Capstone Project:

Financial Analysis Of Ventilator Manufacturing

Table of content

01 Problem Statement

02 Hypothesis

03 Hypothesis Proving

04 Solution

05 Conclusions

06 Improvements

07 Architecture/Design/Dev

Problem statement

Feasible to manufacture "ventilators" with no financial risks

NOTE: Low cost ventilator will be built using MIT research paper

shortage of ventilators

Manufacture Ventilators

pandemic situation

01 Financial risk

02 Machinery

03 Capacity

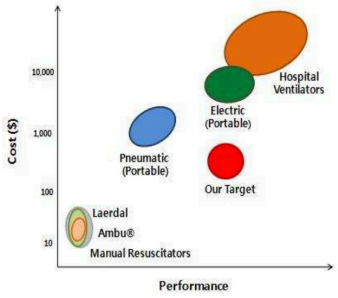
04 Low cost

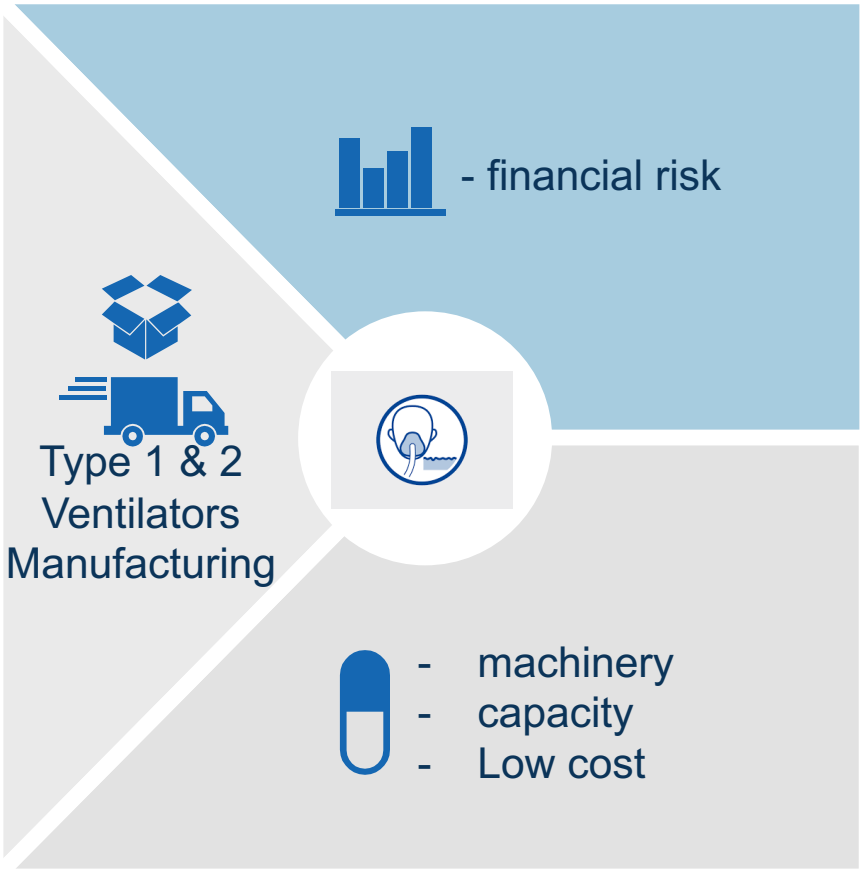
Hypothesis

Hypothesis

H_0: Feasible to build "ventilators" with no financial risks.

H_1: Not feasible to build "ventilators" with no financial risks.





Assumptions

- no defects in manufacturing
- 100% capacity of worker (i.e. no sickness/vacations)

No financial risk	Utilize existing machinery
Type 1 or 2 or both	40 hours per day working hours
Low cost ventilators (MIT based – low cost portable ventilators)	Utilize existing work force capacity

Hypothesis proving

Hypothesis proving

The hypothesis will be ACCEPTED/REJECTED using **Linear Programing (LP) technique**

Linear programing technique will be used in finding optimial manufacturing of “Ventilators” with given/provided constraints.

Why linear programing?

- The ventilator manufacturing problem deals with the maximisation of profit margin from investment
- It requires allocation of optimal workforce (capacity/manpower) to produce ventilators in such a way there is no overtime cost required to be paid
- Linear programming helps in attaining the optimal value of Type X/Y ventilators to be manufactured to maximise profits
- It also helps in performing “what if analysis”
  - addressing defects in manufacturing
  - Workforce/manpower/capacity not available

Solution

Table showing profit per unit in building ventilators of Type X/Y along with available capacity

Description	***Type X***	***Type Y***
profit per unit	**25**	**30**
capacity per hour	**200**	**140**
capacity per week	**6000**	**4000**

Linear programing modelling – “Ventilator manufacturing”

Step 1 — **Decision variables**

$Number\ of\ Type\ X\ ventilators: X$

$Number\ of\ Type\ Y\ ventilators: Y$

Step 2 — **Constraints**

$X \leq 6000$

$Y \leq 4000$

$\frac{X}{200} + \frac{Y}{140} \leq 40$

Step 3 — **Objective Function**

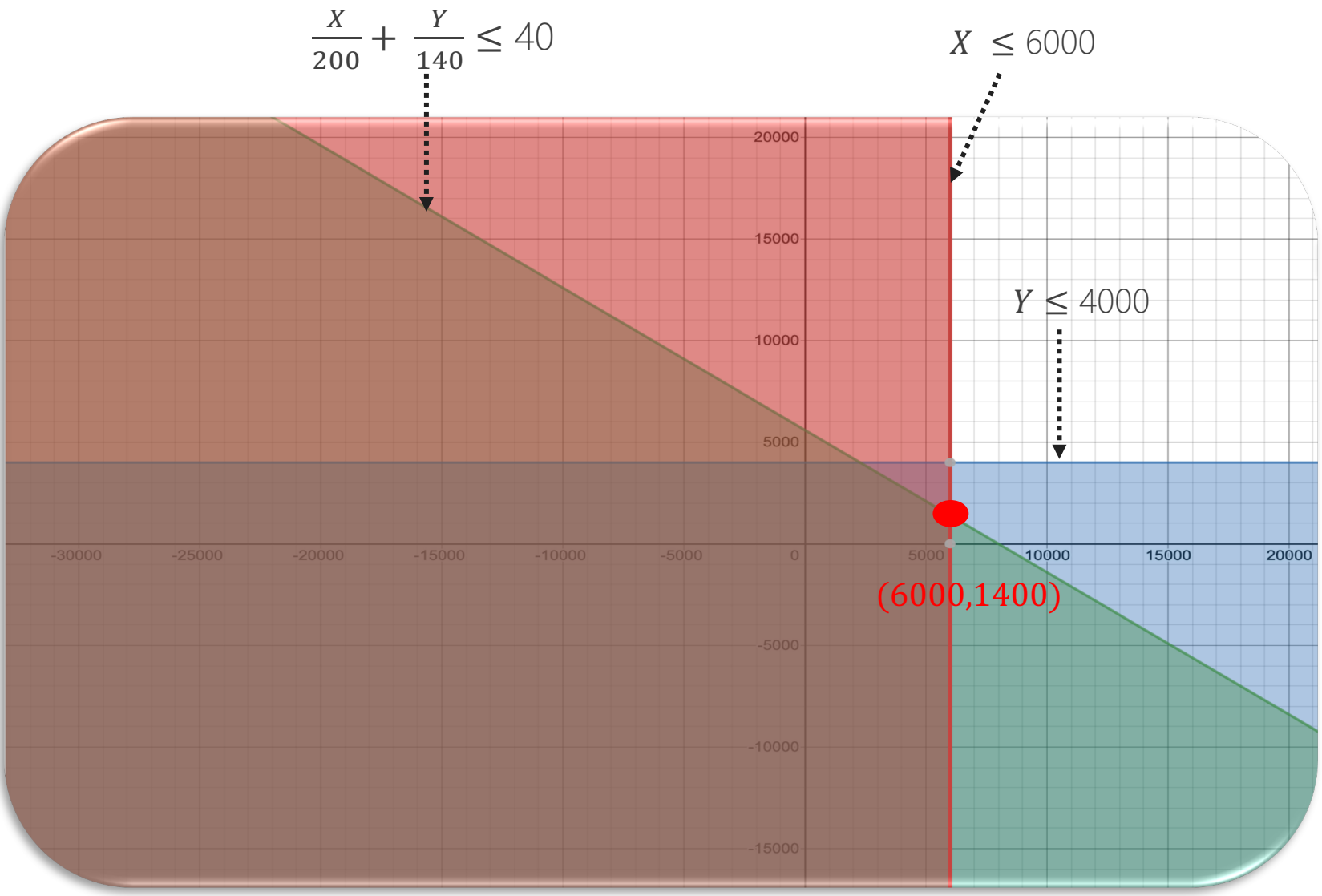
$Maximize\ profit(25 * X + 30 * Y)$

Finding optimal X,Y and profit

S No	Technique	Description	Solver
1.	Graphically	Graphical technique can be used to find the optimal values of X, Y to maximize the profit	NA
2.	Programatically using OR Tools	OR-Tools is an open source software suite for optimization, tuned for tackling linear programming, and constraint programming. Using OR tools to find optimal values of X, Y to maximize the profit. This technique will use python for LP and configurable solver.	SCIP
3.	AMPL	AMPL provides modelling language to express linear programming chores, hence it will be used to find optimal values of X, Y to maximize the profit. It provides various configurable solver like CPLEX, MINOS etc	MINOS
4.	New LP modeling language	Using JAVACC a new constraint modeling programing lanauage will be created. This english like langauge will be used to define, linear programing chores and then will be converted to OR tools. A solver is configured as *solver SCIP*. Refer to to section "Architecture/Design/Build" for more details Github source code (complete framework with reactjs based editor)	Configurable

LP Graphically

Graphical output



Interpretation

$X = 6000 = No\ of\ Type\ X\ ventilators$

$Y = 1400 = No\ of\ Type\ Y\ ventilatos$

$$Profit = 25 * 6000 + 30 * 1400$$
$$= 192000$$

LP using OR Tools

Code

```python
from ortools.linear_solver import pywraplp
def main():
    solver = pywraplp.Solver.CreateSolver('SCIP')

    infinity = solver.infinity()
    # type x
    x = solver.IntVar(0.0, infinity, 'x')
    # type y
    y = solver.IntVar(0.0, infinity, 'y')

    # constraints
    # type x capacity
    solver.Add(x <= 6000)
    # type y capacity
    solver.Add(y <= 4000)
    # weekly hours available for manufacturing
    # solver.Add(x +  y  <= 13600)
    solver.Add(x / 200 + y / 140 <= 40)

    # objective function
    solver.Maximize(25 * x + 30 * y)
    status = solver.Solve()
```

Output

```
=========================================================
                        Solution
=========================================================
Objective value          = 192000
Type X                   = 6000.0
Type Y                   = 1400.0
Slack on weekly hours    = 0.0
Slack on X Type demand   = 0.0
Slack on Y Type demand   = 2600.0
```

Interpretation

$$X = 6000 = No\ of\ Type\ X\ ventilators$$

$$Y = 1400 = No\ of\ Type\ Y\ ventilatos$$

$$Profit = 192000$$

LP using AMPL

Code

```
var x >= 0;
var y >= 0;

maximize profit: 25 * x + 30 * y;


subject to weekly:  (x / 200) +  (y / 140) <= 40;
subject to demandx:  x <= 6000;
subject to demandy:  y <= 4000;
```

Output

```
Console
AMPL
ampl: model finance.mod;
ampl: solve;
MINOS 5.51: optimal solution found.
2 iterations, objective 192000
ampl: display profit, x, y, demandx.slack, demandy.slack;
profit = 192000
x = 6000
y = 1400
demandx.slack = 0
demandy.slack = 2600
```

Interpretation

$X = 6000 = No\ of\ Type\ X\ ventilators$

$Y = 1400 = No\ of\ Type\ Y\ ventilatos$

$Profit = 192000$

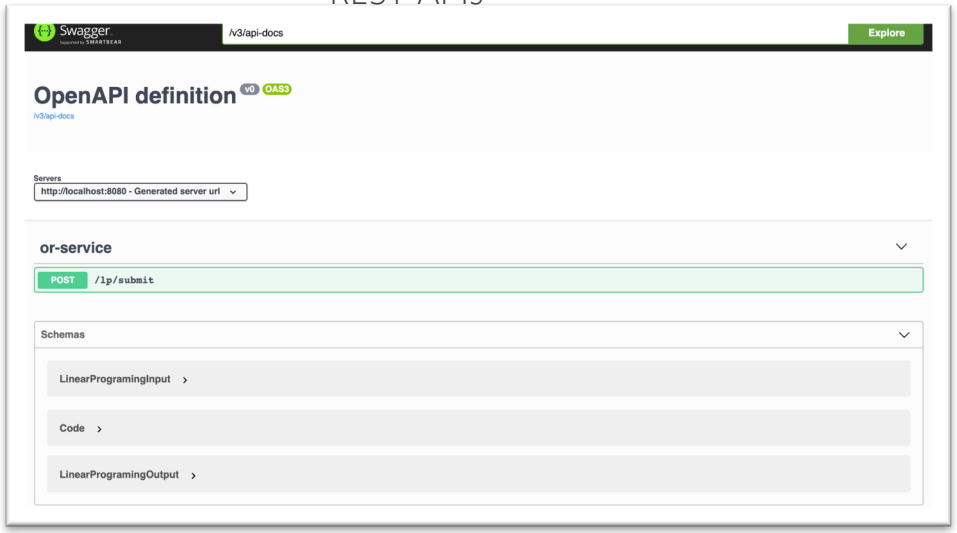
Using new LP modelling language

Code

New OR Language

```
solver: GLOP
variables:  x,y
maximize: 25 * x + 30 * y
constraints:
     weekly: 0.005 * x + 0.007142857 * y <= 40
     demandx:  1 * x <= 6000
     demandy:  1 * y <= 4000
```

REST APIs



Output


```
{
  "solution": "OPTIMAL",
  "objectiveFunctionValue": 192000.00084000002,
  "solutionValue": {
    "x": "6000",
    "y": "1400"
  },
  "slacks": {
    "demandy": 2600,
    "demandx": 0,
    "weekly": 0
  },
  "code": {
    "solver": "GLOP",
    "variables": " x,y",
    "objectiveFunction": "maximize: +25.0 * x+30.0 * y",
    "constraints": [
      "weekly: +0.005 * x +0.007142857 * y <= 40.0",
      "demandx: +1.0 * x <= 6000.0",
      "demandy: +1.0 * y <= 4000.0"
    ]
  }
}
```


Interpretation

$$X = 6000 = No\ of\ Type\ X\ ventilators$$

$$Y = 1400 = No\ of\ Type\ Y\ ventilatos$$

$$Profit = 192000$$

Conclusions

Conclusion/Summarization

- Hypothesis:

  H_0: Feasible to build "ventilators" with no financial risks.

  H_1: Not feasible to build "ventilators" with no financial risks.

- Given:
  - no defects in manufacturing
  - 100% capacity of worker (i.e. no sickness/vacations)
  - Objective function
  - constraints

Objective Function

$$Maximize\ profit(25 * X + 30 * Y)$$

Constraints

$$X \leq 6000$$

$$Y \leq 4000$$

$$\frac{X}{200} + \frac{Y}{140} \leq 40$$

- Optimal solution:
  - No. of Type X ventilators = 6000
  - No. of Type Y ventilators = 1400
  - Profit in manufacturing Type X and Type Y (objective function value) = 192000
- Interpretation/Conclusions:
  - *Profitable to build ventilators of Type X and Type Y*
  - *There is slack of 2600 in building Type Y ventilators, however these is no capacity left*
  - *Since ventilator of Type Y is cheaper to make, the profit can be increased if more weekly capacity of it is purchased*
- Null Hypothesis H_0:
  - True

Improvements

Improvements

- Following are proposed additions to the model:
  - Include configurable defect percentage in manufacturing of ventilators
    - $defects\ in\ Type\ X\ ventailators\ X_d\%$
    - $defects\ in\ Type\ Y\ ventailators\ Y_d\%$

  - Include configurable sick leave and vacations taken by workers. $\mathbf{S\%}$ represent combined total in percentage

“Ventilator manufacturing "with defect & leaves %age

Step 1 — **Decision variables**

$Number\ of\ Type\ X\ ventilators: X$

$Number\ of\ Type\ Y\ ventilators: Y$

$Sick\ leaves + Vaccation\ leaves = S\ \%$

$Defect\ in\ Type\ X\ ventilators = X_d\%$

$Defect\ in\ Type\ Y\ ventilators = Y_d\%$

Step 2 — **Constraints**

$$X \leq 6000 - \frac{X_d}{100} * 6000$$

$$Y \leq 4000 - \frac{Y_d}{100} * 4000$$

$$\frac{X}{200} + \frac{Y}{140} \leq 40 - \frac{S}{100} * 40$$

Step 3 — **Objective Function**

$$Maximize\ profit(25 * X + 30 * Y)$$

Defect rate and non available worker percentage

- *From* ***<u>six – sigma</u>*** *following values were derived:*
  - $defects\ in\ Type\ X\ ventailators\ X_d = \mathbf{0.5}\%$ = 30
  - $defects\ in\ Type\ Y\ ventailators\ Y_d = \mathbf{0.5}\%$ = 20
  - NOTE: in total normal acceptable defect percentage is 1%

- *From US* ***<u>bureau</u>*** *following value is derived:*
  - $Sick\ Leaves + Vaccations\ Leaves = S = \mathbf{10.0}\%$ i.e. on average around $\mathbf{4}$ hours per week

"Ventilator manufacturing" with defect & leaves %age

Step 1 — **Decision variables**

$Number\ of\ Type\ X\ ventilators: X$

$Number\ of\ Type\ Y\ ventilators: Y$

$Sick\ leaves + Vaccation\ leaves = S\ \% = 10\% = 4\ per\ week$

$Defect\ in\ Type\ X\ ventilators = X_d\% = 0.5\% = 30$

$Defect\ in\ Type\ Y\ ventilators = Y_d\% = 0.5 = 20$

Step 2 — **Constraints**

$X \leq 5970$

$Y \leq 3980$

$\frac{X}{200} + \frac{Y}{140} \leq 36$

Step 3 — **Objective Function**

$Maximize\ profit(25 * X + 30 * Y)$

Linear Programing using OR-tools– ”Ventilator manufacturing”

Code

```python
from ortools.linear_solver import pywraplp
def main():
    solver = pywraplp.Solver.CreateSolver('SCIP')

    infinity = solver.infinity()
    # type x
    x = solver.IntVar(0.0, infinity, 'x')
    # type y
    y = solver.IntVar(0.0, infinity, 'y')

    # constraints
    # type x capacity
    solver.Add(x <= 5970)
    # type y capacity
    solver.Add(y <= 3980)
    # weekly hours available for manufacturing
    # solver.Add(x +  y  <= 13600)
    solver.Add(x / 200 + y / 140 <= 36)

    # objective function
    solver.Maximize(25 * x + 30 * y)
    status = solver.Solve()
```

Output

```
==========================================================
                        Solution
==========================================================
Objective value          = 175080
Type X                   = 5970.0
Type Y                   = 861.0
Slack on weekly hours    = 4.0
Slack on X Type demand   = 30.0
Slack on Y Type demand   = 3139.0
```

Interpretation

$X = 5970 = No\ of\ Type\ X\ ventilators$

$Y = 861 = No\ of\ Type\ Y\ ventilatos$

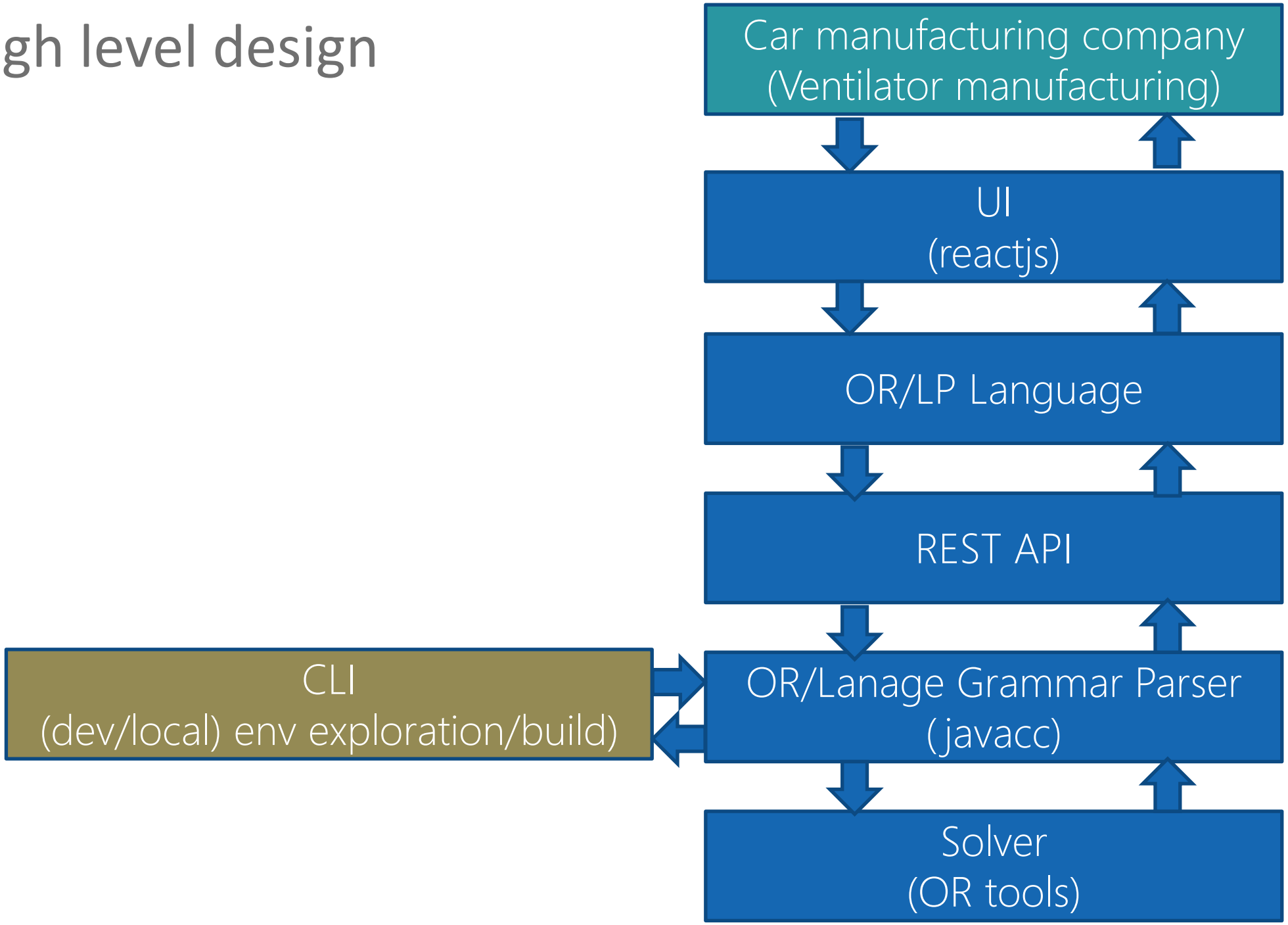
$Profit = 175080$

Architecture/Design/Dev

Design principles

- Provide inutitive constraint programing lanauge (grammar), thereby boosting productivity in devlopment natively and on cloud
- Langauge agnostic
- Cloud agnostic
- Cloud native
- Micro-service architecture,. i.e., once deployed, exeucting differrent linear programing lanauge is as simple as calling REST APIs. This also makes "*what if analysis*" very easy
- DRY
- SOLID
- Single responsibility
- Layered architecture
- Scalable
- Containerization (aides in scaliability & also make it cloud aganostic and native)
- Building model is 10% of complete model software development, hence design choices should fascilitate in CI/CD automation

High level design



OR language grammar

- JavaCC (Java Compiler Compiler) is an open-source parser generator and lexical analyser generator written in the Java programming language. JavaCC is similar to yacc in that it generates a parser from a formal **grammar** written in EBNF notation. Unlike yacc, however, JavaCC generates top-down parsers.

Grammar

```
PARSER_END(ORParser)

<*> SKIP: {
    <" " | "\t">
}

TOKEN : {
    < ALPHANUM : (["a"-"z", "A"-"Z", "_"])+ (["0"-"9", "a"-"z", "A"-"Z", "_"])* >
}
TOKEN : {
    < NUM : (["0"-"9"])+ ("." (["0"-"9"])+)?>
}

TOKEN : {
    < NUMVAR : (<NUM> "*")? <ALPHANUM> >
}

TOKEN : {
   < OPERA      : ( "+" | "-" ) >
}

TOKEN : {
   < LOGICAL_OPERA      : ( ">" | ">=" | "<" | "<=" ) >
}

TOKEN : {
   < CRLF: (["\r","\n"])+ >
}

TOKEN:
{
    < SOLVER: "solver:" >
  | < VARIABLES: "variables:" >
  | < FUNCTION_TYPE: ("maximize:" | "minimize:") >
  | < CONSTRAINT: "constraints:" <CRLF> >

}
```

Example

```
solver: GLOP
variables: wrenches,pliers
maximize: 0.13 * wrenches + 0.10 * pliers
constraints:
    steel: 1.5 * wrenches + 1 * pliers <= 27000
    molding: 1.0 * wrenches + 1 * pliers <= 21000
    assembly: 0.3 * wrenches + 0.5 * pliers <= 9000
    demand1: 1 * wrenches <= 15000
    demand2: 1 * pliers <= 16000
```

Why new LP language?

- English like lanauge to express LP chores
- No vendor locking
- Solver agnostic
    - Currently using OR tools
- Fascilates in building LP as Service design paradim/micro-service architecutre
- Less learning curve

LP language REST APIs – Open APIs 3.0 Specification (UI)

http://localhost:8080/swagger-ui.html

Swagger
Supported by SMARTBEAR

/v3/api-docs

Explore

OpenAPI definition v0 OAS3

/v3/api-docs

Servers

http://localhost:8080 - Generated server url

or-service

POST /lp/submit

Schemas

```
LinearProgramingInput {
    code                    string
}

Code {
    solver                  string
    variables               string
    objectiveFunction       string
    constraints
                             [string]
}

LinearProgramingOutput {
    solution                string
    objectiveFunctionValue  number($double)
    solutionValue
                             {
                               < * >:          string
                            }
    slacks
                             {
                               < * >:          number($double)
                            }
    code                    Code {
                                solver          string
                                variables       string
                                objectiveFunction string
                                constraints
                                 > [...]
                            }
}
```

LP language online editor

http://localhost:3000

OR Experiment

Name

FinanceCapstoneProject

OR Language

```
solver: SCIP
variables:  x,y
maximize: 25 * x + 30 * y
constraints:
    weekly: 0.005 * x + 0.007142857 * y <= 40
    demandx:  1 * x <= 6000
    demandy:  1 * y <= 4000
```


```
solver: SCIP
variables:  x,y
maximize: 25 * x + 30 * y
constraints:
    weekly: 0.005 * x + 0.007142857 * y <= 40
    demandx:  1 * x <= 6000
    demandy:  1 * y <= 4000
```


Author

Gaurav

Execute

Output


```
{
  "solution": "OPTIMAL",
  "objectiveFunctionValue": 192000.00084000002,
  "solutionValue": {
    "x": "6000",
    "y": "1400"
  },
  "slacks": {
    "demandy": 2600,
    "demandx": 0,
    "weekly": 0
  },
  "code": {
    "solver": "SCIP",
    "variables": " x,y",
    "objectiveFunction": "maximize: +25.0 * x+30.0 * y",
    "constraints": [
      "weekly: +0.005 * x +0.007142857 * y <= 40.0",
      "demandx: +1.0 * x <= 6000.0",
      "demandy: +1.0 * y <= 4000.0"
    ]
  }
}
```

Development resources/information

- The complete source code of framework built to create Linear Programing language can be found here.
- Technology matrix

S No	Technology
1.	JAVACC 7.0.9
2.	REACT JS 17.0.1
3.	PYTHON 3.9.0
4.	JAVA 14.0.2
5.	OR Tools 6.5
6.	Kubernetes
7.	Terraform
8.	Maven 3.6.3
9.	IntelliJ
10.	pip
11.	pipenv
12.	npm

How to run?

- Build project

```
mvn clean install
```

- Run

```
java –jar target/or-language-0.0.1-SNAPSHOT.jar
```

- Run LP editor UI

```
PORT=3000 npm start
```

- Link to access LP editor

```
http://localhost:3000
```

- Link to access REST API (swagger)

```
localhost:8080/swagger-ui.html
```

References

References

- Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. 2010. *Linear Programming and Network Flows, 4th ed*. John Wiley and Sons, Inc., Hoboken, NJ. *Grad level*.
- Bertsimas, D. and Tsitsiklis, J. N. 1997. *Introduction to Linear Optimization*. Athena Scientific, Nashua, NH. *Graduate-level text on linear programming, network flows, and discrete optimization*.
- Chvatal, V. 1983. *Linear Programming*. W. H. Freeman, New York. *Undergrad or grad*.
- Cook, W. J., Cunningham, W. H., Pulleyblank, W. R. and Schrijver, A. 1997. *Combinatorial Optimization*. Wiley Interscience, New York.
- Fang, S.-C. and Puthenpura, S. 1993. *Linear Optimization and Extensions: Theory and Algorithms*. Prentice Hall, Upper Saddle River, NJ.
- Bertsimas, D. (2004) Data, Models, and Decisions: The Fundamentals of Management Science