SD1 & VT Industries

Group Members: Tracy Le, Alex Harms, Eric Mullen, Katelyn Lamison, Renee Mei Jing Teoh

Presentation Agenda

01 **Background**
A brief description on how VT Industries tests

02 **Problem Being Addressed**

03 **Design Decisions**

04 **Initial Implementation Research & Challenges**

05 **Progress Status & Remaining Timeline**

06 **Demo**
Showcase of data and demo of what has been done so far

01

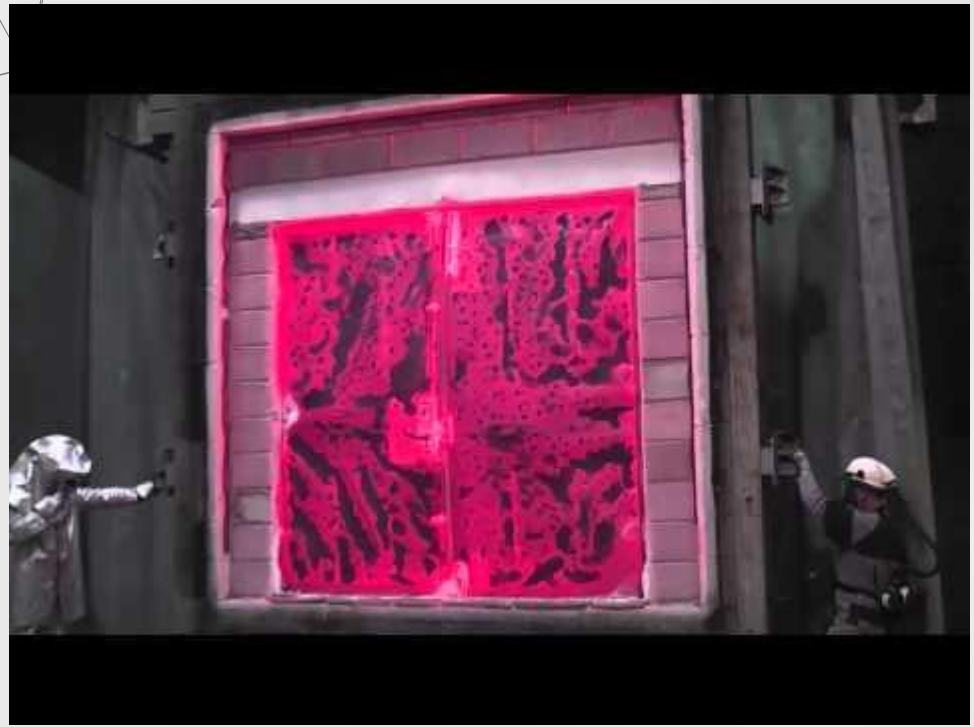
Background



Fire-Rated Doors

A door that can withstand 90 minutes of intense heat/fire on one side(up to 2000 °F) while still remaining cool to the touch on the other side. The door should be able to retain strength and sturdiness after the 90 minutes as well

Similar Test Process Video



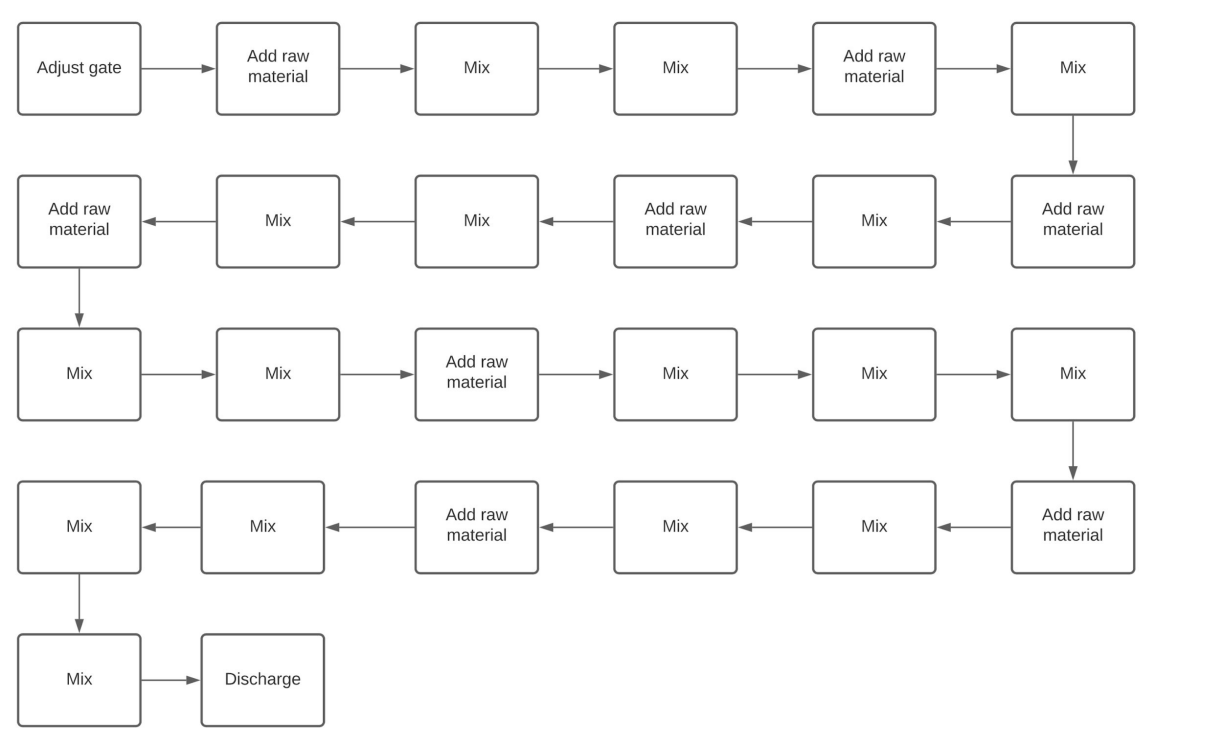
02

Problem Being Addressed

Given raw materials → find a way to show what properties of the door are outputted

If time allows.. → create an application that allows a user to input data and get expected properties

Overview of Manufacturing Process



03

Design Decisions

Design Decisions

Main project component:

- Machine learning
- Supervised learning
- Multiple models (one for each target)

Extended project component:

- Mobile application
- Interactive
- Immediate feedback
- Simple user interface



04

Initial Implementation Research

Initial Implementation Research

Main project component:

- Python
- Pandas, NumPy, scikit-learn/sklearn libraries
- Google CoLab & Data Studio
- Regression Models
- Algorithms: Decision trees (Random Forest)

Extended project component:

- Android Studio

Challenges

Challenges

- Delay in obtaining the data from the client
  - ★ Privacy issues
- Cleaning the data obtained
  - ★ Due to time constraints
  - ★ Unclear about manufacturing process and which data that will be affecting the results

05

Progress Status & Remaining Timeline

Progress Status

- Program that reads in spreadsheet, and replaces specific keywords
  - ★ For privacy purposes
- Obtained historical weather data from OpenWeather
- Understanding the data models
- Cleanup/Reformatting of data
- Research into feature importance/how to process data

Remaining Timeline

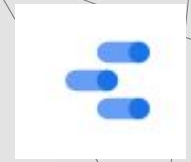
- Focus on understanding the client's data (Manufacturing process, data that potentially affect the fire door, etc.)
- Clean the data and show the data on a dashboar and make flowchart
- Match the data with historical weather data
- Use regression model to predict the optimum strength parameters
- Verify and make adjustment to the accuracy of the regression model



Jupyter Notebook



Google Colaboratory

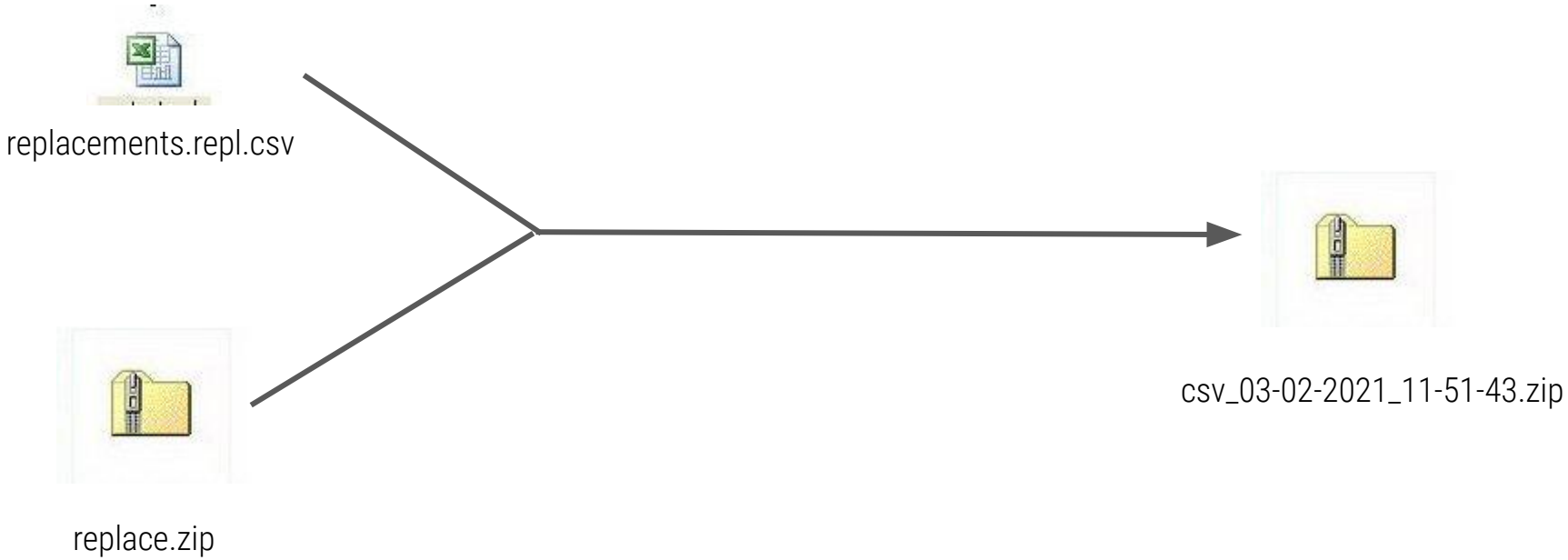


Data Studio

06
Data

Replace All Program

[Alex H]


replacements.repl.csv
replace.zip
csv_03-02-2021_11-51-43.zip

Data Parsing & Cleanup

[Alex H]

	A	B	C	D	E	F	G	H	I	J	K
1	textbox1	batchid	textbox45	Recipefilename	txtShowAll	area	textbox42	textbox47	starttimeexecution	endtimeexecution	textbox8

3	unitname	textbox17	textbox18	textbox20	Duration	txtOperation	textbox27	textbox28		

	textbox29	txtPhase	textbox103	textbox104	textbox105	starttime_1	paramname	endtime	textbox112

	A	B	C	D	E	F	G	H	I	J	K	L
1	batch_id	batch_start	batch_end	batch_duration	operation	start	end	duration	param_name	param_type	value	unit

```go
const (
    ReportParameter = ParamType("Report Parameter")
    RecipeParameter = ParamType("Recipe Parameter")
)

type ParamType string

type BatchDetail struct {
    BatchID    int           `json:"batch_id"`
    StartTime  time.Time     `json:"batch_start"`
    EndTime    time.Time     `json:"batch_end"`
    Duration   time.Duration `json:"duration"`
    Operations []Operation   `json:"operations"`
}

type Operation struct {
    Operation string           `json:"operation"`
    Start     time.Time        `json:"start"`
    End       time.Time        `json:"end"`
    Duration  time.Duration    `json:"duration"`
    Params    []OperationParam `json:"params"`
}

type OperationParam struct {
    Name      string     `json:"param_name"`
    ParamType ParamType  `json:"param_type"`
    Value     data.Value `json:"value"`
    Unit      string     `json:"unit,omitempty"`
}
```

Batch Detail

```go
const (
    LowDensity    = CoreDensity("low")
    MediumDensity = CoreDensity("medium")
    HighDensity   = CoreDensity("high")
)

type CoreDensity string

type DataPoint struct {
    Density CoreDensity           `json:"density"`
    Values  map[string]data.Value `json:"values"`
}

type ShippingQuality []DataPoint
```

QA Values

```go
type Value struct {
    // Can have decimal, so I think this would be the best way to store it
    Number *float64 `json:"num_val"`
    Str    string   `json:"str_val"`
}
```

Value

Weather Collection

[Eric M & Alex H]

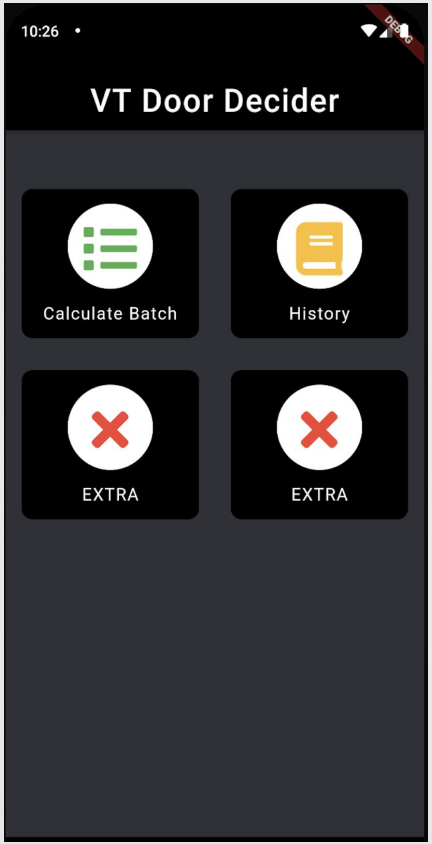
```
date reg,temp,temp_min,temp_max,humidity
2019-01-01 23:00:00 +0000 UTC,19.48,13.73,24.39,81.71
2019-01-02 23:00:00 +0000 UTC,17.34,13.39,21.09,81.29
2019-01-03 23:00:00 +0000 UTC,26.88,23.83,29.56,80.5
2019-01-04 23:00:00 +0000 UTC,35.97,32.48,38.69,78.08
```

date reg	temp	temp_min	temp_max	humidity
2019-01-01	19.48	13.73	24.39	81.71
2019-01-02	17.34	13.39	21.09	81.29
2019-01-03	26.88	23.83	29.56	80.5
2019-01-04	35.97	32.48	38.69	78.08
2019-01-05	37.24	34.04	40.2	84.88

Application Prototype

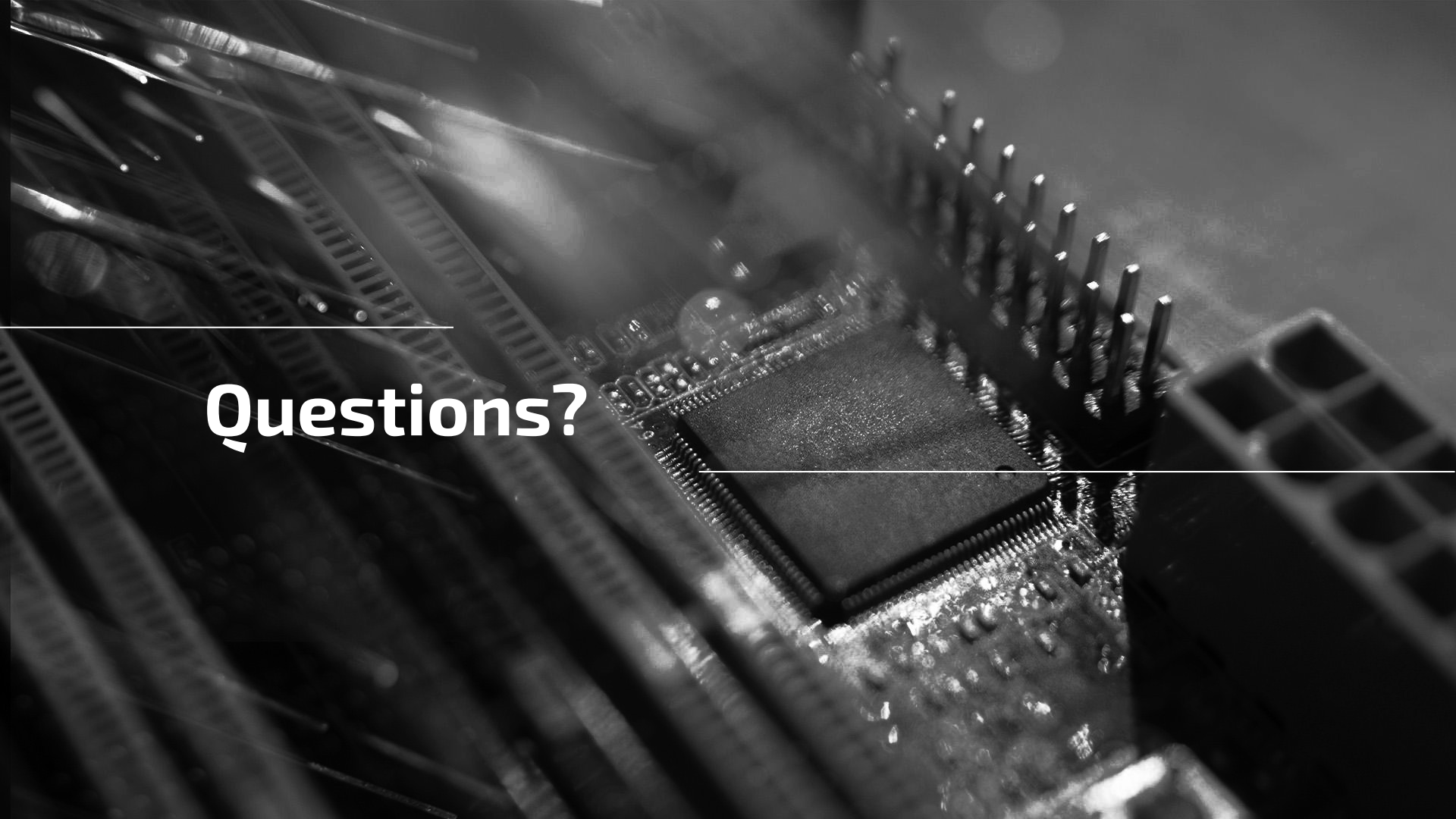
[Eric M]

Home Screen



Calculation Screen




Questions?