biofuel 0.7% 132 MW
import 918 MW
export 804 MW
net -114 MW
GENERATION - FUEL TYPE (2023-06-22T01:00:07.024+00:00)
nuclear 46.1% 8,906 MW
gas 24.4% 4,727 MW
hydro 23.3% 4,513 MW
wind 5.3% 1,021 MW
solar 0.2% 42 MW
biofuel 0.7% 126 MW
import 1,331 MW
export 766 MW
net -565 MW

SUMMARY

The Ontario Energy Application is a cutting-edge digital tool designed to revolutionize energy management practices in the province of Ontario, Canada. This presentation provides a comprehensive summary of the application's key features, benefits, and its role in promoting sustainable energy usage. With its user-friendly interface and advanced analytics capabilities, the Ontario Energy Management Application aims to empower individuals, businesses, and policymakers to make informed decisions and optimize energy consumption.

THANK YOU