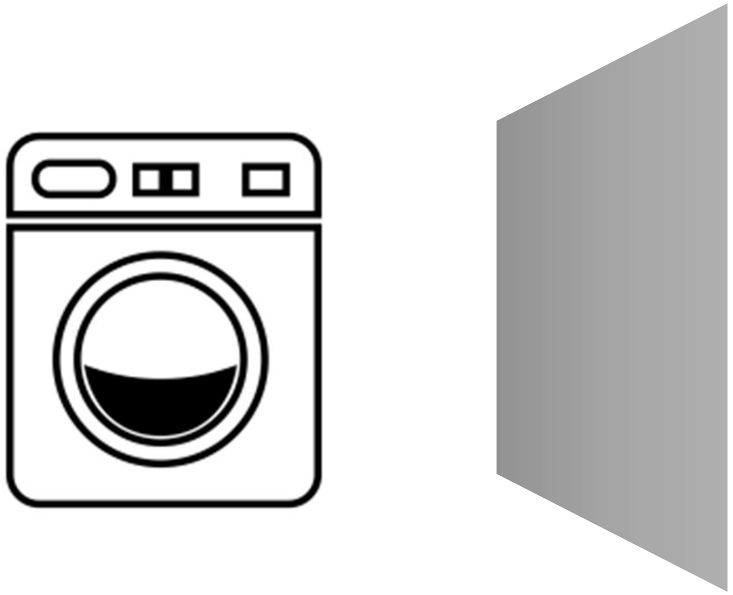
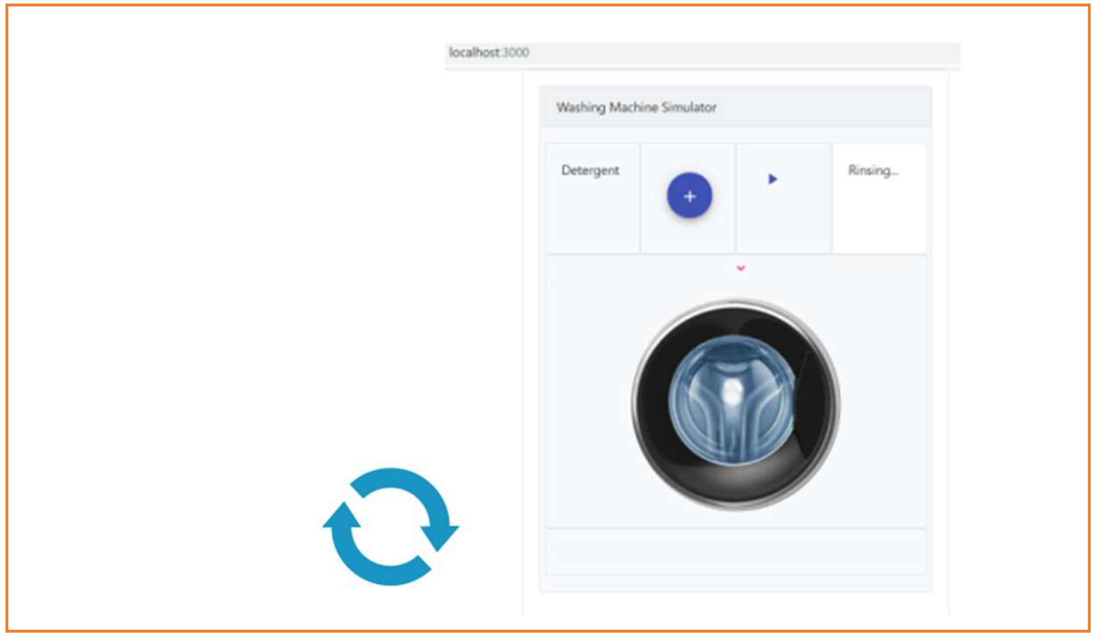
Washing Machine| Web Simulator

Presentation



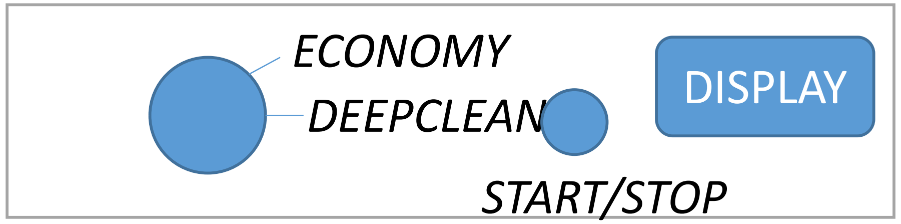


Agenda

Overview| Architecture| Getting Started -API | ReactJS | Summary

Source: All Icons courtesy: https://thenounproject.com/

Overview| Use case/Context



Steps

1. Set Mode in Dial
2. Fill Detergent
3. Start

First set of Features

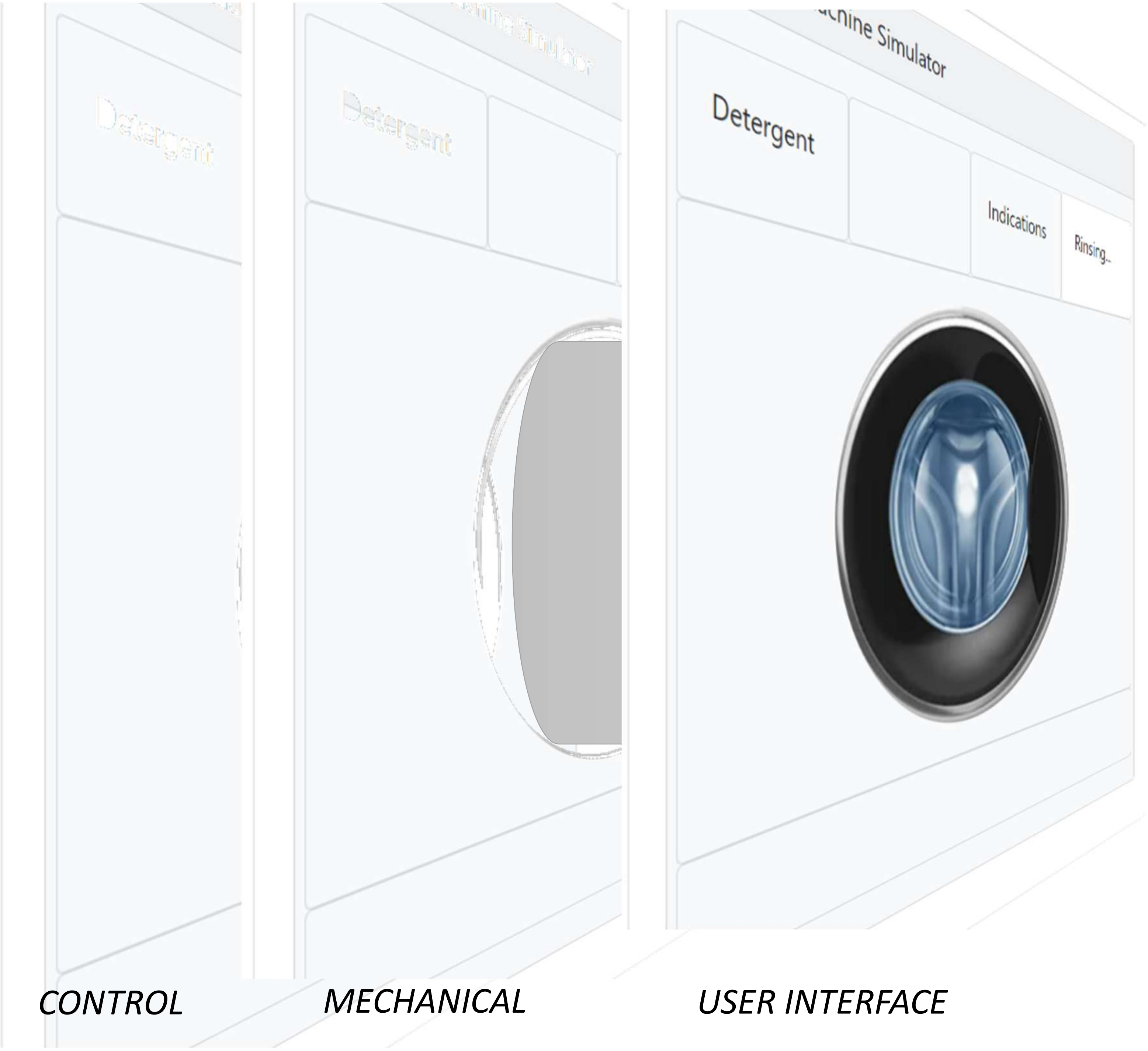
1. *Economical Wash Mode*
2. *Sensitive Wash Mode*

Assumptions:

These wash modes all have common Sequence of Operations

1. *Collecting Water*
2. *Ready for Wash*
3. *Rinse and Repeat*
4. *Drain Water*
5. *Repeat Rinsing*
6. *Drain Water*
7. *Dry*

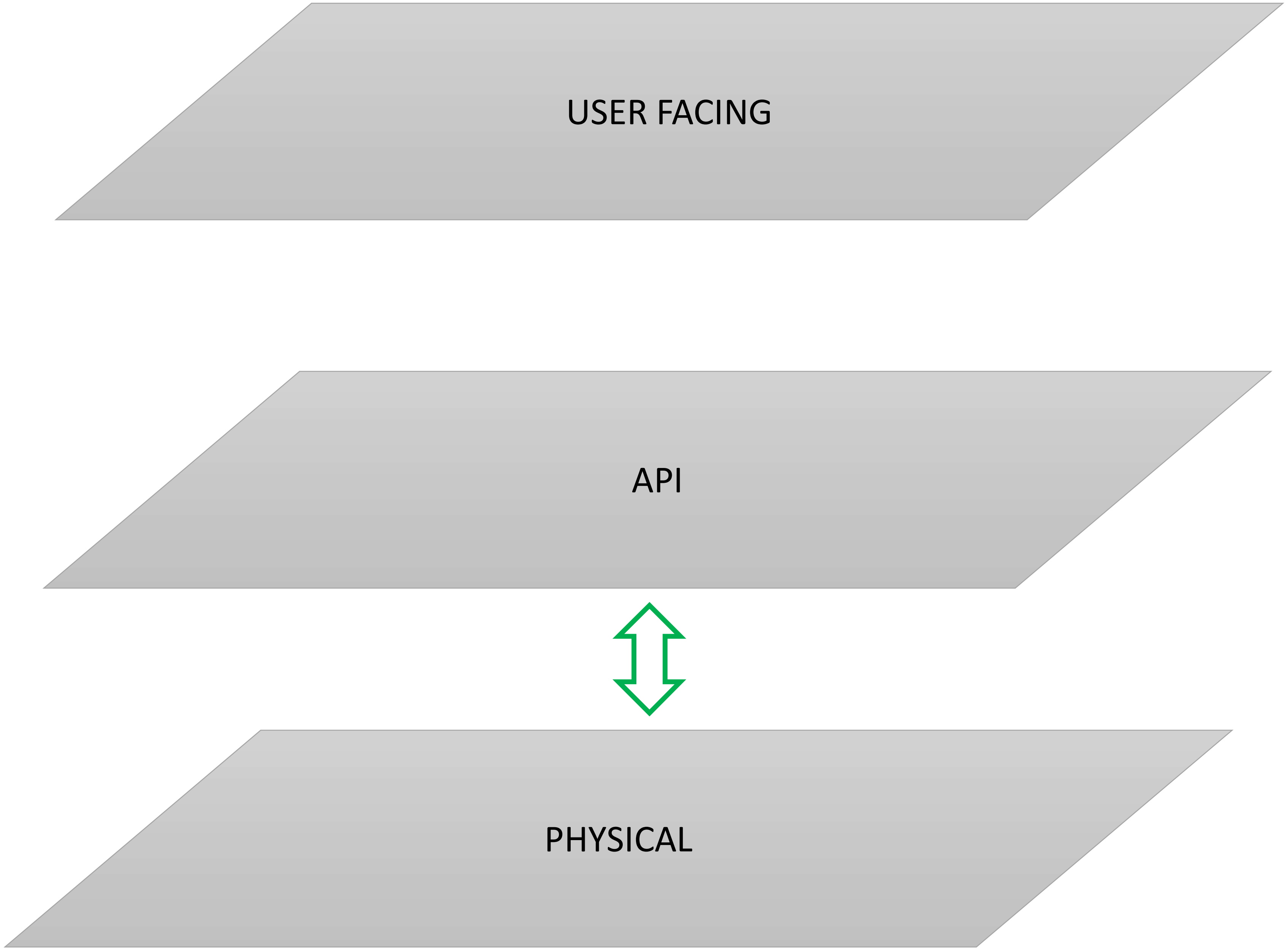
Context Setting



CONTROL

MECHANICAL

USER INTERFACE

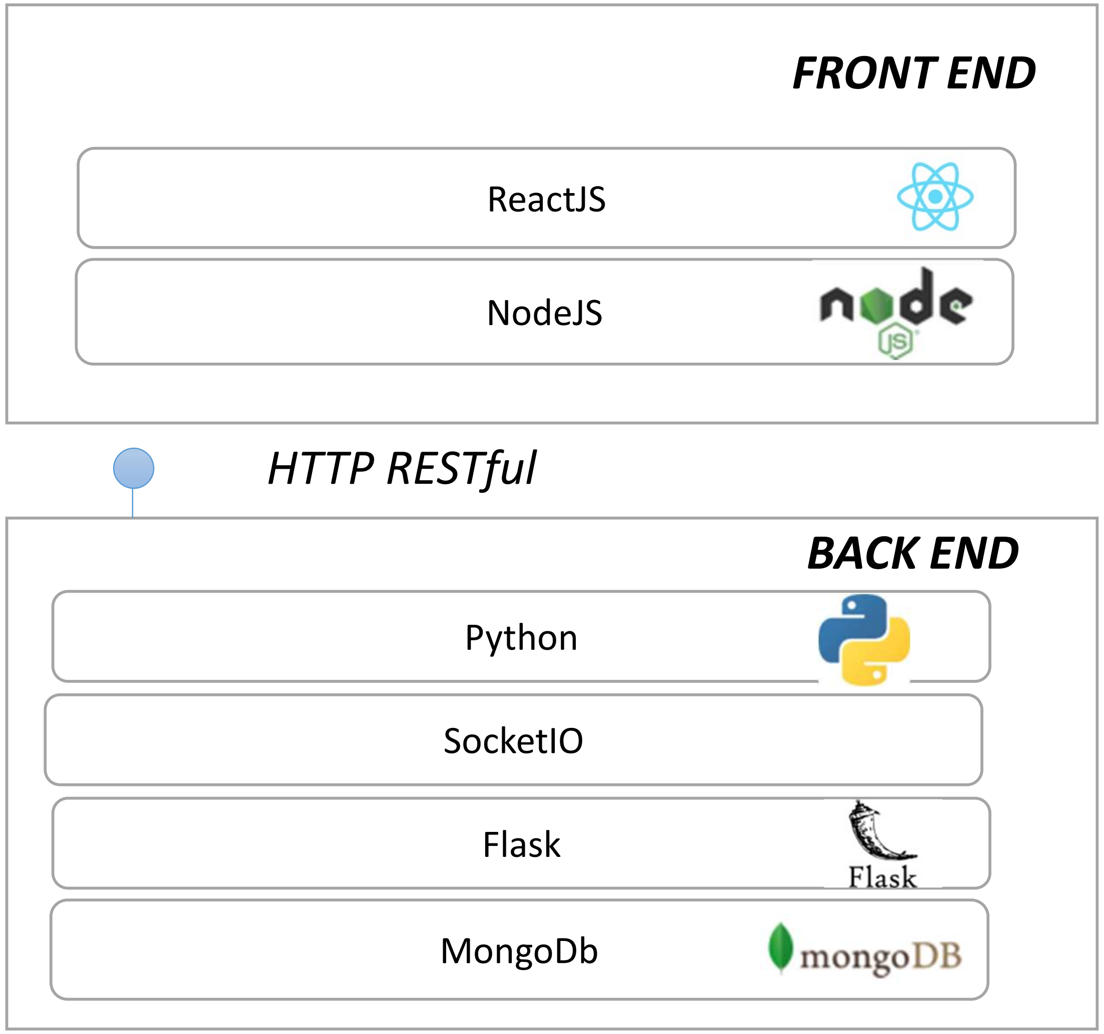


USER FACING

API

PHYSICAL

Overview| Architecture



FRONT END

ReactJS

NodeJS

HTTP RESTful

BACK END

Python

SocketIO

Flask

MongoDb

The Front End is build using ReactJS

Service/API Testing is done using Postman

The Services/Back End is built using Python with Flask microframework serving HTTP RESTful services.

Architecture – Layered View

Architecture | Back End

Routes define the service endpoints. They are linked to a URL in manage.py using add_url_rule app.add_url_rule('/api/engine', view_func=routes.startEngine)

Controllers expose the methods which can be exposed as service end points. They are invoked from route.

Architecture – Layered View

Architecture | Front End

Components are the way react organizes elements –Login, ProductsList are examples. They are defined as routes in App.js

Services make call to APIs and return response

Architecture – Layered View

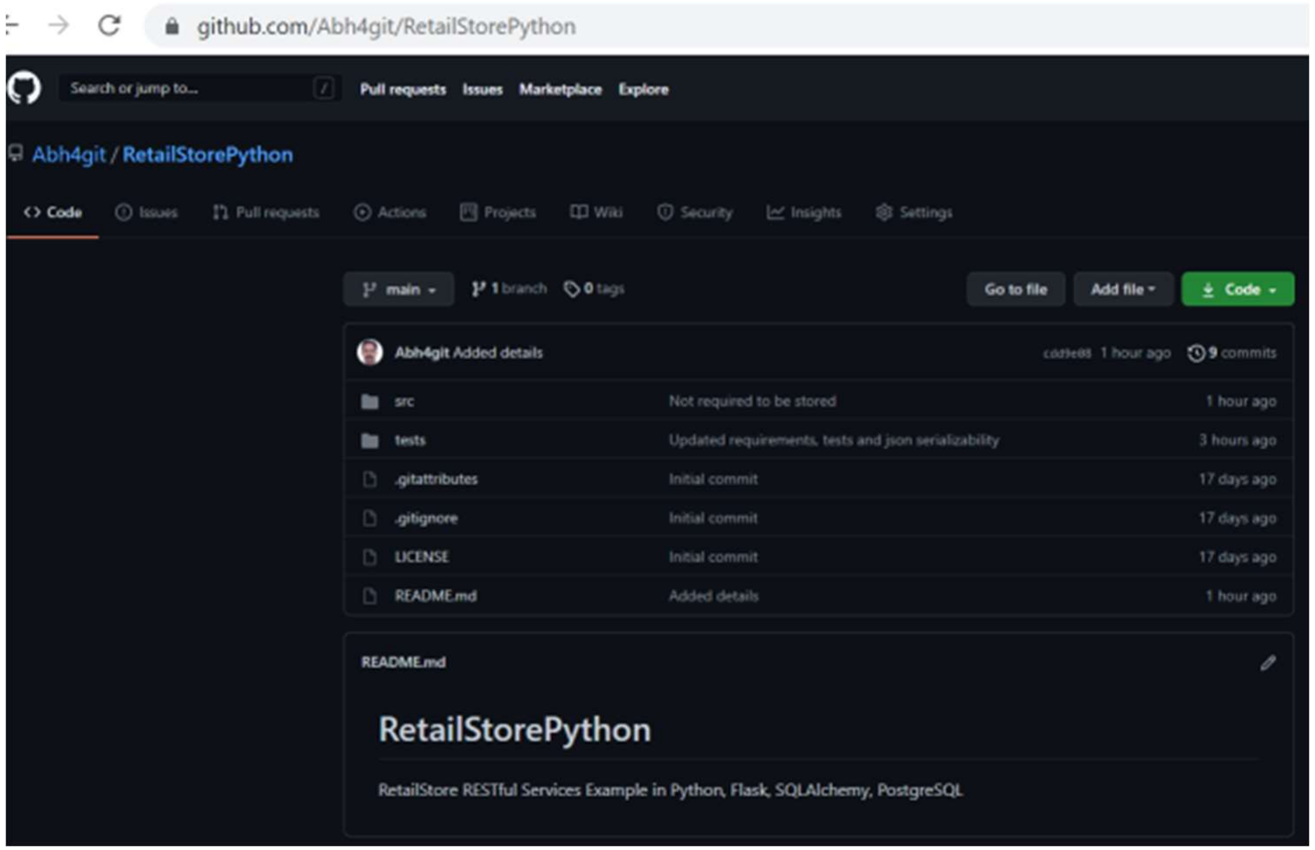
Architecture | Interactions

Components are the way react organizes elements –Login, ProductsList are examples. They are defined as routes in App.js

Services make call to APIs and return response

Architecture – Sequence Diagram

Getting Started | Back End



Abh4git Added details cdd9e88 1 hour ago 9 commits

src	Not required to be stored	1 hour ago
tests	Updated requirements, tests and json serializability	3 hours ago
.gitattributes	Initial commit	17 days ago
.gitignore	Initial commit	17 days ago
LICENSE	Initial commit	17 days ago
README.md	Added details	1 hour ago

README.md

RetailStorePython

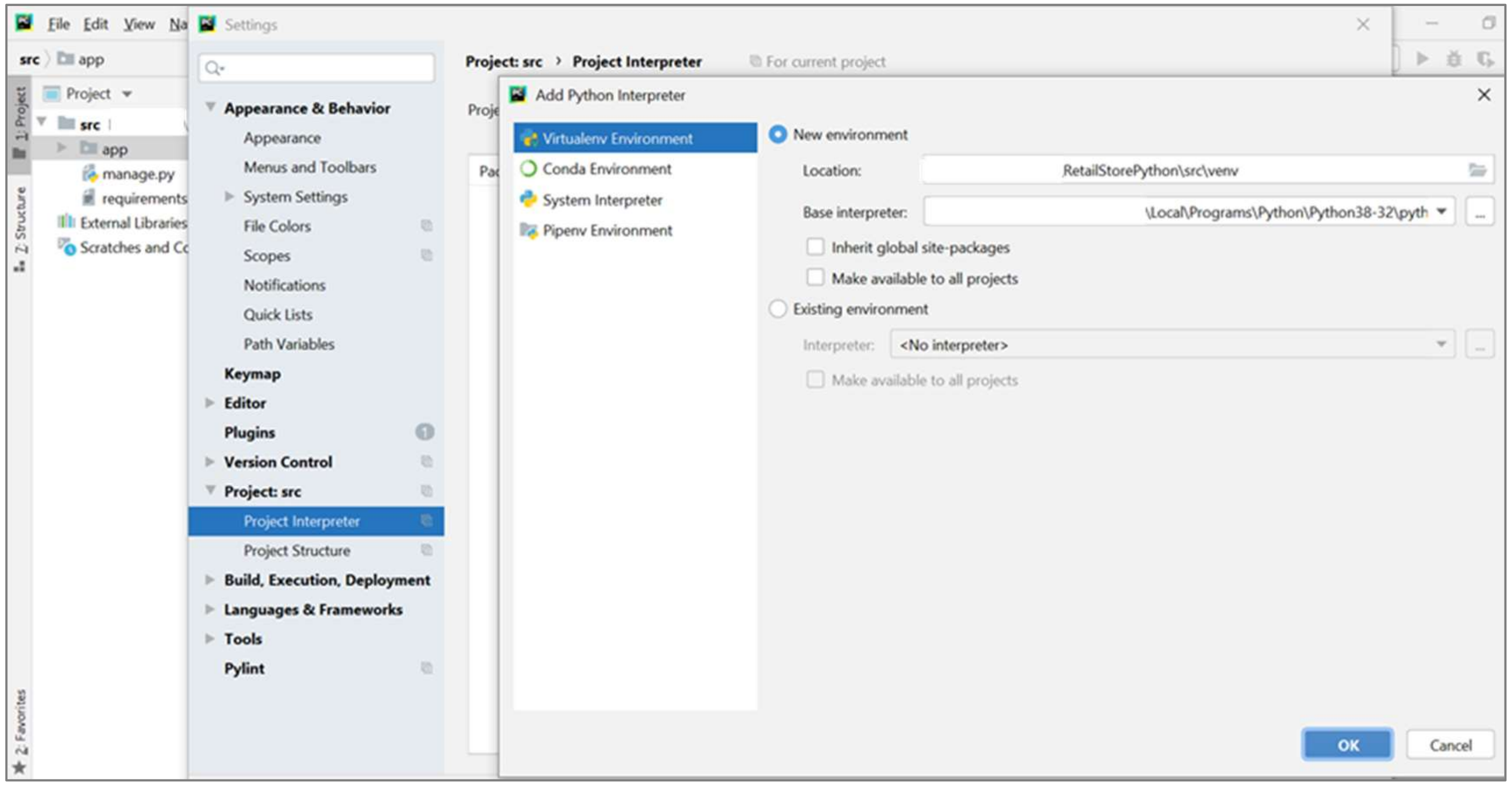
RetailStore RESTful Services Example in Python, Flask, SQLAlchemy, PostgreSQL

GitHub URL:

https://github.com/Abh4git/RetailStorePython

GitHub Repo for Back End

Getting Started | Back End



● New environment

Location: RetailStorePython\src\venv

Base interpreter: \Local\Programs\Python\Python38-32\pyth

□ Inherit global site-packages

□ Make available to all projects

○ Existing environment

Interpreter: <No interpreter>

□ Make available to all projects

1. *Open PyCharm in src folder and setup Interpreter and select Virtual Environment*

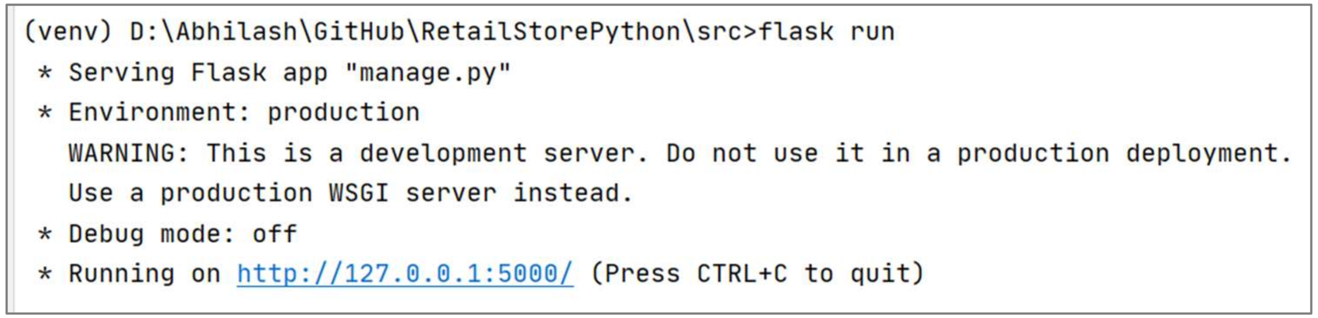
2. *Pip install as shown below:*

```
src>pip install -r requirements.txt
```

3. Set FLASK_APP environment variable like below
src> **set FLASK_APP=manage.py** *(In Linux use export)*

Back End running

Getting Started | Running Back End and Testing it



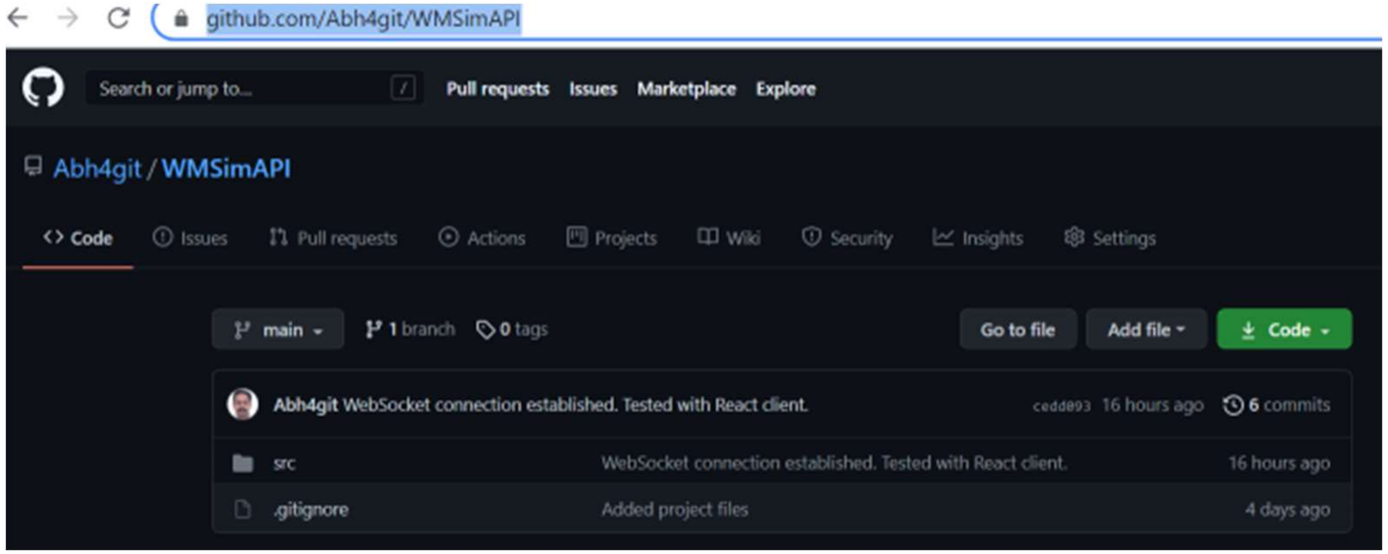
```
(venv) D:\Abhilash\GitHub\RetailStorePython\src>flask run
 * Serving Flask app "manage.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

1. *Start execution from terminal*

*src> **flask run***

The Test results show if the service is running fine.

Getting Started | Front End



Abh4git WebSocket connection established. Tested with React client. cedd093 16 hours ago 6 commits

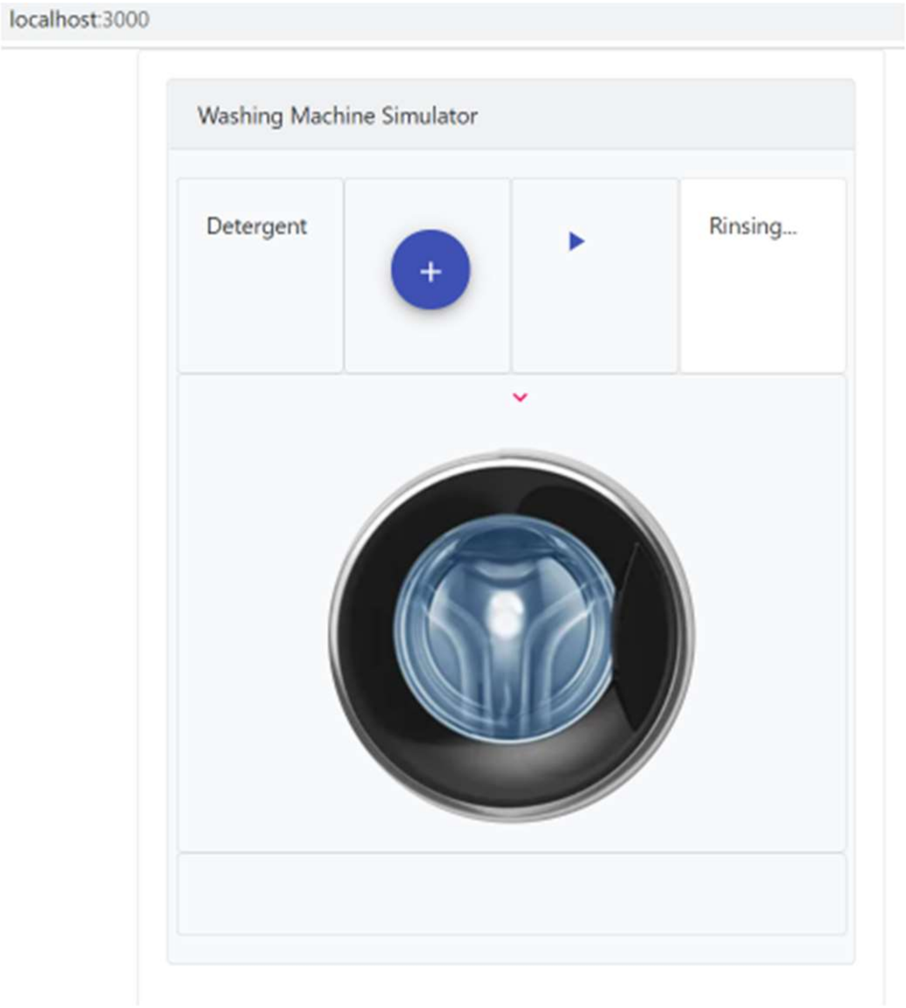
src	WebSocket connection established. Tested with React client.	16 hours ago
.gitignore	Added project files	4 days ago

GitHub URL:

https://github.com/Abh4git/WMSimAPI

GitHub Repo –Front End

Getting Started | Front End



Pre-requisite: NodeJs need to be installed:

1. *Install dependencies*

*src> **npm install***

2. Start program

*src> **npm start***

Executing the Front End

Summary| Architecture, Design and Getting it running

1. *Back End using Python, Flask, SocketIO*
2. *Front End using NodeJS, ReactJS*
3. *Step by step approach explained.*

Thank you!