SMART ENERGY SYSTEM

Faculty Of Science And Engineering

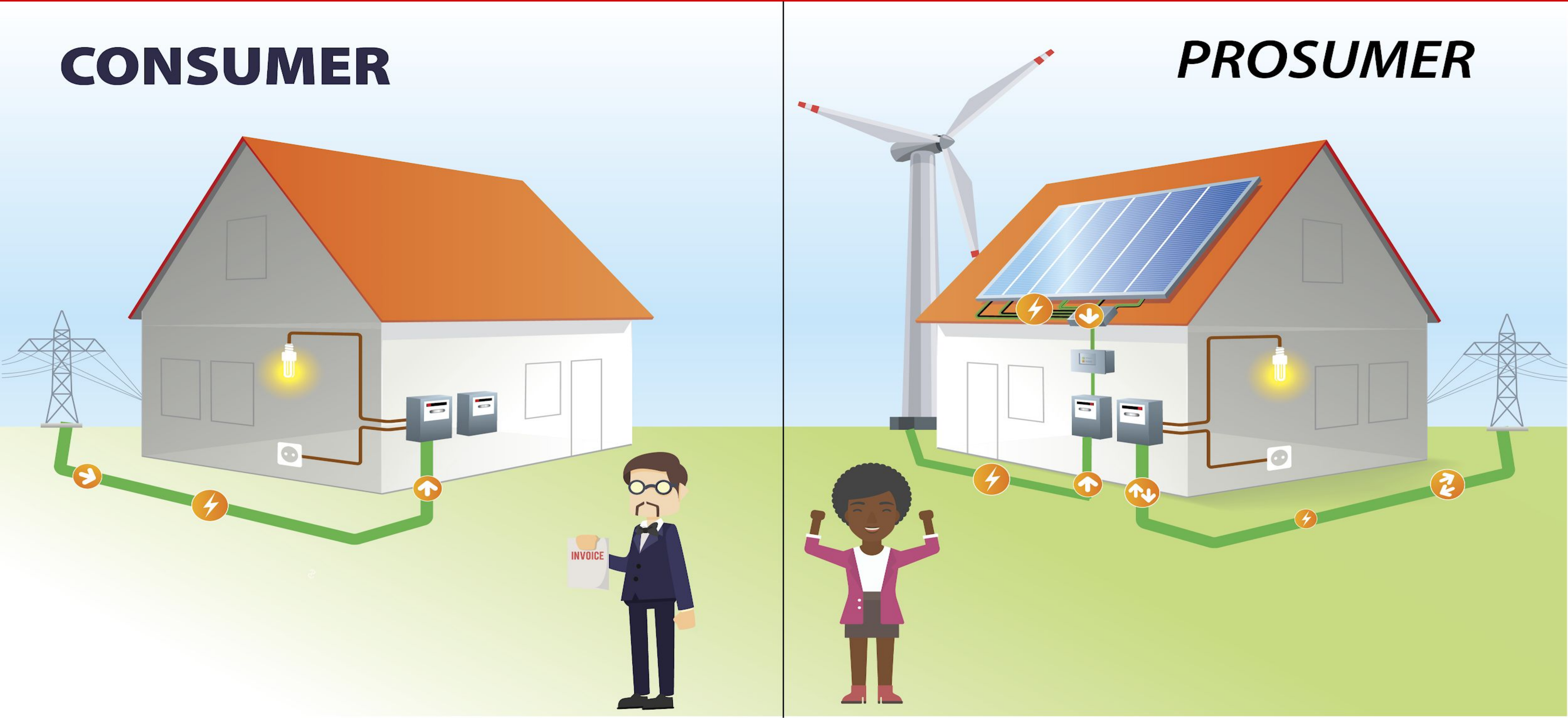
Web and Cloud Computing (INMWCC)

Group 30

Swastik S Nayak

Siddharth B

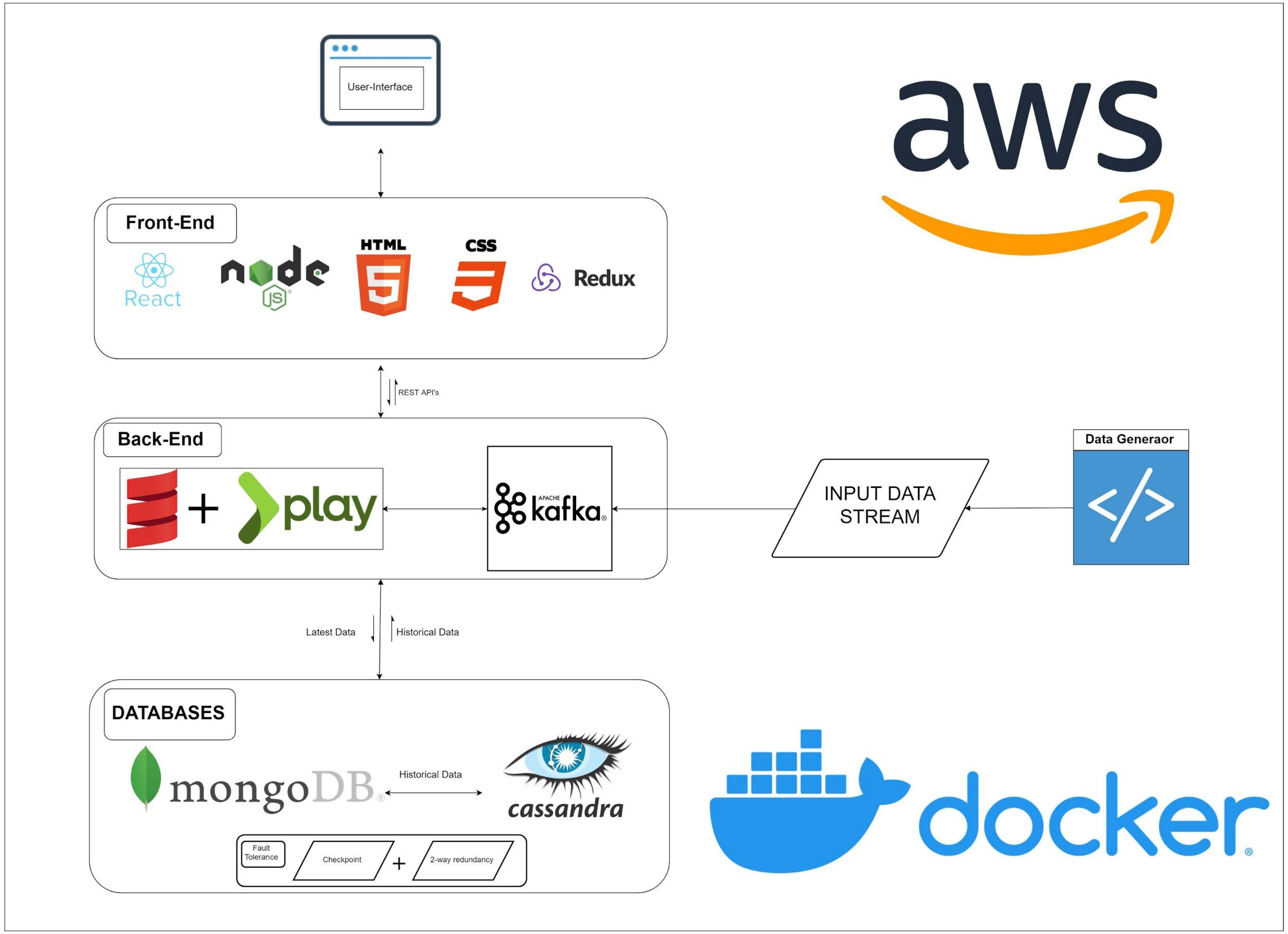
Anil P.Mathew


CONSUMER
PROSUMER
INVOICE

Abstract

- The transition towards more sustainable energy transforms the passive consumers of energy into active prosumers that both consume and produce energy.
- This sparked the interest to enable a dynamic energy system where the prosumers can sell energy based on their current consumption and availability.
- The system allows the prosumers to trade their energy amongst their peer which promotes awareness to conserve energy, all the while promoting a clean and sustainable energy source.

Architecture



User-Interface

Front-End

React

node JS

HTML 5

CSS 3

Redux

REST API's

Back-End

+ play

APACHE kafka

INPUT DATA STREAM

Data Generaor

Latest Data

Historical Data

DATABASES

mongoDB

Historical Data

cassandra

Fault Tolerance

Checkpoint + 2-way redundancy

aws

docker

Functional Requirements

- Dashboard depicting the current energy consumption.
- A feature to trade the energy and limit the trade as per the prosumers requirements.
- Prosumer should be able to add/edit/remove an appliance from his registered list of appliances.
- Prosumer should be able to set a threshold of energy to trade.
- A monthly breakdown of the energy consumption and energy trade for the prosumer’s viewing.
- Admin Dashboard.

Technologies used

TECHNOLOGY	KEY FEATURES	USAGE
Scala	Scalable or Multi-paradigm (FP+OOP), concise, strongly typed, immutable-first language, non existent boiler plate codes, interoperability with java, easy to maintain, leverage the advantages of JVM.	The backend code will consist mainly of scala to enable a scalable, concise, clean and maintainable code. The language will allow for a faster turnaround time to reduce the coding efforts. Will contain the business logic or the Engine which drives the web application.
Play Framework-2	Productive environment, reactive, Typesafe, Amazing error handling, Flexible, Modern stack, supports Java and Scala.	Will predominantly deal with the RESTful APIs.
Kafka	High Throughput, lower latency, fault tolerant and durability.	Deals with streaming of data across the system.
MongoDB	Easy-to-scale-out, handles heterogenous data, simple query syntaxes, Document oriented storage (json structure)	Will be the primary database backing up the system.
Cassandra	Handles massive amount of unstructured data, high availability and fault tolerant, elastic scalability, high performance.	Secondary database to create a fault tolerant environment.
AWS	Easy to use, secure and reliable, agile, scalable environment.	Cloud platform to support the web application
Maven/SBT	Build tool.	To handle continuous integration, profiling, and dependencies across the project.

rijksuniversiteit
groningen

Technologies used

TECHNOLOGY	KEY FEATURES	USAGE
ReactJs	Maintainable, faster rendering, SEO friendly, variety of developer tool sets.	Will focus mainly on the web interfaces.
NodeJs	Single Page Apps, real time solutions, REST, data streaming	Will provide websockets.
Redux	Predictable states, Maintainability, Ease of testing.	Maintains the state of the global app.
Docker	Compatibility and maintainability, simplicity and faster configuration, rapid development, deployment and testing.	Used to simulate the environment and create docker images for continuous deployment, testing and releases.

Thank You