REACT AUTOMATION STUDIO:

A NEW FACE TO CONTROL LARGE SCIENTIFIC EQUIPMENT

William Duckitt

Outline

Introduction to React Automation Studio

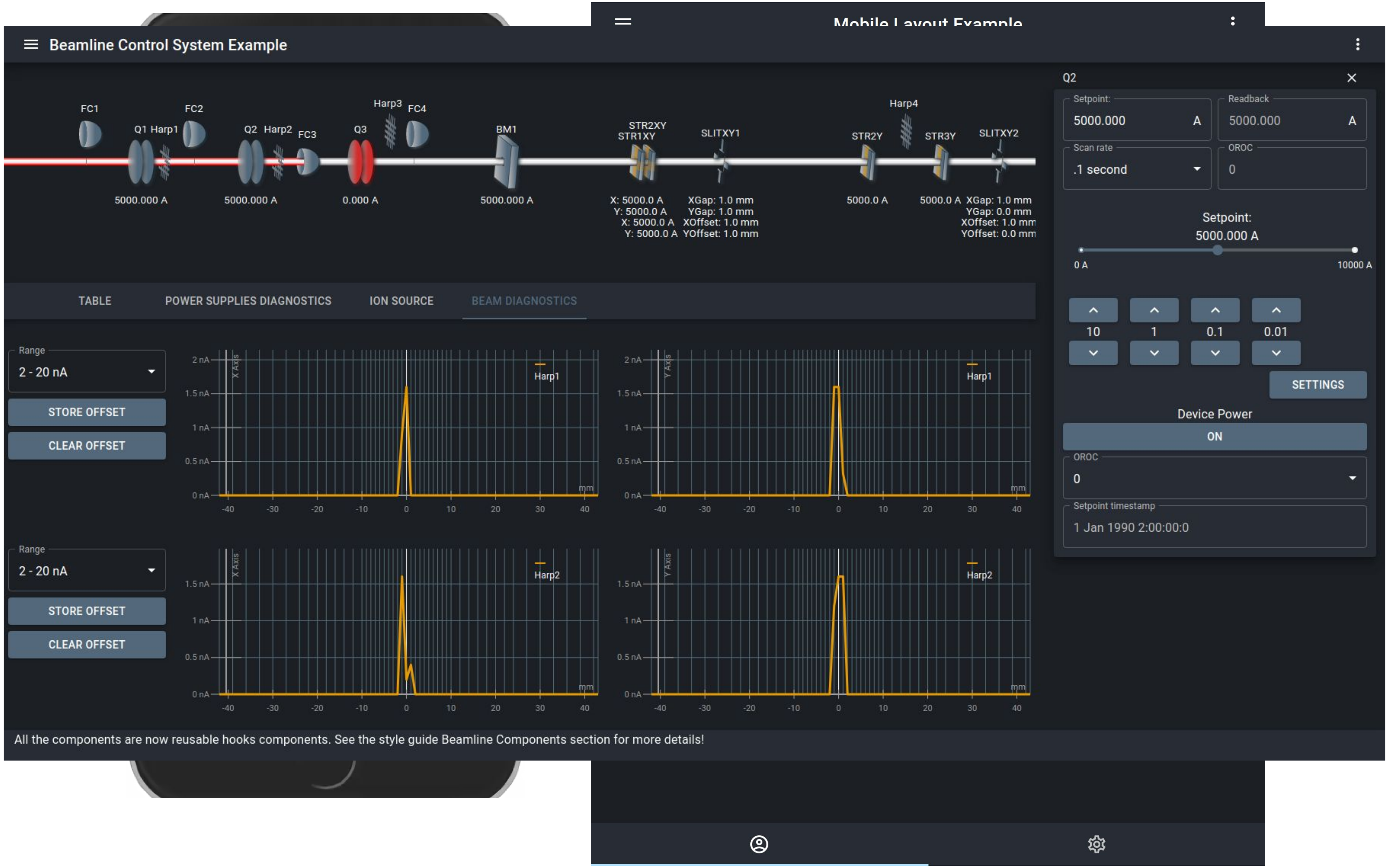
What, why and how we did it

Software Stack and Protocols

Current Features

Live Demos

React Automation Studio



Range: 2 - 20 nA

Range: 2 - 20 nA

Setpoint: 5000.000 A

Readback: 5000.000 A

Scan rate: .1 second

OROC: 0

OROC: 0

Setpoint timestamp: 1 Jan 1990 2:00:00:0

All the components are now reusable hooks components. See the style guide Beamline Components section for more details!

Progressive web application (PWA) framework to create cross platform and cross device UIs for EPICS

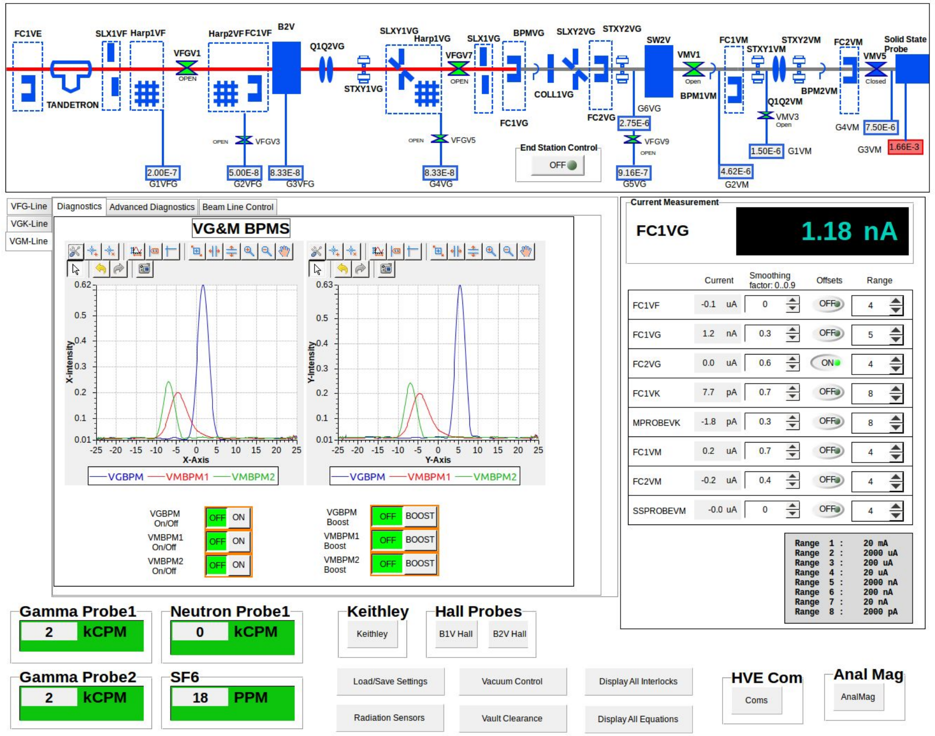
Runs on your desktop

On your mobile

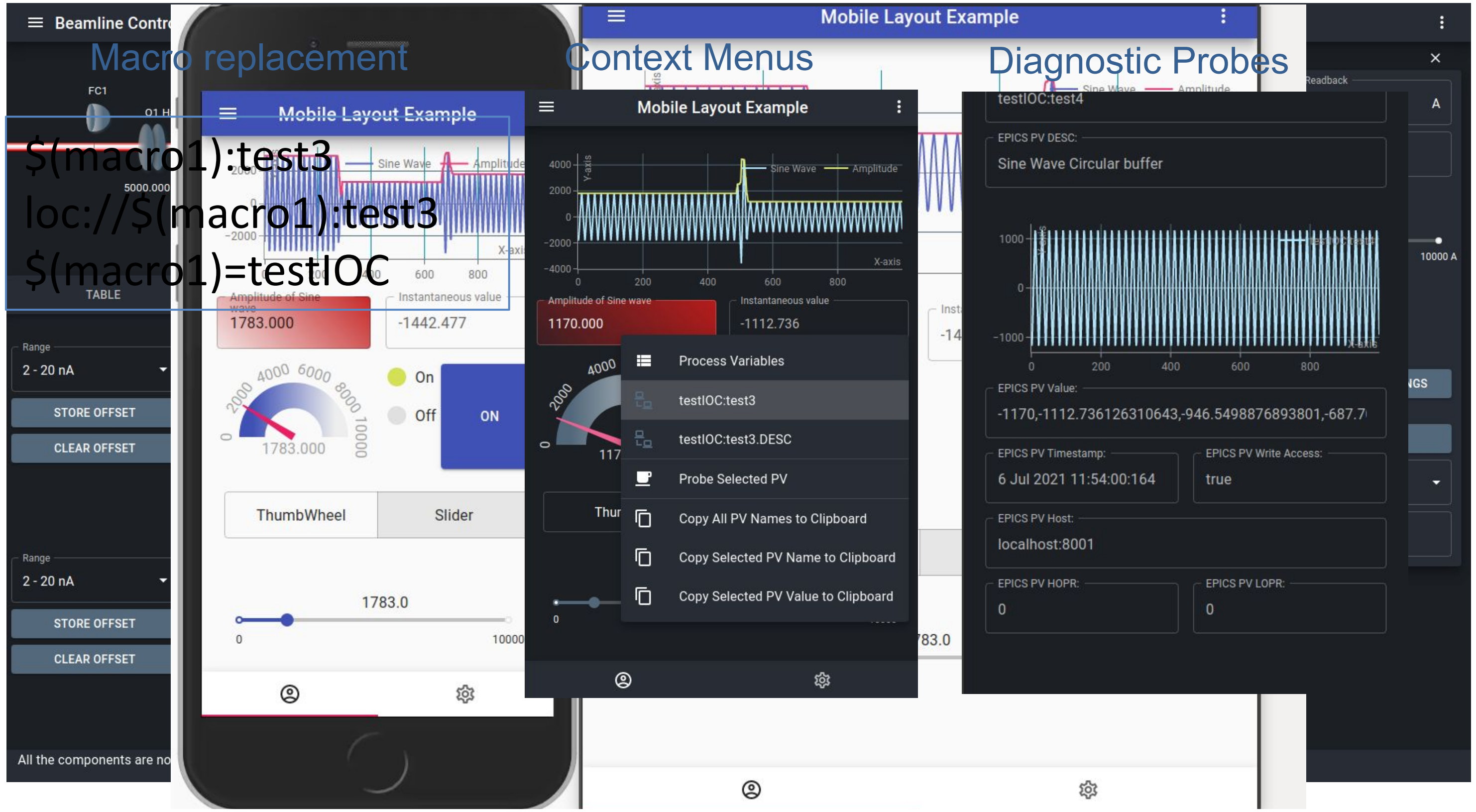
And in the web browser

Evolutionary leap in control system UI design

CS-Studio



React Automation Studio


Macro replacement

$(macro1):test3
loc://$(macro1):test3
$(macro1)=testIOC

Context Menus

Diagnostic Probes

EPICS PV DESC: Sine Wave Circular buffer

EPICS PV Value: -1170,-1112.736126310643,-946.5498876893801,-687.7

EPICS PV Timestamp: 6 Jul 2021 11:54:00:164

EPICS PV Write Access: true

EPICS PV Host: localhost:8001

EPICS PV HOPR: 0

EPICS PV LOPR: 0

We used all our experience and knowledge in creating unified EPICS-EtherCAT control systems with centralized and hierarchically structured CS-Studio UI’s

Built a system that is leveraged on top the R&D of the worlds biggest corporations and their open-sourced software frameworks

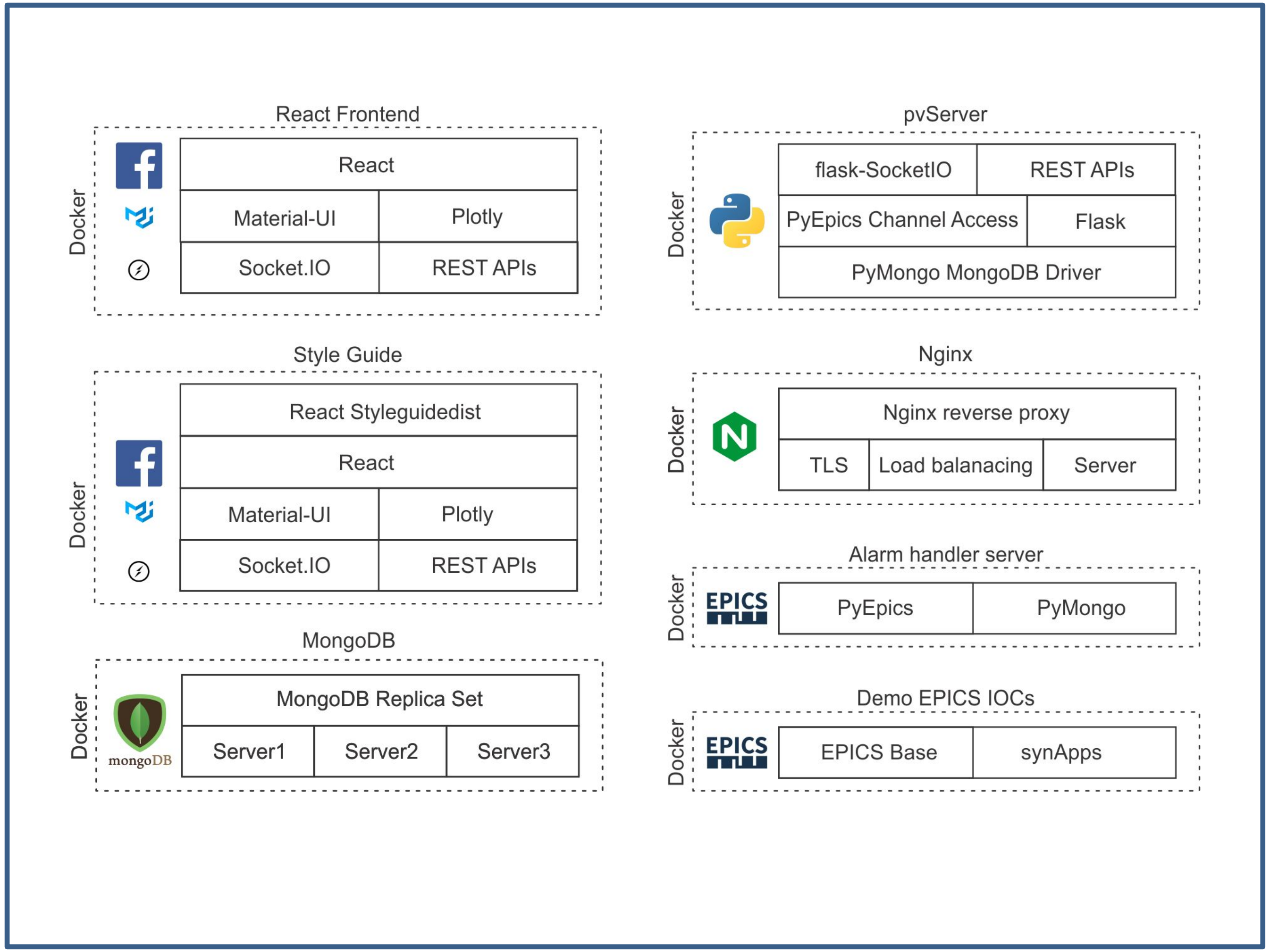
Cross-platform and cross-device

Themeable

Implements the best features of CS-studio, without its limitations.

Software Stack

Git Mono Repo



React Frontend

Docker

React

Material-UI | Plotly

Socket.IO | REST APIs

Style Guide

Docker

React Styleguidedist

React

Material-UI | Plotly

Socket.IO | REST APIs

MongoDB

Docker

MongoDB Replica Set

Server1 | Server2 | Server3

pvServer

Docker

flask-SocketIO | REST APIs

PyEpics Channel Access | Flask

PyMongo MongoDB Driver

Nginx

Docker

Nginx reverse proxy

TLS | Load balanacing | Server

Alarm handler server

Docker

PyEpics | PyMongo

Demo EPICS IOCs

Docker

EPICS Base | synApps

Development and production versions have been containerized with Docker, deployed with one command as micro-services

Code entry is in React JavaScript, don’t need knowledge of HTML or CSS

Wrappers on Material-UI components and Plotly graphs

PyEpics based backend

Socket-IO communication between components and pvServer

Containerized Demo IOC

Web based access rights administration

Fully interactive help and style guide

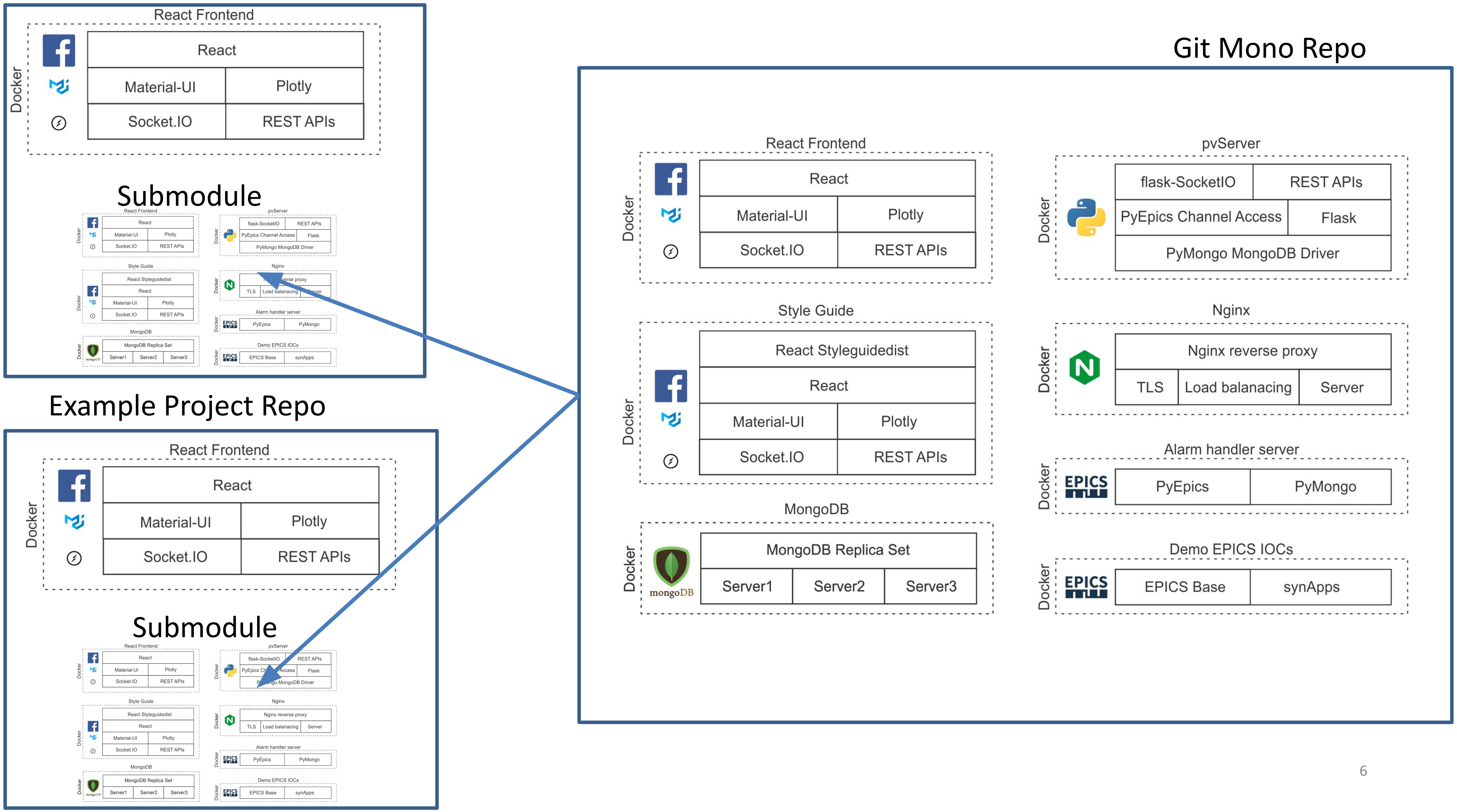
Software Stack

iThemba LABS Repo

Non-Public

Example Project Repo

Public
Customizable



Git Mono Repo

React Frontend
Docker
React
Material-UI | Plotly
Socket.IO | REST APIs

pvServer
Docker
flask-SocketIO | REST APIs
PyEpics Channel Access | Flask
PyMongo MongoDB Driver

Style Guide
Docker
React Styleguidedist
React
Material-UI | Plotly
Socket.IO | REST APIs

Nginx
Docker
Nginx reverse proxy
TLS | Load balanacing | Server

Alarm handler server
Docker
PyEpics | PyMongo

MongoDB
Docker
MongoDB Replica Set
Server1 | Server2 | Server3

Demo EPICS IOCs
Docker
EPICS Base | synApps

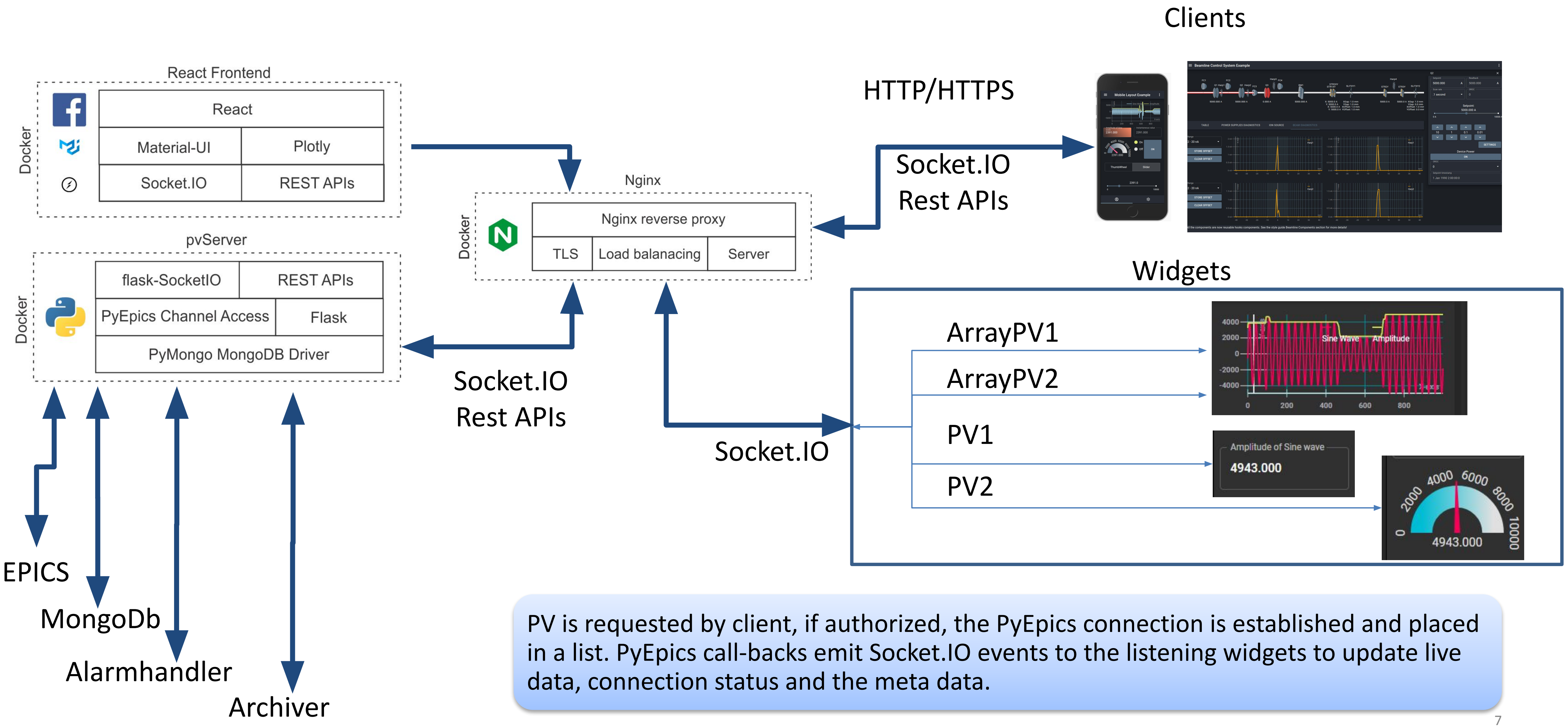
iThemba LABS Repo — React Frontend
Docker
React
Material-UI | Plotly
Socket.IO | REST APIs

Submodule

Example Project Repo — React Frontend
Docker
React
Material-UI | Plotly
Socket.IO | REST APIs

Submodule

Software Protocol



Clients

HTTP/HTTPS

Socket.IO
Rest APIs

React Frontend

Docker

React

Material-UI

Plotly

Socket.IO

REST APIs

Nginx

Docker

Nginx reverse proxy

TLS

Load balanacing

Server

pvServer

Docker

flask-SocketIO

REST APIs

PyEpics Channel Access

Flask

PyMongo MongoDB Driver

Socket.IO
Rest APIs

Socket.IO

Widgets

ArrayPV1

ArrayPV2

PV1

PV2

EPICS

MongoDb

Alarmhandler

Archiver

PV is requested by client, if authorized, the PyEpics connection is established and placed in a list. PyEpics call-backs emit Socket.IO events to the listening widgets to update live data, connection status and the meta data.

Major Milestones

1. Project initiated in January 2019
2. 1st Public release in October 2019
   a. Fully containerized
   b. Packaged with all major control system widgets
   c. File based user access control
3. 2nd Major release in August 2020
   a. Database integration
   b. Moved to React -Hooks based components, with centralised widget component.
   c. Initial release of the Alarm Handler & load save appliances
4. 3rd Major release in June 2021.
   a. Production ready Alarm Handler, with email, Whatsapp and Signal notifications,
   b. Nginx based load balancing, static file serving
   c. Web-based administration and access control.
   d. Production ready Archiver appliance
   e. External authentication with Active Directory and Google.
   f. Improvement in security with the move from access tokens to memory based short lived access tokens and long lived refresh tokens in secure cookie storage.

V3.0.1 28 June 2021 Features

- 36 Production Ready Widgets
- Archived Data viewer
- Load/Save UI with life-cycle management
- Fully fledged Alarm Handler server and client, supporting, email, SMS, WhatsApp and Signal message notifications
- Built in demos
- Fully interactive Help and Style Guide
- Multi theme support
- Secure, client pages are served over HTTPS
- Load balancing with Nginx
- Web-based administration and access control
- Access control can restrict or grant read and write access to PV
- Protected Routes restrict pages to privileged user access groups
- External Authentication via Active Directory or Google

Prerequisites

Linux ,currently supporting Ubuntu 18.04, in theory any Linux environment with docker will work

Experimental Physics and Industrial Control System

React JS Getting Started: https://reactjs.org/docs/getting-started.html
Hooks: https://reactjs.org/docs/hooks-intro.html

Docker Getting started: https://www.docker.com/play-with-docker

Getting Started: Clone the Example Project

In your home folder clone with recurse submodules:

```
git clone --recurse-submodules https://github.com/wduckitt/React-Automation-Studio-Example-Project-1.git
```

cd into: ~/React-Automation-Studio-Example-Project-1

```
git checkout tags/V3.0.1
```

```
touch .env
```

Then run

```
docker-compose -f docker-compose-prod-with-demoioc.yml up
```

To compile the production version with the demoIOC's and style guide

This will take 20 minutes or so the first time.

When all microservices are up go to: http://localhost:5000/ to view web app.

Follow the instructions on how to launch in dev mode.

Github Repos

Master Repo:

https://github.com/wduckitt/React-Automation-Studio

Example Project:

https://github.com/wduckitt/React-Automation-Studio-Example-Project-1

Standalone Alarm Handler
https://github.com/wduckitt/React-Automation-Studio-Alarm-Handler-Standalone

Contributions and bug fixes via pull requests

Contact: rasadmin@tlabs.ac.za

Live Demos

If you missed the live Demos session go to: https://www.youtube.com/channel/UCKX2dcKoMXsN_10zjB4dlPg for screen recordings.

Conclusion

Successfully designed PWA software platform to enable the control of large scientific equipment through EPICS

The system is cross device and cross platform compatible

We encourage the community to test, evaluate and contribute to React Automation Studio

Thank you