

*Due to a lack of understanding of the project this section will be dedicated to the professor in proving our abilities that we know how to code. Each student will add their own documentation of what they have done as well as their diagrams.

Marcus Bedeau

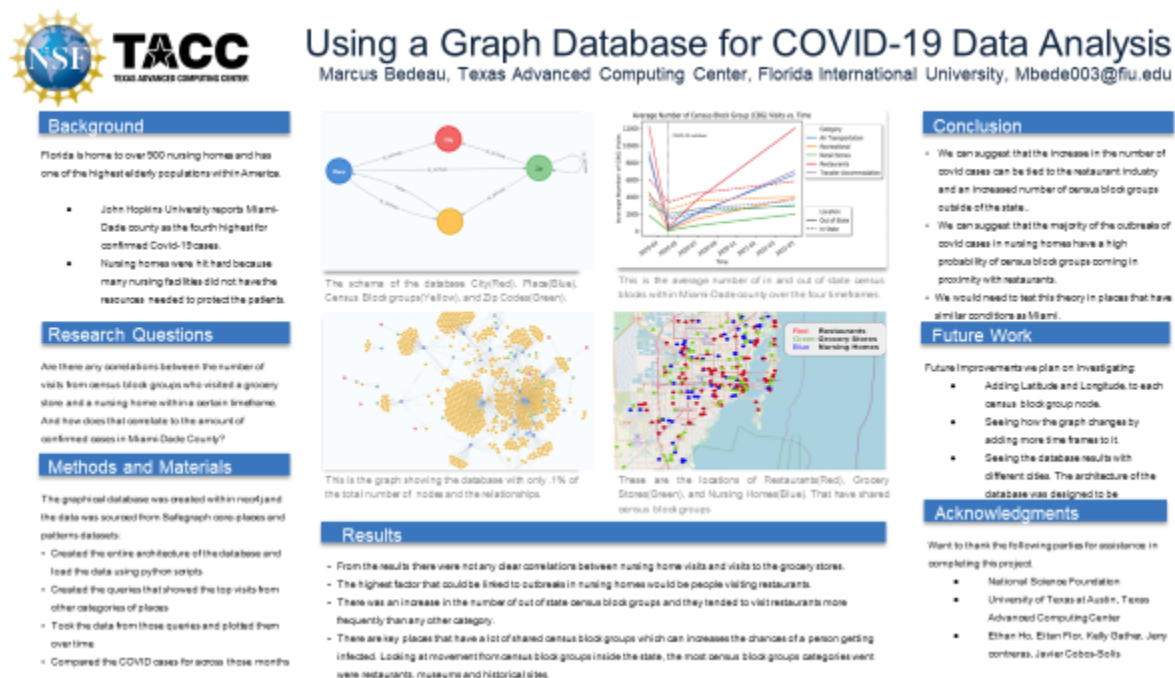
Pipeline for SafeGraph COVID-19 mobility data to Neo4j Database

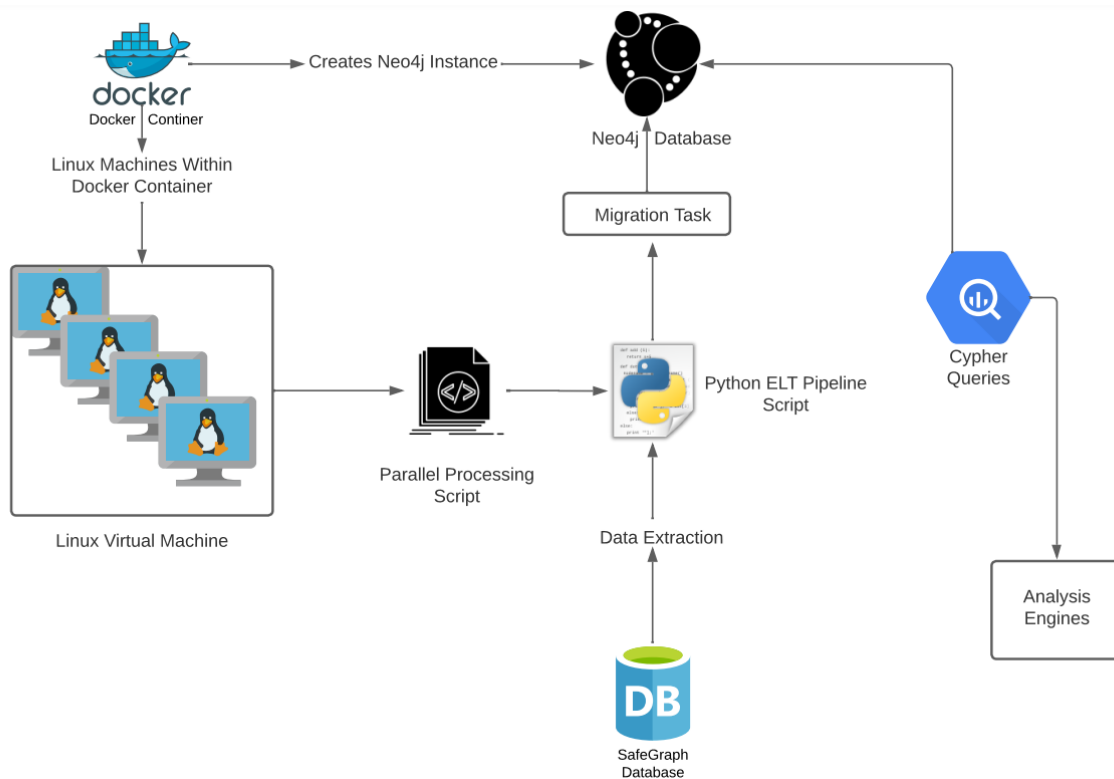
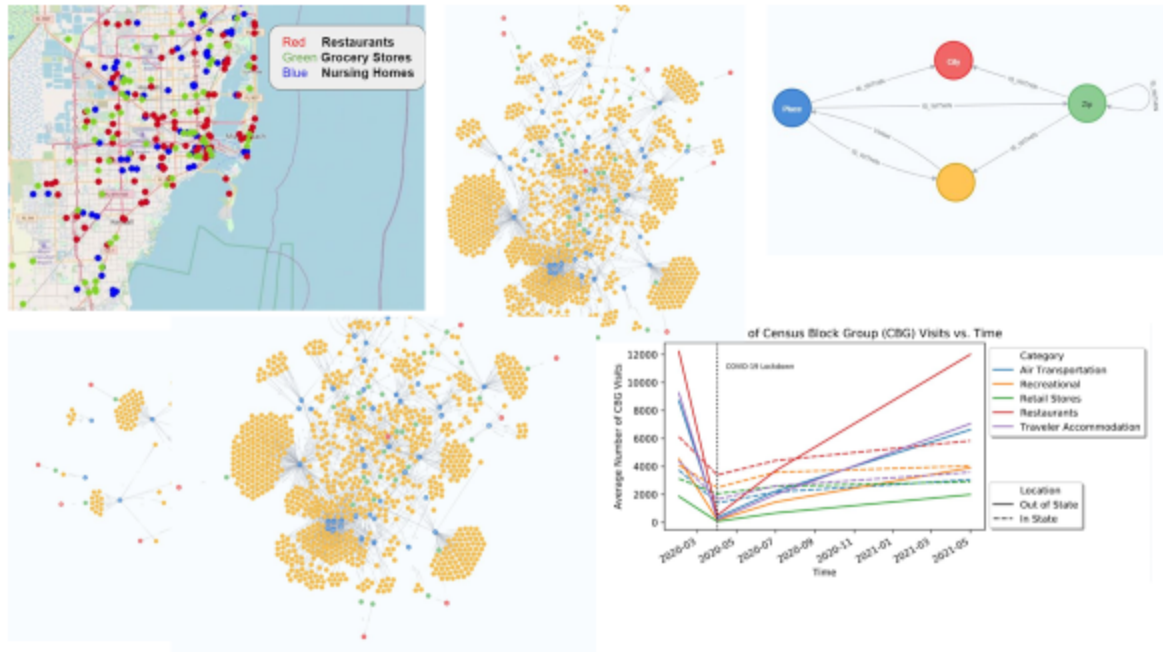
Github- https://github.com/unclekenroy/safegraph_neo4j

Poster- [Poster Bedeau](#)

SafeGraph to Neo4j Pipeline

This repository contains a pipeline for SafeGraph COVID-19 mobility data to Neo4j DB. It was written by Marcus Bedeau for his TACC 2021 REU project.





Below is the code I wrote to set up my environment.

Quick Start

1. Spin up a Neo4j instance:

```
```bash
```

```

docker run --rm -it \
 -p7474:7474 -p7687:7687 \
 -v $HOME/neo4j/data:/data \
 -v $HOME/neo4j/logs:/logs \
 -v $HOME/neo4j/import:/var/lib/neo4j/import \
 -v $HOME/neo4j/plugins:/plugins \
 --env NEO4J_AUTH=neo4j/test \
 neo4j:latest
...

```

## 2. Import SafeGraph data from CSV:

```

```bash
poetry install
poetry run python3 scripts/just_cbg.py -f data/safegraph/feb2020.csv
```

```

## ## Authors

- Marcus Bedeau (`unclekenroy`)
- Ethan Ho (`eho-tacc`)

## ## Neo4j on TACC

Ethan Ho 7/16/2021

```

- Managed to get neo4j Docker image running on TACC
- Can copy the `neo4j.conf` from Docker container:
```bash
docker run --rm -it neo4j:latest cat /var/lib/neo4j/conf/neo4j.conf > neo4j.conf
...

- Then rsync it to the `neo4j/conf` dir and invoke in detached container:
```bash
$ ls neo4j
conf data import logs plugins
$ singularity pull neo4j_latest.sif docker://neo4j:latest
$ nohup singularity exec \
 -B neo4j/data:/data \
 -B neo4j/logs:/logs \
 -B neo4j/import:/var/lib/neo4j/import \
 -B neo4j/conf:/var/lib/neo4j/conf \
 -B neo4j/plugins:/plugins \
 --env NEO4J_AUTH=neo4j/password \
 neo4j_latest.sif /docker-entrypoint.sh neo4j &
$ curl localhost:7474
{
 "bolt_routing" : "neo4j://localhost:7687",
 "transaction" : "http://localhost:7474/db/{databaseName}/tx",
 "bolt_direct" : "bolt://localhost:7687",
 "neo4j_version" : "4.3.2",
 "neo4j_edition" : "community"
}

```

```
}
...
```

Then we can connect via bolt using the Python API:

```
```python  
from neo4j import GraphDatabase  
driver = GraphDatabase.driver(uri="bolt://localhost:7687", auth=("neo4j", "password"))  
session = driver.session()  
session.run('MATCH (n) RETURN n LIMIT 25')  
# <neo4j.work.result.Result at 0x2ae5e1427d60>  
```
```

As a one-liner:

```
```bash  
$ python3 -c 'from neo4j import GraphDatabase; driver =  
GraphDatabase.driver(uri="bolt://localhost:7687", auth=("neo4j", "password")); session =  
driver.session(); print(session.run("MATCH (n) RETURN n LIMIT 25"))'  
<neo4j.work.result.Result object at 0x2ba4ffee8040>  
```
```

**Below is a script that are the ETL Pipeline within python which can be found in github.**

```
#!/usr/bin/env python3
```

```
import os
import time
import argparse
import pandas as pd #importing neccisary libraries
import numpy as np
import ast
import datetime
import calendar
from neo4j import GraphDatabase
from progress.bar import Bar
import logging

logging.basicConfig(level=logging.WARNING)
CSVS = ['feb2020.csv', 'april2020.csv', 'july2020.csv', 'june2021.csv']

def get_db_session(db_uri="bolt://localhost:7687", auth=("neo4j", "test")):
 """
 driver = GraphDatabase.driver(uri=db_uri, auth=auth)
 return driver.session()
 """

def process_safegraph_month(safegraph_csv_fp):
```

```

#####
session = get_db_session()
df = pd.read_csv(safegraph_csv_fp) # read csv to pandas

#isolate specified columns will be changing them to more meaningful names

Will add another part to have all the csv be read at once so I do not have to keep loading it.
#This for loop creates and loads all nodes and edges

breakpoint()
#len(df)
#this for loop I am createing all the places and cities and zips

with Bar(f'Processing {len(df)} rows', max=len(df)) as bar:
 for i, row in df.iterrows():

 # DEBUG: benchmark performance
 start = time.time()

 #setting variables that I will use to create the nodes.
 miamivv = row.visitor_home_cbgs
 pk= row.placekey
 name = row.location_name
 zipc= row.postal_code
 address= row.street_address
 lat= row.latitude
 long= row.longitude
 topCat= row.top_category
 subCat= row.sub_category
 city= row.city
 if((str(row.poi_cbg) == 'nan') == False):
 poi_cbg = str(row.poi_cbg)
 poiST= str(poi_cbg[:2])
 poiCTY= str(poi_cbg[2:5])
 check = 0 # This is telling the program that this row has the information we need

 else:
 check = 1 # if not we skip over this row

 if (check == 0):
 #These are the queries that I will pass to neo4j that will create all the place, city and
 zip nodes
 q1='MERGE(p:Place{name:'+""'+name+""'+', city:'+""'+city+""'+', zipcode:'+""'+
+str(zipc)+""'+', placeKey:'+""'+pk+""'+', address:'+""'+address+""'+',
topCatagory:'+""'+str(topCat)+""'+', subCatagory:'+""'+str(subCat)+""'+', cbg: '+""'+str(poi_cbg) +""'+
+', location:point({x:toFloat('+str(lat)+'),y:toFloat('+str(long)+'))})'
 q2='MERGE(z:Zip{name: '+""'+str(zipc)+""'+', zipC:'+""'+str(zipc)+""'+', city:'+""'+
+city+""'+'})'

```

```

q3='MERGE(c:City{name: '+' +city+'+', city:'+' +city+'+'});'
q4='MATCH (p:Place), (z:Zip) WHERE p.name = '+' +name +'' + 'AND p.city = '+' +city +'' + 'AND z.city = '+' +city +'' + 'AND p.zipcode = '+' +str(zipc)+' + 'AND z.name = '+' +str(zipc)+' + ' MERGE(p)-[r:IS_WITHIN]->(z);'
q5='MATCH (c:City), (z:Zip) WHERE c.name = '+' +city +'' + 'AND z.city = '+' +city +'' + 'AND z.name = '+' +str(zipc)+' + ' MERGE(z)-[r:IS_WITHIN]->(c);'
q6='MERGE(g:CensusBG{cbg:'+' +str(poicbg)+' +', State:'+' +str(poiST)+' +', County:'+' +str(poiCTY)+' +'})'
q7='MATCH (p:Place), (g:CensusBG) WHERE p.placeKey = '+' +pk +'' + 'AND g.State = '+' +str(poiST)+' + 'AND g.cbg = '+' +str(poicbg)+' + ' MERGE(p)-[r:IS_WITHIN]->(g);'

```

#Match staments are the edges that connect the two nodes

#running the queries

```

session.run(q1)
session.run(q2)
session.run(q3)
session.run(q4)
session.run(q5)
session.run(q6)
session.run(q7)

```

logging.info(f'run queries: {(time.time() - start):.4f}')

# miamivv = df.visitor\_home\_cbgs

#json ->dictionary-> nparray

if isinstance(miamivv, float):

iti= 0

else:

test=ast.literal\_eval(miamivv)

test2 = list(test.items())

testnpp = np.array(test2)

iti = len(testnpp)

if iti != 0:

da=0

dasum=0

percentage = []

rawVists = row.raw\_visit\_counts

dates= row.date\_range\_start

rawVisitors = row.raw\_visitor\_counts

#parsing the date month and year

year=dates[:4]

month=dates[5:7]

monthName=calendar.month\_name[int(month)]

```

for x in range(iti):

 da = int(testnpp[x][1])
 dasum += da # sum
 percentage = []

 for j in range(len(testnpp)):
 #here I am createing Census block nodes and connecting them with the places
 they have been.
 odds = (float((testnpp[j][1])/dasum)
 percentage += [odds]
 p1= float((percentage[j] *100)
 evc= float(((p1/100) * rawVists))
 NoCBGVisitors = float(100*((rawVisitors-dasum)/rawVisitors))
 node= testnpp[j][0] # this is the keys

 if(node[0].isupper() == True):
 nodee =(node)
 state = node[:2]
 county = node[3:]

 else:
 nodee = float(node)
 prop= testnpp[j][1] # this is the values
 state = node[:2]
 county = node[2:5]

 qq1= 'MERGE(g:CensusBG{cbg:'+"" +str(nodee)+""+ ' , State:'+""+state+""+' ,
County:'+""+county+""+' })'
 qq2 ='MATCH (p:Place), (g:CensusBG) WHERE p.placeKey = '+"" +pk +"" + '
AND g.cbg = '+"" +str(nodee)+"" + ' MERGE(g)-[b:Visited{ PercentofVisits:'+str(p1)+',
EstimatedVisits:'+str(evc)+' , NoCbgPercentage:'+str(NoCBGVisitors)+' ,
month:'+""+str(monthName)+""+' , year:'+str(year)+'}]->(p)'

 session.run(qq1)
 session.run(qq2)
 logging.info(f'until node/edge creation: {(time.time() - start):.4f}')
 logging.info(f'full iter: {(time.time() - start):.4f}')
 # assert 0
 bar.next()

def main(safegraph_csv_fp, n_threads, **kwargs):
 # data_dir = os.path.join('data', 'safegraph')
 # safegraph_csv_fp = os.path.join(data_dir, 'feb2020.csv')
 assert os.path.isfile(safegraph_csv_fp), f"file at {safegraph_csv_fp} does not exist"
 _ = process_safegraph_month(safegraph_csv_fp=safegraph_csv_fp)

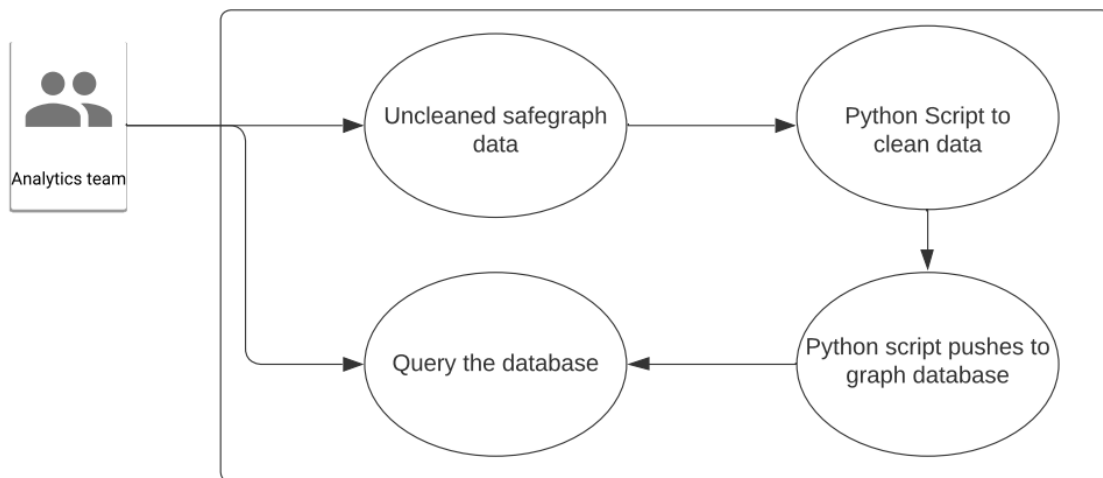
```

```
def get_opts() -> dict:
 """Get options from command line"""
 parser = argparse.ArgumentParser(description="")
 parser.add_argument("-v", "--verbose", action="store_true", required=False, help="")
 # parser.add_argument('file', type=argparse.FileType('r'), help="")
 parser.add_argument("-f", "--safegraph-csv-fp", type=str, required=True, help="path(s) to
safegraph CSV file")
 parser.add_argument("-n", "--n-threads", type=int, default=16, required=False, help="number
of threads to use")
 return vars(parser.parse_args())

if __name__ == '__main__':
 main(**get_opts())
```

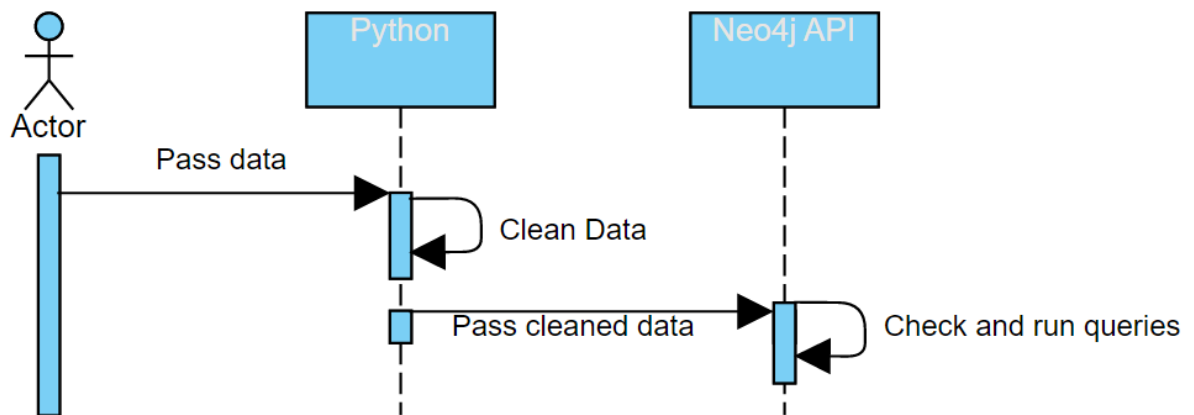
## Diagrams

### Use Case Diagram

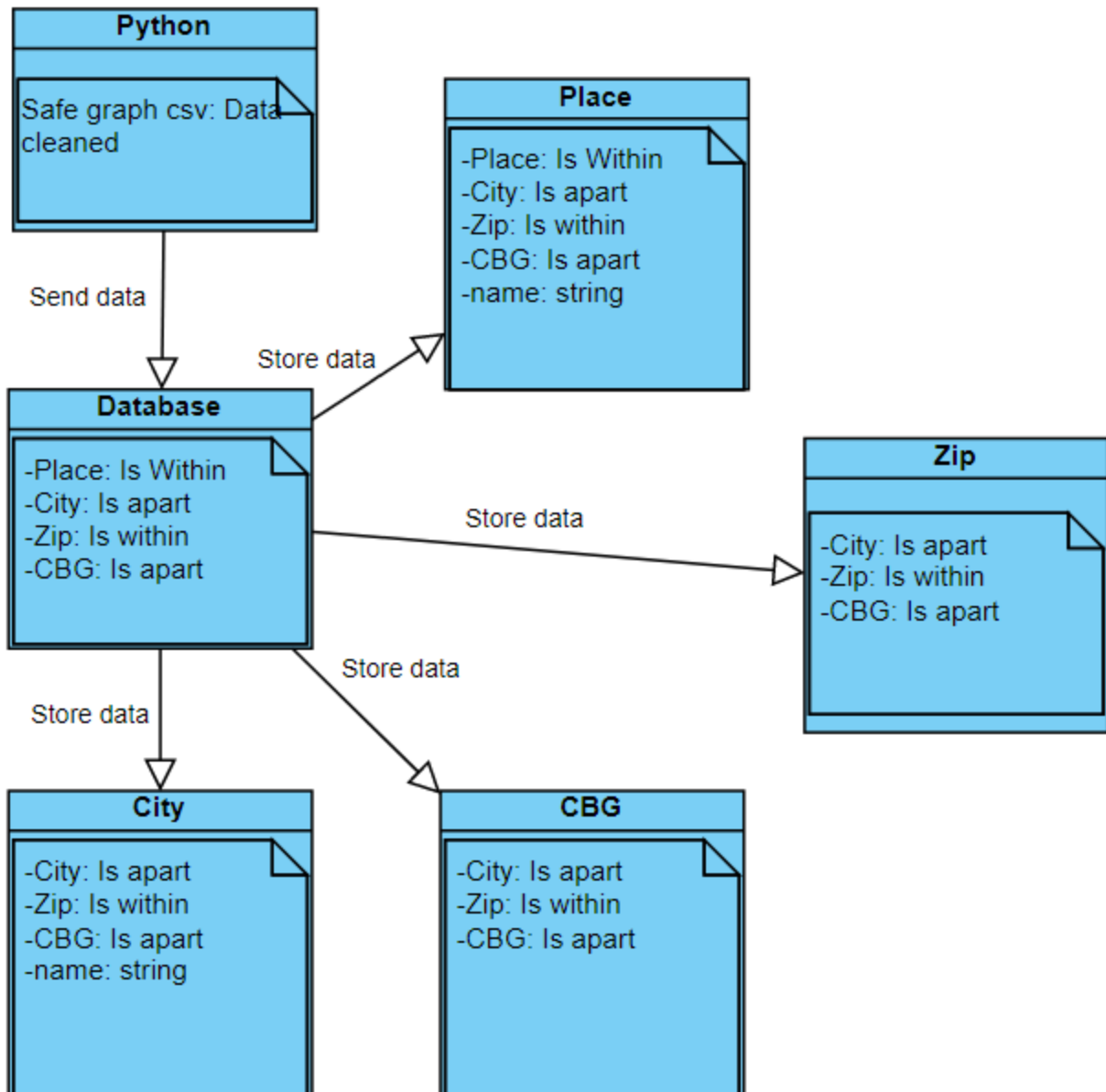


### Sequence Diagram





## Class Diagram



Jennifer Hierro

\*Code for documentation and diagrams.

## Jennifer Hierro – NuMom Spring 2022 coding contributions user stories #389 & #427.

| User story ID | Belongs to | Description                                                                          | Member(s)                                                                            |
|---------------|------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| #01           | Developers | Review Pull Requests from Previous Semester/Sprint (Each Sprint)                     | All team members                                                                     |
| #02           | Developers | Post a new Issue to the Github                                                       | All team members                                                                     |
| #371          | Developers | Task #1: Welcome Spring 2022!                                                        | Entire Team                                                                          |
| #389          | Developers | Task 2: Password Strength Checker                                                    | Entire Team                                                                          |
| #384          | Developers | Clinics page filter dropdown buttons are too small and hidden under the clinics list | Julian Lopez                                                                         |
| #387          | Developers | Fix the UI in the Settings Page                                                      | Fernando Perez Morilla                                                               |
| #404          | Developers | Checklist Button images look bad when pressed                                        | Julian Lopez                                                                         |
| #383          | Developers | Update the Welcome banner to make the Homepage more friendly                         | Julian Lopez                                                                         |
| #390          | Developers | Fix Welcome sign                                                                     | Amy Garcia Fernandez, Anael Ais                                                      |
| #419          | Developers | Fix Checklist Buttons: Pressing a button that is checked does not remove check mark  | Julian Lopez                                                                         |
| #395          | Developers | Remove Back Button on Sign Up Page                                                   | Fernando Perez Morilla                                                               |
| #425          | Developers | Translation JSON files are incomplete and untranslated                               | Julian Lopez                                                                         |
| #409          | Developers | Learn Tab Formatting                                                                 | Fernando Perez Morilla                                                               |
| #410          | Developers | ImmunizationSchedule vaccine check off                                               | Julian Lopez                                                                         |
| #382          | Developers | Visual Changes to Do's and Dont's Section                                            | Gabriel Gomez                                                                        |
| #412          | Developers | Add a "show password" button to the Login and Sign up screens                        | Amy Garcia Fernandez                                                                 |
| #428          | Developers | Avoid showing home button on Learn / Resources                                       | Fernando Perez Morilla                                                               |
| #422          | Developers | STD List Inconsistency                                                               | Oscar Sanchez                                                                        |
| #427          | Developers | Grey Random Space in Locations                                                       | Jennifer Hierro                                                                      |
| #1000         | Developers | Finish the Final Deliverable and Prepare for Showcase                                | Anael Ais, Amy Garcia Fernandez, Julian Lopez, Fernando Perez Morilla, Oscar Sanchez |
| #1001         | Developers | Continue working on an unfinished user story from last sprint                        | Anael Ais, Eric Campillo, Kimberly Metelus                                           |

Proof of #389 :

```

1 100 src/Components/SignUpPassword.jsx
2 16 export default SignUpPassword = (props) => {
3 17 const {password, setPassword} = useState('');
4 18 const {repeat, setRepeat} = useState('');
5 19 const {passwordStrength, setPasswordStrength} = useState('');
6 20 const {charCount, setCount} = useState('');
7 21 const {carryLowerCase, setCarryLower} = useState(false);
8 22 const {carryUpper, setCarryUpper} = useState(false);
9 23 const {number, setNum} = useState(false);
10 24 const {carrySymbol, setCarrySymbol} = useState(false);
11 25 const {liveInfants} = props.route.params;
12 26 const {name} = props.route.params;
13 27 const {dob} = props.route.params;
14 28 @ -34,14 -48,114 @@ export default SignUpPassword = (props) => {
15 29 }, [liveInfants]);
16 30 }, [liveInfants]);
17 31 }, [liveInfants]);
18 32 }, [liveInfants]);
19 33 }, [liveInfants]);
20 34 }, [liveInfants]);
21 35 }, [liveInfants]);
22 36 }, [liveInfants]);
23 37 }, [liveInfants]);
24 38 }, [liveInfants]);
25 39 }, [liveInfants]);
26 40 }, [liveInfants]);
27 41 }, [liveInfants]);
28 42 }, [liveInfants]);
29 43 }, [liveInfants]);
30 44 }, [liveInfants]);
31 45 }, [liveInfants]);
32 46 }, [liveInfants]);
33 47 }, [liveInfants]);
34 48 }, [liveInfants]);
35 49 }, [liveInfants]);
36 50 }, [liveInfants]);
37 51 }, [liveInfants]);
38 52 }, [liveInfants]);
39 53 }, [liveInfants]);
40 54 }, [liveInfants]);
41 55 }, [liveInfants]);
42 56 }, [liveInfants]);
43 57 }, [liveInfants]);
44 58 }, [liveInfants]);
45 59 }, [liveInfants]);
46 60 }, [liveInfants]);
47 61 }, [liveInfants]);
48 62 }, [liveInfants]);
49 63 }, [liveInfants]);
50 64 }, [liveInfants]);
51 65 }, [liveInfants]);
52 66 }, [liveInfants]);
53 67 }, [liveInfants]);
54 68 }, [liveInfants]);
55 69 }, [liveInfants]);
56 70 }, [liveInfants]);
57 71 }, [liveInfants]);
58 72 }, [liveInfants]);
59 73 }, [liveInfants]);
60 74 }, [liveInfants]);
61 75 }, [liveInfants]);
62 76 }, [liveInfants]);
63 77 }, [liveInfants]);
64 78 }, [liveInfants]);
65 79 }, [liveInfants]);
66 80 }, [liveInfants]);
67 81 }, [liveInfants]);
68 82 }, [liveInfants]);
69 83 }, [liveInfants]);
70 84 }, [liveInfants]);
71 85 }, [liveInfants]);
72 86 }, [liveInfants]);
73 87 }, [liveInfants]);
74 88 }, [liveInfants]);
75 89 }, [liveInfants]);
76 90 }, [liveInfants]);
77 91 }, [liveInfants]);
78 92 }, [liveInfants]);
79 93 }, [liveInfants]);
80 94 }, [liveInfants]);
81 95 }, [liveInfants]);
82 96 }, [liveInfants]);
83 97 }, [liveInfants]);
84 98 }, [liveInfants]);
85 99 }, [liveInfants]);
86 100 }, [liveInfants]);
87 101 }, [liveInfants]);
88 102 }, [liveInfants]);
89 103 }, [liveInfants]);
90 104 }, [liveInfants]);
91 105 }, [liveInfants]);
92 106 }, [liveInfants]);
93 107 }, [liveInfants]);
94 108 }, [liveInfants]);
95 109 }, [liveInfants]);
96 110 }, [liveInfants]);
97 111 }, [liveInfants]);
98 112 }, [liveInfants]);
99 113 }, [liveInfants]);
100 114 }, [liveInfants]);
101 115 }, [liveInfants]);
102 116 }, [liveInfants]);
103 117 }, [liveInfants]);
104 118 }, [liveInfants]);
105 119 }, [liveInfants]);
106 120 }, [liveInfants]);
107 121 }, [liveInfants]);
108 122 }, [liveInfants]);
109 123 }, [liveInfants]);
110 124 }, [liveInfants]);
111 125 }, [liveInfants]);
112 126 }, [liveInfants]);
113 127 }, [liveInfants]);
114 128 }, [liveInfants]);
115 129 }, [liveInfants]);
116 130 }, [liveInfants]);
117 131 }, [liveInfants]);
118 132 }, [liveInfants]);
119 133 }, [liveInfants]);
120 134 }, [liveInfants]);
121 135 }, [liveInfants]);
122 136 }, [liveInfants]);
123 137 }, [liveInfants]);
124 138 }, [liveInfants]);
125 139 }, [liveInfants]);
126 140 }, [liveInfants]);
127 141 }, [liveInfants]);
128 142 }, [liveInfants]);
129 143 }, [liveInfants]);
130 144 }, [liveInfants]);
131 145 }, [liveInfants]);
132 146 }, [liveInfants]);
133 147 }, [liveInfants]);
134 148 }, [liveInfants]);
135 149 }, [liveInfants]);
136 150 }, [liveInfants]);
137 151 }, [liveInfants]);
138 152 }, [liveInfants]);
139 153 }, [liveInfants]);
140 154 }, [liveInfants]);
141 155 }, [liveInfants]);
142 156 }, [liveInfants]);
143 157 }, [liveInfants]);
144 158 }, [liveInfants]);
145 159 }, [liveInfants]);
146 160 }, [liveInfants]);
147 161 }, [liveInfants]);
148 162 }, [liveInfants]);
149 163 }, [liveInfants]);
150 164 }, [liveInfants]);
151 165 }, [liveInfants]);
152 166 }, [liveInfants]);
153 167 }, [liveInfants]);
154 168 }, [liveInfants]);
155 169 }, [liveInfants]);
156 170 }, [liveInfants]);
157 171 }, [liveInfants]);
158 172 }, [liveInfants]);
159 173 }, [liveInfants]);
160 174 }, [liveInfants]);
161 175 }, [liveInfants]);
162 176 }, [liveInfants]);
163 177 }, [liveInfants]);
164 178 }, [liveInfants]);
165 179 }, [liveInfants]);
166 180 }, [liveInfants]);
167 181 }, [liveInfants]);
168 182 }, [liveInfants]);
169 183 }, [liveInfants]);
170 184 }, [liveInfants]);
171 185 }, [liveInfants]);
172 186 }, [liveInfants]);
173 187 }, [liveInfants]);
174 188 }, [liveInfants]);
175 189 }, [liveInfants]);
176 190 }, [liveInfants]);
177 191 }, [liveInfants]);
178 192 }, [liveInfants]);
179 193 }, [liveInfants]);
180 194 }, [liveInfants]);
181 195 }, [liveInfants]);
182 196 }, [liveInfants]);
183 197 }, [liveInfants]);
184 198 }, [liveInfants]);
185 199 }, [liveInfants]);
186 200 }, [liveInfants]);
187 201 }, [liveInfants]);
188 202 }, [liveInfants]);
189 203 }, [liveInfants]);
190 204 }, [liveInfants]);
191 205 }, [liveInfants]);
192 206 }, [liveInfants]);
193 207 }, [liveInfants]);
194 208 }, [liveInfants]);
195 209 }, [liveInfants]);
196 210 }, [liveInfants]);
197 211 }, [liveInfants]);
198 212 }, [liveInfants]);
199 213 }, [liveInfants]);
200 214 }, [liveInfants]);
201 215 }, [liveInfants]);
202 216 }, [liveInfants]);
203 217 }, [liveInfants]);
204 218 }, [liveInfants]);
205 219 }, [liveInfants]);
206 220 }, [liveInfants]);
207 221 }, [liveInfants]);
208 222 }, [liveInfants]);
209 223 }, [liveInfants]);
210 224 }, [liveInfants]);
211 225 }, [liveInfants]);
212 226 }, [liveInfants]);
213 227 }, [liveInfants]);
214 228 }, [liveInfants]);
215 229 }, [liveInfants]);
216 230 }, [liveInfants]);
217 231 }, [liveInfants]);
218 232 }, [liveInfants]);
219 233 }, [liveInfants]);
220 234 }, [liveInfants]);
221 235 }, [liveInfants]);
222 236 }, [liveInfants]);
223 237 }, [liveInfants]);
224 238 }, [liveInfants]);
225 239 }, [liveInfants]);
226 240 }, [liveInfants]);
227 241 }, [liveInfants]);
228 242 }, [liveInfants]);
229 243 }, [liveInfants]);
230 244 }, [liveInfants]);
231 245 }, [liveInfants]);
232 246 }, [liveInfants]);
233 247 }, [liveInfants]);
234 248 }, [liveInfants]);
235 249 }, [liveInfants]);
236 250 }, [liveInfants]);
237 251 }, [liveInfants]);
238 252 }, [liveInfants]);
239 253 }, [liveInfants]);
240 254 }, [liveInfants]);
241 255 }, [liveInfants]);
242 256 }, [liveInfants]);
243 257 }, [liveInfants]);
244 258 }, [liveInfants]);
245 259 }, [liveInfants]);
246 260 }, [liveInfants]);
247 261 }, [liveInfants]);
248 262 }, [liveInfants]);
249 263 }, [liveInfants]);
250 264 }, [liveInfants]);
251 265 }, [liveInfants]);
252 266 }, [liveInfants]);
253 267 }, [liveInfants]);
254 268 }, [liveInfants]);
255 269 }, [liveInfants]);
256 270 }, [liveInfants]);
257 271 }, [liveInfants]);
258 272 }, [liveInfants]);
259 273 }, [liveInfants]);
260 274 }, [liveInfants]);
261 275 }, [liveInfants]);
262 276 }, [liveInfants]);
263 277 }, [liveInfants]);
264 278 }, [liveInfants]);
265 279 }, [liveInfants]);
266 280 }, [liveInfants]);
267 281 }, [liveInfants]);
268 282 }, [liveInfants]);
269 283 }, [liveInfants]);
270 284 }, [liveInfants]);
271 285 }, [liveInfants]);
272 286 }, [liveInfants]);
273 287 }, [liveInfants]);
274 288 }, [liveInfants]);
275 289 }, [liveInfants]);
276 290 }, [liveInfants]);
277 291 }, [liveInfants]);
278 292 }, [liveInfants]);
279 293 }, [liveInfants]);
280 294 }, [liveInfants
```

**Code for #389:**

```
import React, {useEffect, useState} from 'react';
import {
 AsyncStorage,
 Keyboard,
 Text,
 TextInput as TextBox,
 TouchableOpacity,
 View,
 TouchableHighlight,
 KeyboardAvoidingView,
} from 'react-native';
import appStyles from './AppStyles';
import Button from './Button';
import translate from './getLocalizedText';
export default SignUpPassword = (props) => {
 const [password, setPassword] = useState("");
 const [repeat, setRepeat] = useState("");
 const [passwordStrength, setPassStrength] = useState("");
 const [charCount, setCount] = useState("");
 const [carryLowerCase, setCarryLower] = useState(false);
 const [carryUpper, setCarryUpper] = useState(false);
 const [number, setNum] = useState(false);
 const [carrySymbol, setCarrySymbol] = useState(false);
 const {liveMiami} = props.route.params;
 const {name} = props.route.params;
 const {dob} = props.route.params;
 @@ -34,14 +40,114 @@ export default SignUpPassword = (props) => {
 })
 .done();
 }, []);
// if user makes changes to password, save password, check password strength
let changePassword = (password) => {
 setPassword(password);
 checkPassStrength(password);
};

// checking the user's password
let checkPassStrength = (password) => {

 //initialize variables
 setCarryLower(false);
 setCarryUpper(false);
 setCarrySymbol(false);
```

```

setNum(false);

//variable depicting the number of characters in each
let charCount = 0;

//check if the password contains a lowercase letter then increase charCount
if (/[a-z]/.test(password)) {

 charCount++;
 setCarryLower(true);
}
//check if the password contains an uppercase letter then increase charCount
if (/[A-Z]/.test(password)) {

 charCount++;
 setCarryUpper(true);
}
//check if the password contains a number then increase charCount
if (/[0-9]/.test(password)) {

 charCount++;
 setNum(true);
}
//check if the password contains a symbol in it then increase charcount
if (/[^A-Za-z0-9]/.test(password)) {

 charCount++;
 setCarrySymbol(true);
}

setCount(charCount);

// if the password length is <= 4 and is in the list of badpasswords
if (password.length <= 4 && badPasswords.includes(password)) {

 setPassStrength(0); // bad password
}

// if the password length is >= 5 and has three of the four types of groups
else if (password.length >= 5 && charCount == 3) {

 setPassStrength(1); // medium password
}

```

```

// if the password length is >= 5 and has all four types of groups then strong password
else if (password.length >= 5 && charCount == 4) {
setPassStrength(2); // strong password
}
else {

 setPassStrength(0); // for any other variations of the password
}
};

```

```

let passwordStyle = (passwordStrength) => {
 let color;
 let message = "";

 if (passwordStrength == 0) {

 color = appStyles.pinkColor;
 message = 'Password Strength: Poor';

 }
 else if (passwordStrength == 1) {

 color = appStyles.blueColor;
 message = 'Password Strength: Medium';

 }
 else {
 color = '#298000';
 message = 'Password Strength: High';
 }
 return (
 <View>
 <Text style={{color: colors, textAlign: 'center'}}>{message}</Text>

 </View>
);
};

```

```

let onPress = () => {
 if (password !== repeat) {

```

```

 alert(translate('passwordMismatch'));
 } else if (!password || !repeat) {
 alert(translate('fillOutAllFields'));
 } else if (password.length < 6) {
 } else if (password.length <= 4) {
 alert(translate('passwordTooShort'));
 } else if (passwordStrength == 0) {
 alert(translate('passwordWeak'))
 } else {
 // props.setUserInfo({password});
 // AsyncStorage.setItem('pass', password);
 // AsyncStorage.setItem('repeat', repeat);
 props.navigation.navigate('SignUpYesorNoPregnant', {
 liveMiami,
 name,
 dob,
 email,
 phone,
 password,
 question: translate('areYouPregnant'),
 value: 'pregnant',
 });
 }
};
return (
 <KeyboardAvoidingView
 behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
 style={appStyles.signupContainer}
 enabled={false}
 >
 <TouchableHighlight
 onPress={Keyboard.dismiss}
 underlayColor="transparent"
 accessible={false}
 >
 <>
 <View style={appStyles.container}>
 <View
 style={{
 paddingTop: appStyles.win.height * 0.15,
 justifyContent: 'center',
 alignItems: 'center',
 position: 'absolute',
 }}
 >

```



```

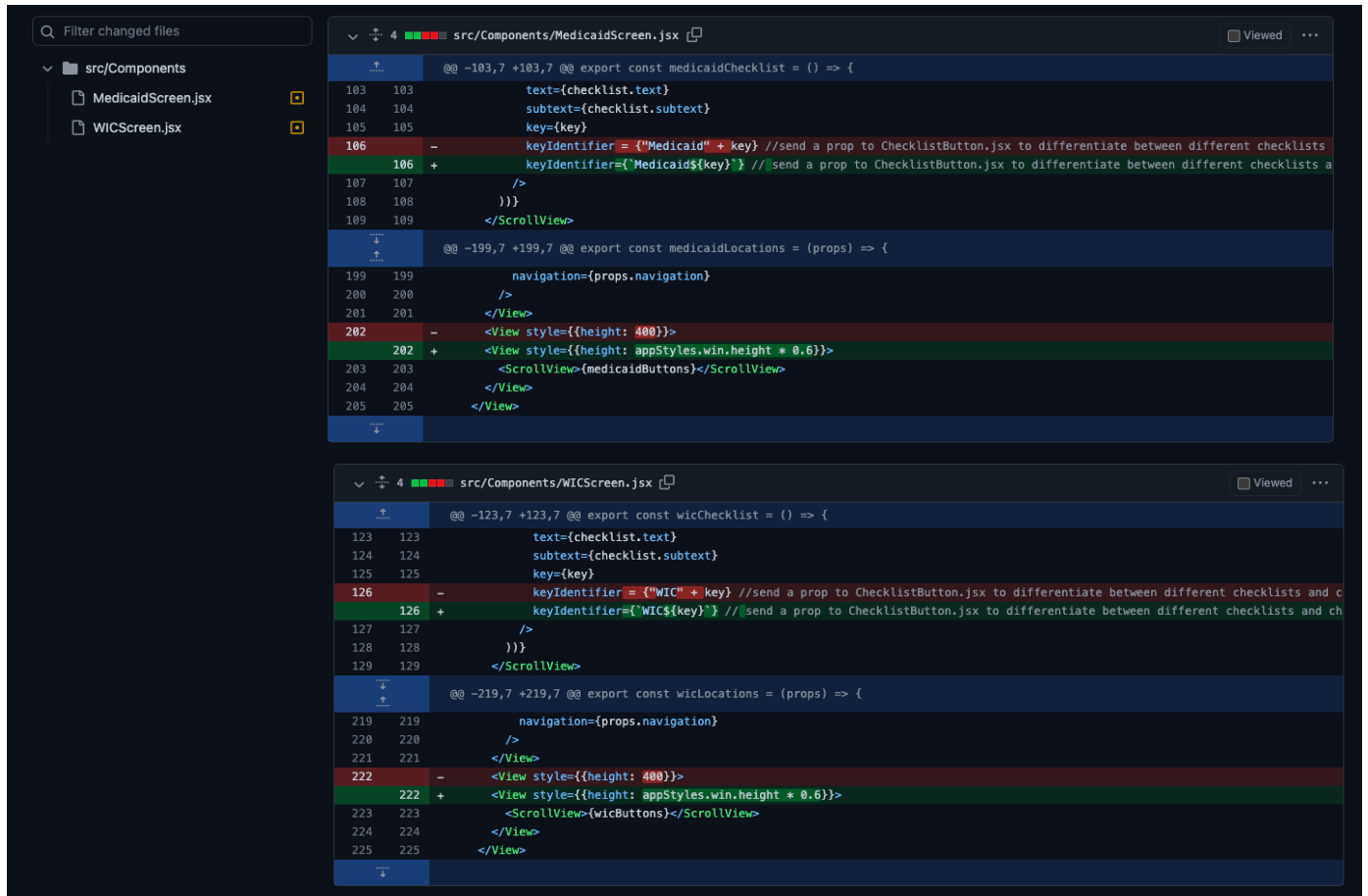
>
<Text style={appStyles.titleBlue}>
 {translate('createPassword')}
</Text>
<View style={{paddingTop: appStyles.win.height * 0.05}}>
 <TextBox
 placeholder={translate('passwordInput')}
 onChangeText={setPassword}
 secureTextEntry
 value={password}
 style={appStyles.TextInputMask}
 />
 <TextBox
 placeholder={translate('repeatPasswordInput')}
 onChangeText={setRepeat}
 secureTextEntry
 value={repeat}
 style={appStyles.TextInputMask}
 />
</View>
</View>
</View>
<View
 style={{
 width: '100%',
 alignItems: 'center',
 margin: '15%',
 }}
>
 <Button
 style={appStyles.button}
 text={translate('continueButton')}
 onPress={onPress}
 />
</View>
</>
</TouchableHighlight>
</KeyboardAvoidingView>
);
};

```

User story #427 – Grey random spaces in Location

In file - src/Components/WICScreen.jsx:

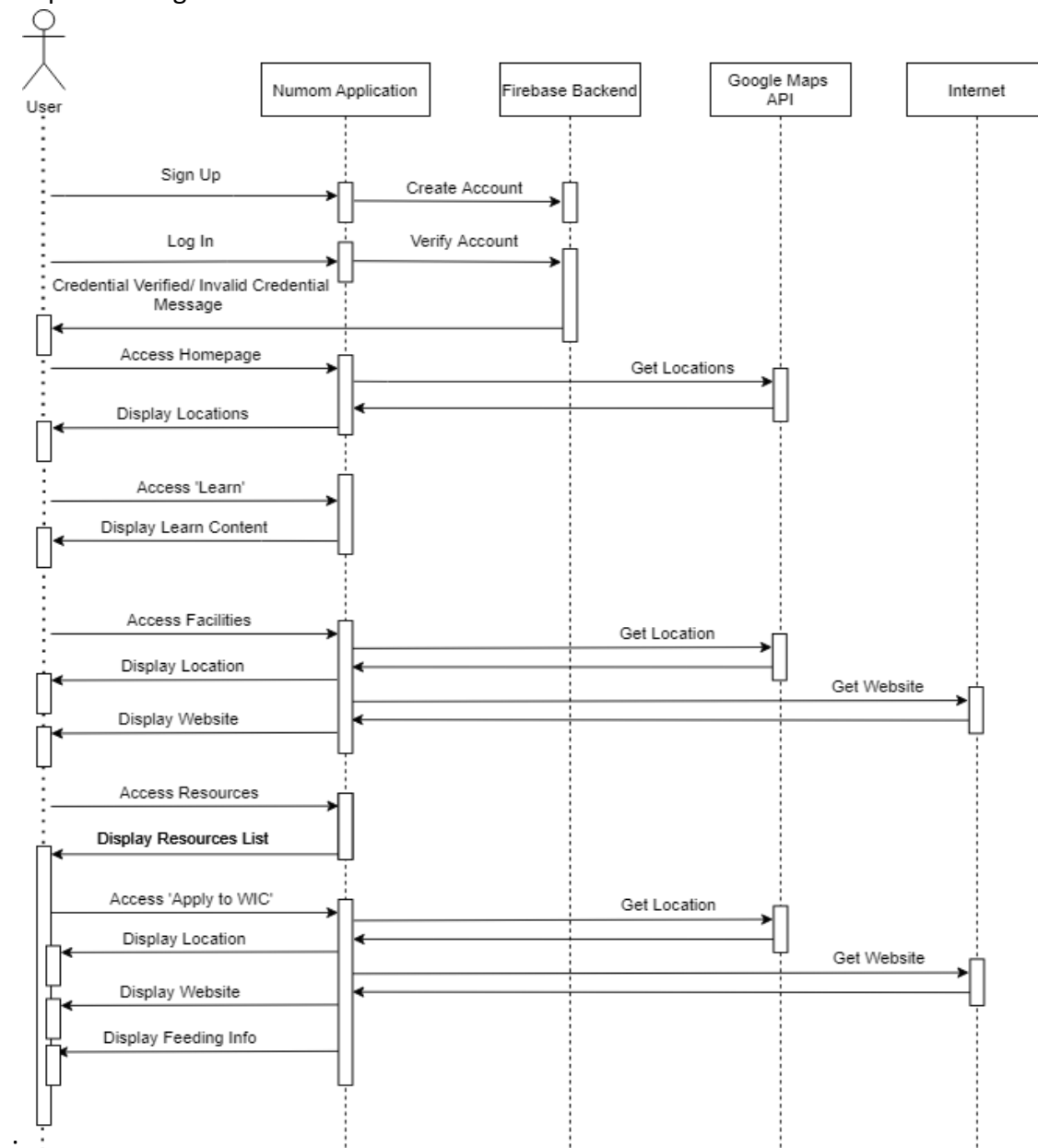
Changes added:



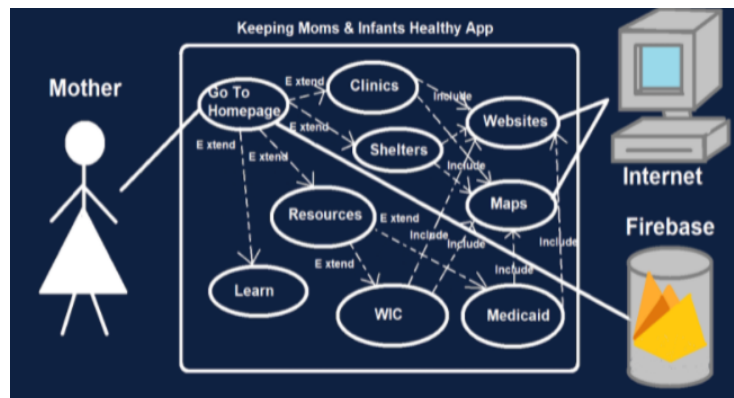
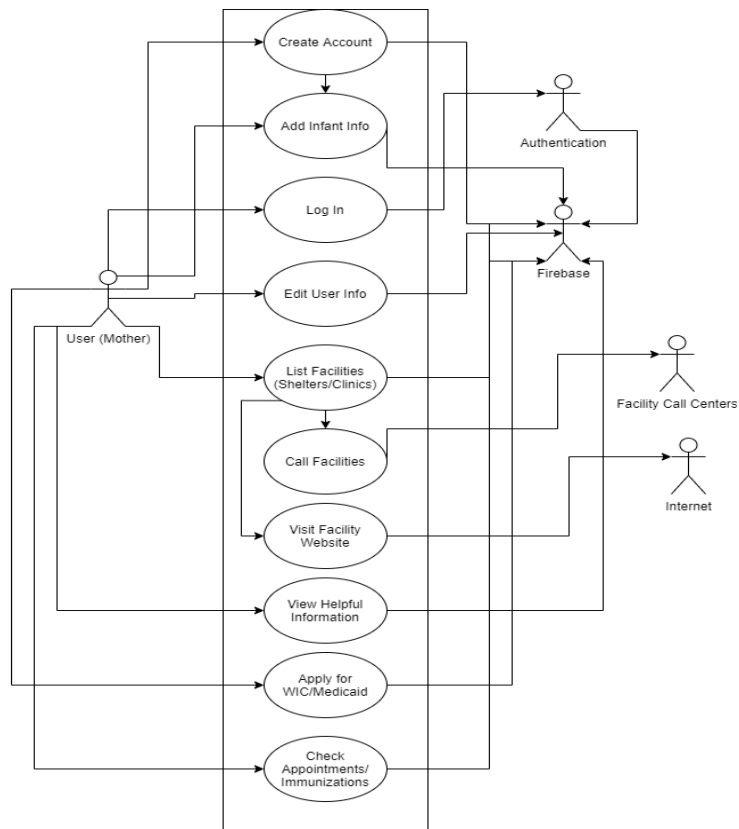
```
src/Components/MedicaidScreen.jsx
@@ -103,7 +103,7 @@ export const medicaidChecklist = () => {
 103 103 text={checklist.text}
 104 104 subtext={checklist.subtext}
 105 105 key={key}
 106 - keyIdentifier = {"Medicaid" + key} //send a prop to ChecklistButton.jsx to differentiate between different checklists
 106 + keyIdentifier={"Medicaid${key}"} // Send a prop to ChecklistButton.jsx to differentiate between different checklists a
 107 107 })
 108 108 })
 109 109 </ScrollView>
@@ -199,7 +199,7 @@ export const medicaidLocations = (props) => {
 199 199 navigation={props.navigation}
 200 200 </>
 201 201 </View>
 202 - <View style={{height: 400}}>
 202 + <View style={{height: appStyles.win.height * 0.6}}>
 203 203 <ScrollView>{medicaidButtons}</ScrollView>
 204 204 </View>
 205 205 </View>

src/Components/WICScreen.jsx
@@ -123,7 +123,7 @@ export const wicChecklist = () => {
 123 123 text={checklist.text}
 124 124 subtext={checklist.subtext}
 125 125 key={key}
 126 - keyIdentifier = {"WIC" + key} //send a prop to ChecklistButton.jsx to differentiate between different checklists and c
 126 + keyIdentifier={"WIC${key}"} // Send a prop to ChecklistButton.jsx to differentiate between different checklists and ch
 127 127 })
 128 128 })
 129 129 </ScrollView>
@@ -219,7 +219,7 @@ export const wicLocations = (props) => {
 219 219 navigation={props.navigation}
 220 220 </>
 221 221 </View>
 222 - <View style={{height: 400}}>
 222 + <View style={{height: appStyles.win.height * 0.6}}>
 223 223 <ScrollView>{wicButtons}</ScrollView>
 224 224 </View>
 225 225 </View>
```

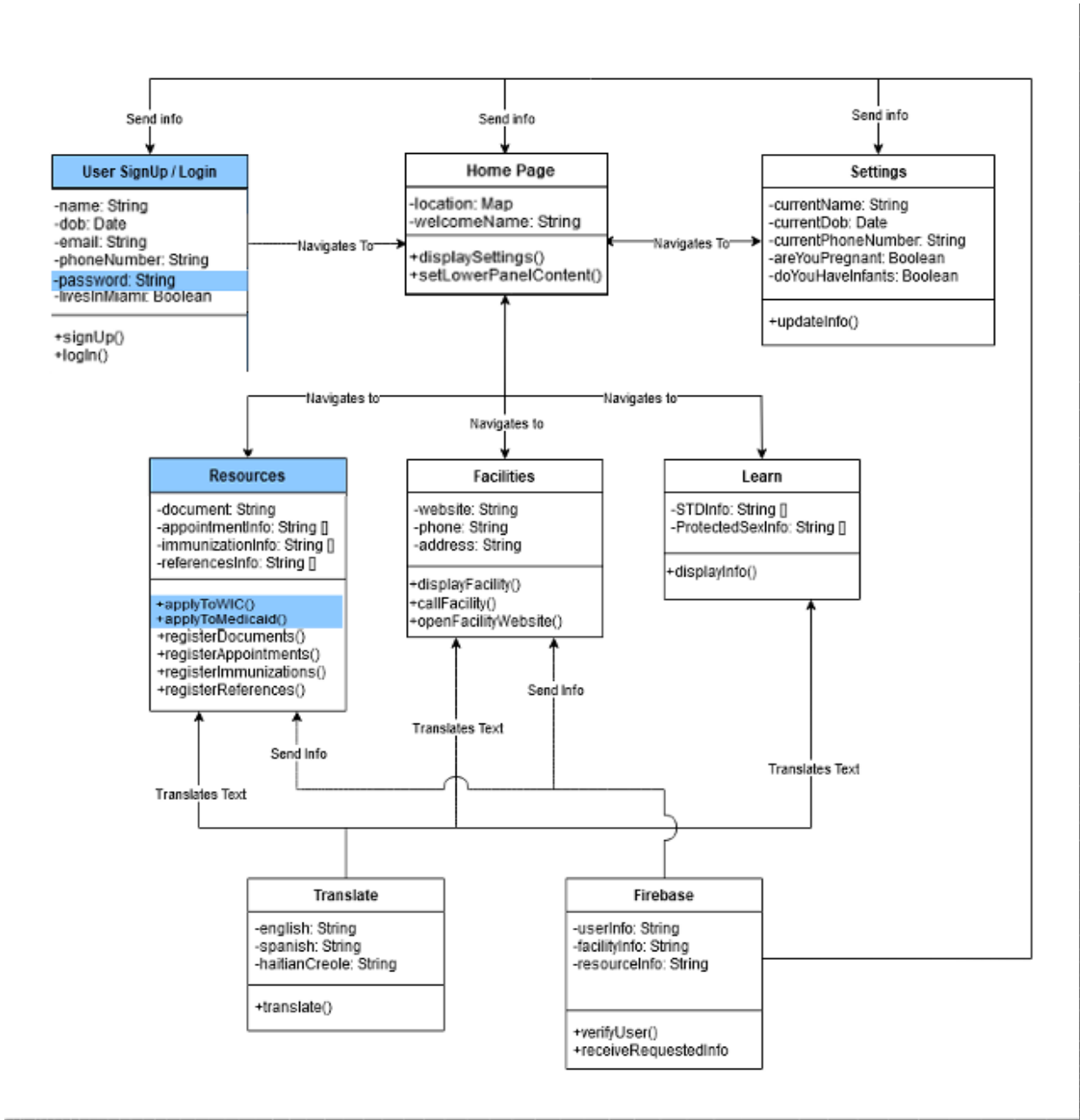
## Sequence Diagram:



Use Case Diagram:



Class Diagram – Contributions highlighted



Maria Perez

## Documentation (Code and Diagrams)

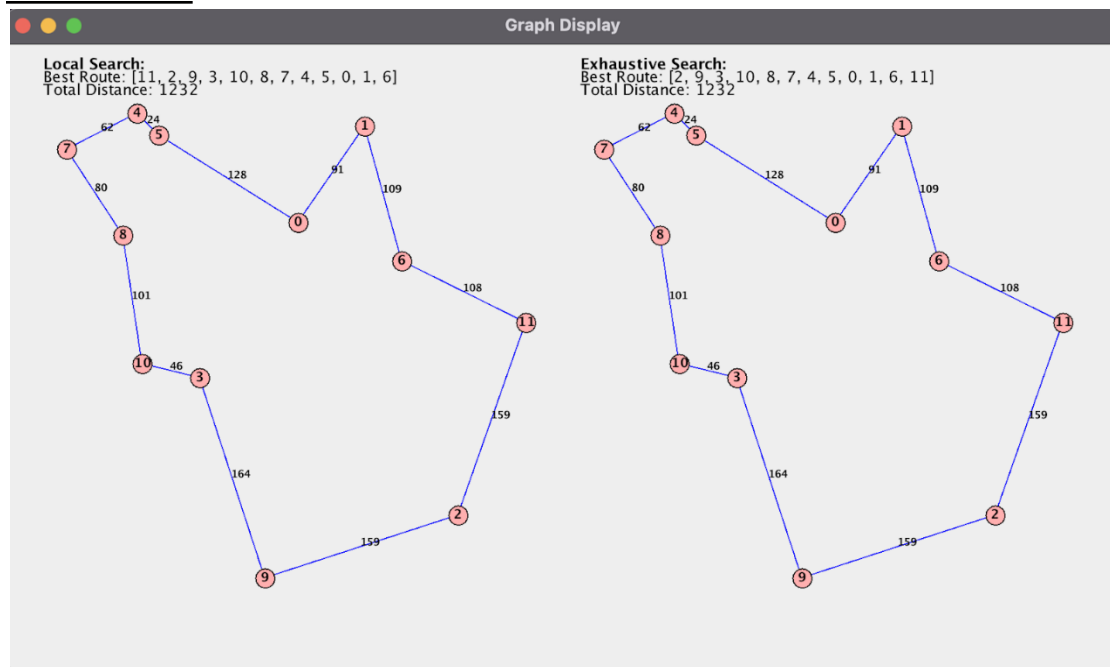
Maria Perez

### Algorithm Techniques Final Project

#### Description:

This project was inspired by the traveling salesman problem, in which one is given a set of points, or cities, and has to find the shortest path that visits all of the points exactly once while making sure that the starting and ending point are the same. In the screenshots shown on the next page, on the images to the right, the input is being displayed in a txt file. Part of the input consists of the number of cities, or points, in the graph, which is represented by the number in the first row of the input file. In both of the examples below the number of points is 12. The rest of the numbers that follow in the next rows of the input file are x and y coordinates that represent the positions of the points. The goal was to calculate the shortest path through all the points with two different algorithmic methods: local search and exhaustive search, and then display those results in a clear visual manner, showing how the two methods contrast each other. Overall, the different graph outputs for each respective method (local and exhaustive) are represented visually as can be seen in the images that follow. The graphical representations display the shortest path for each method along with the pairwise distances between each point. The best routes and total distances are also displayed numerically at the top of the screen under each respective method title (local search and exhaustive search).

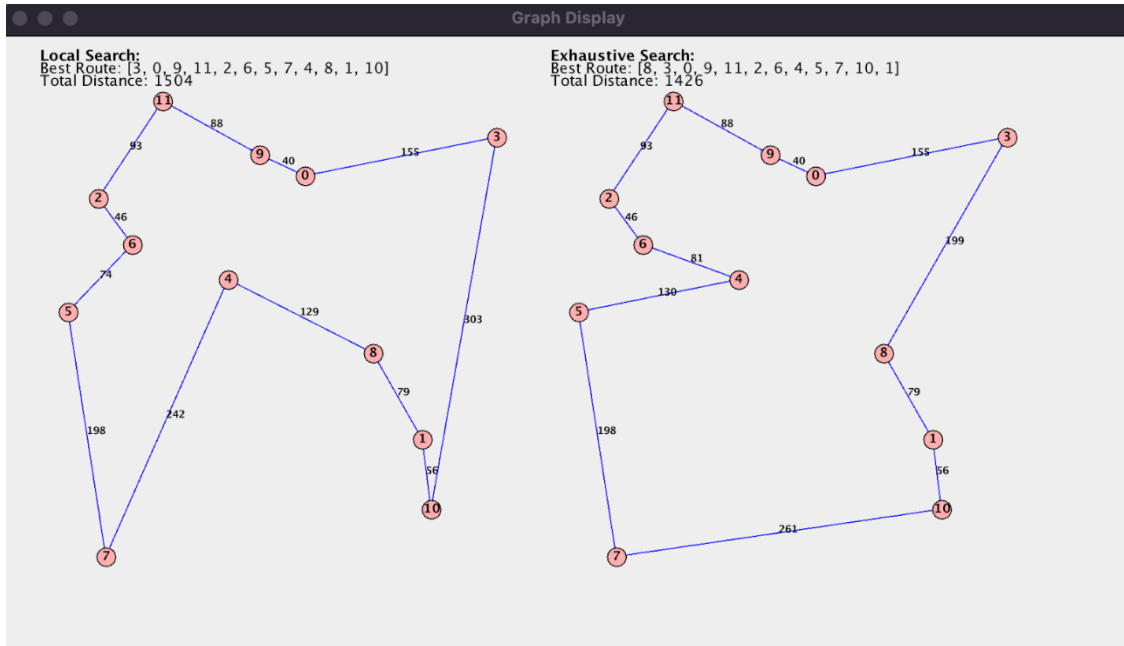
## Screenshots:



**Figure 1:** A display of two graphs side-by-side that have the same points in common. The graph on the left shows the shortest path between all of the points, starting and ending on the same point, using the local search algorithmic method. The graph on the right shows the shortest route between all the points using the exhaustive search algorithmic method. Each graph has weights to help calculate the shortest distance and the best route overall. These weights are the pairwise distances between each point and they are shown above the blue lines connecting the points together. These blue lines are a visual depiction of the route that touches every single point while achieving the shortest total distance for that specific algorithmic method (local and exhaustive search, respectively).

```
graph.txt
12
222 132
274 57
347 360
145 253
96 47
113 64
303 162
41 75
85 142
196 409
100 242
400 210
```

**Figure 2:** This image shows the input that generated the two graphs above. This txt file contains in the first row the number of vertices, or points, that should be displayed for both graphs above. In this case that number is twelve. The following numbers in the next rows are the x and y coordinates for these points, with each x and y separated by a space and each new line representing a new set of coordinates.



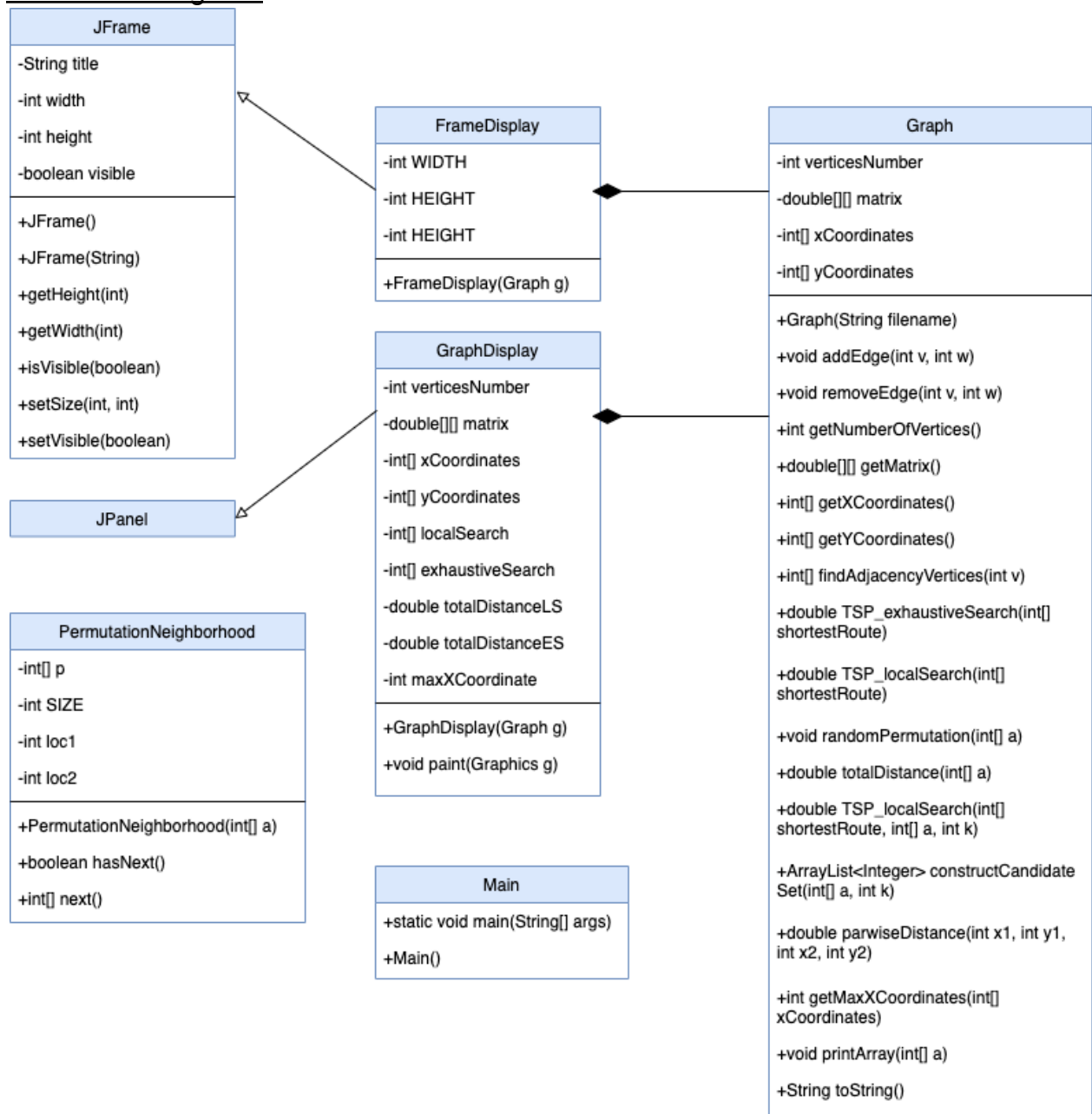
**Figure 3:** Much like the first image above, this screenshot also depicts two graphs with the same points as well as the two different approaches to the result discussed earlier: local search and exhaustive search. The graph on the left, unlike that of Figure 1, is noticeably different from the one on the right. They don't match up and that is because the local search method on the left generates a different path than the exhaustive search method on the right. The exhaustive search method is more efficient here because, as it says in the top under the exhaustive search title, the total distance for this method was shorter at 1426 as opposed to the total distance for local search at 1504.

```
graph.txt
12
233 105
326 316
69 123
385 74
172 188
45 214
96 160
75 410
287 247
197 88
333 372
120 45
```

**Figure 4:** This image is a txt file that served as input for the graphical representations that were generated in the image above and follows the same rules as that of the previous txt file shown in Figure 2.



## UML Class Diagram:



## Code:

### Main.java:

```
import java.util.Arrays;
import javax.swing.*;
```

```

public class Main {
 public static void main(String[] args)
 {
 new Main();
 }

 public Main()
 {
 Graph g = new Graph("graph.txt");

 System.out.println(g.toString());

 FrameDisplay frame = new FrameDisplay(g);
 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 frame.setVisible(true);
 }
}

```

### **FrameDisplay.java:**

```

import javax.swing.*.*;

public class FrameDisplay extends JFrame
{
 int WIDTH = 900;
 int HEIGHT = 900;

 public FrameDisplay(Graph g)
 {
 setTitle("Graph Display");
 setSize(WIDTH, HEIGHT);

 GraphDisplay panel = new GraphDisplay(g);
 add(panel);
 }
}

```

### **PermutationNeighborhood.java:**

```

public class PermutationNeighborhood
{

 private final int[] p; //permutation
 private final int SIZE; //size of permutation
 private int loc1; //loc1 and loc2 are the locations of
 private int loc2; //p that will be swapped next

 /**

```

```

* Initializes permutation of this object and locations
* whose values will be swapped.
*
* @param a permutation whose neighborhood is to be generated
*/
public PermutationNeighborhood(int[] a)
{
 SIZE = a.length;
 p = new int[SIZE];
 System.arraycopy(a, 0, p, 0, SIZE);

 loc1 = 0;
 loc2 = 1;
}

/**
* Returns true if at least a neighbor remains to be generated; false
* otherwise.
*/
public boolean hasNext()
{
 return loc1 != SIZE - 1;
}

/**
* Returns next permutation neighbor.
*
* @return next permutation neighbor, if one exists; null otherwise
*/
public int[] next()
{
 if (hasNext())
 {
 //copy p to a
 int[] a = new int[SIZE];
 System.arraycopy(p, 0, a, 0, SIZE);

 //exchange elements at locations loc1 and loc2
 a[loc1] = p[loc2];
 a[loc2] = p[loc1];

 //advance loc1 and loc2
 if (loc2 == SIZE - 1)
 {
 loc1++;
 loc2 = loc1 + 1;
 }
 else
 loc2++;

 return a;
 }
 else
 return null;
}

```

```
}
```

### GraphDisplay.java:

```
import javax.swing.*;
import java.awt.*;
import java.util.Arrays;

public class GraphDisplay extends JPanel
{
 private int verticesNumber;
 private double[][] matrix; //adjacency matrix
 private int[] xCoordinates;
 private int[] yCoordinates;
 private int[] localSearch; //local search route
 private int[] exhaustiveSearch; //exhaustive search route
 private double totalDistanceLS; //total distance (local search)
 private double totalDistanceES; //total distance (exhaustive search)
 private int maxXCoordinate;

 public GraphDisplay(Graph g)
 {
 verticesNumber = g.getNumberOfVertices();
 matrix = g.getMatrix();
 localSearch = new int[verticesNumber];
 totalDistanceLS = g.TSP_localSearch(localSearch);
 exhaustiveSearch = new int[verticesNumber];
 totalDistanceES = g.TSP_exhaustiveSearch(exhaustiveSearch);
 xCoordinates = g.getXCoordinates();
 yCoordinates = g.getYCoordinates();
 maxXCoordinate = g.getMaxXCoordinate(xCoordinates);
 }

 public void paint(Graphics g)
 {
 int width = 15;
 int height = 15;
 int labelX = 4;
 int labelY = 10;
 int leftX;
 int topY;
 int startPt;
 int endPt;
 int xStartPt;
 int yStartPt;
 int xEndPt;
 int yEndPt;
 int shift = width / 2;
 int translation = 50;
 int x1;
 int y1;

 int x2;
 int y2;
 int midPtX;
 int midPtY;
 int roundedWeight;

 //labels the local search graph and displays the best route array
 //and the total distance*/
 g.setColor(Color.BLACK);
 g.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 12));
 leftX = 30;
 topY = 20;
 String distance = Integer.toString((int) Math.round(totalDistanceLS));
 g.drawString("Local Search:", leftX, topY);
 g.setFont(new Font(Font.SANS_SERIF, Font.PLAIN, 12));
 topY = 30;
 g.drawString("Best Route: " + Arrays.toString(localSearch), leftX, topY);
 topY = 40;
 g.drawString("Total Distance: " + distance, leftX, topY);

 //draws the best route using local search
 for (int i = 0; i < localSearch.length - 1; i++)
 {
 startPt = localSearch[i];
 endPt = localSearch[i + 1];
 xStartPt = xCoordinates[startPt];
 yStartPt = yCoordinates[startPt];
 xEndPt = xCoordinates[endPt];
 yEndPt = yCoordinates[endPt];

 g.setColor(Color.BLUE);
 g.drawLine(xStartPt + shift, yStartPt + shift, xEndPt + shift, yEndPt + shift);

 //draws the weights
 x1 = xStartPt;
 y1 = yStartPt;
 x2 = xEndPt;
 y2 = yEndPt;

 midPtX = (x1 + x2) / 2;
 midPtY = (y1 + y2) / 2;
 roundedWeight = (int) Math.round(matrix[startPt][endPt]);

 g.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 8));
 g.setColor(Color.BLACK);
 g.drawString(Integer.toString(roundedWeight), midPtX + shift, midPtY + shift);
 }
 /*connects the final pt. in the route to the starting pt.*/
 g.setColor(Color.BLUE);
 }
}
```

```

g.drawLine(xCoordinates[localSearch[0]] + shift, yCoordinates[localSearch[0]] + shift,
xCoordinates[localSearch[localSearch.length - 1]] + shift,
yCoordinates[localSearch[localSearch.length - 1]] + shift);

/*draws the weight of the edge between the final pt. in the route
and the starting pt.*/
startPt = localSearch[0];
endPt = localSearch[localSearch.length - 1];

x1 = xCoordinates[startPt];
y1 = yCoordinates[startPt];
x2 = xCoordinates[endPt];
y2 = yCoordinates[endPt];

midPtX = (x1 + x2) / 2;
midPtY = (y1 + y2) / 2;

roundedWeight = (int) Math.round(matrix[startPt][endPt]);
g.setColor(Color.BLACK);
g.drawString(Integer.toString(roundedWeight), midPtX + shift, midPtY + shift);

/*labels the exhaustive search graph and displays the best route array
and the total distance*/
g.setColor(Color.BLACK);
g.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 12));
leftX = maxXCoordinate + translation;
topY = 20;
distance = Integer.toString((int) Math.round(totalDistanceES));
g.drawString("Exhaustive Search:", leftX, topY);
g.setFont(new Font(Font.SANS_SERIF, Font.PLAIN, 12));
topY = 30;
g.drawString("Best Route: " + Arrays.toString(exhaustiveSearch), leftX, topY);
topY = 40;
g.drawString("Total Distance: " + distance, leftX, topY);
translation = 20;

//draws the best route using exhaustive search
for (int i = 0; i < exhaustiveSearch.length - 1; i++)
{
 startPt = exhaustiveSearch[i];
 endPt = exhaustiveSearch[i + 1];
 xStartPt = xCoordinates[startPt] + maxXCoordinate + translation;
 yStartPt = yCoordinates[startPt];
 xEndPt = xCoordinates[endPt] + maxXCoordinate + translation;
 yEndPt = yCoordinates[endPt];

 g.setColor(Color.BLUE);
 g.drawLine(xStartPt + shift, yStartPt + shift, xEndPt + shift, yEndPt + shift);

 //draws the weights
 x1 = xStartPt;
 y1 = yStartPt;
 x2 = xEndPt;
 y2 = yEndPt;

 midPtX = (x1 + x2) / 2;
 midPtY = (y1 + y2) / 2;
 roundedWeight = (int) Math.round(matrix[startPt][endPt]);

 g.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 8));
 g.setColor(Color.BLACK);
 g.drawString(Integer.toString(roundedWeight), midPtX + shift, midPtY + shift);
}
/*connects the final point in the route to the starting point*/
g.setColor(Color.BLUE);
g.drawLine(xCoordinates[exhaustiveSearch[0]] + maxXCoordinate + translation + shift,
yCoordinates[exhaustiveSearch[0]] + shift,
xCoordinates[exhaustiveSearch[exhaustiveSearch.length - 1]] + maxXCoordinate + translation +
shift,
yCoordinates[exhaustiveSearch[exhaustiveSearch.length - 1]] + shift);

/*draws the weight of the edge between the final pt. in the route
and the starting pt.*/
startPt = exhaustiveSearch[0];
endPt = exhaustiveSearch[exhaustiveSearch.length - 1];

x1 = xCoordinates[startPt] + maxXCoordinate + translation;
y1 = yCoordinates[startPt];
x2 = xCoordinates[endPt] + maxXCoordinate + translation;
y2 = yCoordinates[endPt];

midPtX = (x1 + x2) / 2;
midPtY = (y1 + y2) / 2;

roundedWeight = (int) Math.round(matrix[startPt][endPt]);
g.setColor(Color.BLACK);
g.drawString(Integer.toString(roundedWeight), midPtX + shift, midPtY + shift);

//draws the points for both local and exhaustive search
for (int i = 0; i < verticesNumber; i++)
{
 //draws the points for local search
 leftX = xCoordinates[i];
 topY = yCoordinates[i];
 g.setColor(Color.PINK);
 g.fillOval(leftX, topY, width, height);
 g.setColor(Color.BLACK);
 g.drawString(Integer.toString(i), leftX + labelX, topY + labelY);
}
}
}

g.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 10));

/*if the point has a label with more than one integer,
decrease the labelX to center the label inside the point*/
if (i > 9)
{
 labelX = 1;
}
g.drawString(Integer.toString(i), leftX + labelX, topY + labelY);

//draws the points for exhaustive search
leftX = xCoordinates[i] + maxXCoordinate + translation;
g.setColor(Color.PINK);
g.fillOval(leftX, topY, width, height);
g.setColor(Color.BLACK);
g.drawOval(leftX, topY, width, height);
g.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 10));
g.drawString(Integer.toString(i), leftX + labelX, topY + labelY);
}
}
}

```

### Graph.java:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

public class Graph
{
 private int verticesNumber;
 private double[][] matrix; //adjacency matrix
 private int[] xCoordinates;
 private int[] yCoordinates;

 /**
 * Instantiates a graph and initializes it with info from a text file.
 *
 * @param filename text file with graph info
 */
 public Graph(String filename)
 {
 File input = new File(filename);

 Scanner in = null;
 try
 {
 in = new Scanner(input);
 }
 catch(FileNotFoundException e)
 {
 System.out.println("File not found!");
 System.exit(0);
 }

 while (in.hasNextLine())
 {
 verticesNumber = in.nextInt();
 matrix = new double[verticesNumber][verticesNumber];
 xCoordinates = new int[verticesNumber];
 yCoordinates = new int[verticesNumber];

 //stores the coordinates
 for (int i = 0; i < verticesNumber; i++)
 {
 xCoordinates[i] = in.nextInt();
 yCoordinates[i] = in.nextInt();
 }
 }

 /**
 * @param v given vertex
 * @return list of vertices adjacent to v stored in an array;
 * size of array = number of adjacent vertices
 */
 public int[] findAdjacencyVertices(int v)
 {
 int[] vert = new int[verticesNumber];
 int total = 0;

 for (int i=0; i<verticesNumber; i++)
 {
 if (matrix[v][i] != 0)
 {
 vert[total] = i;
 total++;
 }
 }

 return Arrays.copyOf(vert, total);
 }

 /**
 * Finds a shortest route that visits every vertex
 * exactly once and returns to the starting point.
 * Uses exhaustive search.
 *
 * @param shortestRoute array with a shortest path (return value)
 *
 * @return shortest distance
 */
 public double TSP_exhaustiveSearch(int[] shortestRoute)
 {
 //initialize shortestRoute
 for (int i = 0; i < verticesNumber; i++)
 {
 shortestRoute[i] = i;
 }

 int[] a = new int[verticesNumber];
 TSP_exhaustiveSearch(shortestRoute, a, 0);

 return totalDistance(shortestRoute);
 }

 /**
 * Finds a shortest route that visits every vertex
 * exactly once and returns to the starting point.
 * Uses local search, so optimal solution is not

```

```

//stores the pairwise distances as weights
for(int i=0; i<verticesNumber; i++)
{
 for(int j=0; j<verticesNumber; j++)
 {
 matrix[i][j] = pairwiseDistance(xCoordinates[i], yCoordinates[i], xCoordinates[j],
yCoordinates[j]);
 }
}

in.close();
}

public void addEdge(int v, int w)
{
 matrix[v][w] = 1;
 matrix[w][v] = 1;
}

public void removeEdge(int v, int w)
{
 matrix[v][w] = 0;
 matrix[w][v] = 0;
}

public int getNumberOfVertices()
{
 return verticesNumber;
}

public double[][] getMatrix()
{
 return matrix;
}

public int[] getXCoordinates()
{
 return xCoordinates;
}

public int[] getYCoordinates()
{
 return yCoordinates;
}

/**
 * Finds vertices adjacent to a given vertex.

```

```

* obtained, in general.
*
* @param shortestRoute array with a shortest path (return value)
*
* @return shortest distance
*/
public double TSP_localSearch(int[] shortestRoute)
{
 double bestDistance;

 //generate initial solution as a random permutation
 int[] a = new int[verticesNumber];
 randomPermutation(a);

 //shortestRoute = initial solution
 System.arraycopy(a, 0, shortestRoute, 0, verticesNumber);
 bestDistance = totalDistance(shortestRoute);

 boolean betterSolutionFound;
 /**
 * Loop will continue as long as there is a neighbor that improves
 * best distance obtained so far.
 */
 do
 {
 betterSolutionFound = false;
 PermutationNeighborhood pn = new PermutationNeighborhood(shortestRoute);
 while (pn.hasNext())
 {
 a = pn.next();
 double currentDistance = totalDistance(a);
 if (currentDistance < bestDistance)
 {
 //shortestRoute = current solution
 System.arraycopy(a, 0, shortestRoute, 0, verticesNumber);
 bestDistance = currentDistance;
 betterSolutionFound = true;
 }
 }
 } while (betterSolutionFound);

 return bestDistance;
}

/**
 * Given an array, generates random permutation of values in [0, n-1].
 * where n is size of given array; random permutation will be stored
 * in the array. Uses Fisher-Yates shuffle algorithm.
 *

```

```

* @param a output array
*/
public void randomPermutation(int[] a)
{
 for (int i = 0; i < a.length; i++)
 {
 a[i] = i;
 }

 Random rnd = new Random();
 for (int i = a.length - 1; i > 0; i--)
 {
 //generates a random index in [0, i]
 int randomLocation = rnd.nextInt(i + 1);

 if (randomLocation != i)
 {
 //swap a[i] and a[randomLocation]
 int temp = a[i];
 a[i] = a[randomLocation];
 a[randomLocation] = temp;
 }
 }
}

/**
 * Calculates distance of a given route
 *
 * @param a route
 *
 * @return distance of a route
 */
double totalDistance(int[] a)
{
 int n = verticesNumber;

 //add weights of all edges in the path
 double totalWeight = 0;
 for (int i = 0; i < n; i++)
 {
 double weight = matrix[a[i]][a[(i + 1)%n]];
 totalWeight += weight;
 }

 return totalWeight;
}

/**
 * Recursive algorithm.

```

```

*
* @param a array partially filled with permutation
* @param k index of current element in permutation
*/
private void TSP_exhaustiveSearch(int[] shortestRoute, int[] a, int k)
{
 if (k == a.length)
 {
 if (totalDistance(a) < totalDistance(shortestRoute))
 {
 System.arraycopy(a, 0, shortestRoute, 0, verticesNumber);
 }
 //System.out.print(totalDistance(a) + " ");
 //printArray(a);
 }
 else
 {
 ArrayList<Integer> Sk = constructCandidateSet(a, k);
 for (int s : Sk)
 {
 a[k] = s;
 TSP_exhaustiveSearch(shortestRoute, a, k + 1);
 }
 }
}

/**
 * Construct candidate set (set will contain elements not used
 * in locations [0, k-1] of array a).
 */
private ArrayList<Integer> constructCandidateSet(int[] a, int k)
{
 ArrayList<Integer> candidates = new ArrayList<>();
 boolean[] b = new boolean[a.length];

 for (int i = 0; i < k; i++)
 {
 b[a[i]] = true;
 }

 for (int i = 0; i < a.length; i++)
 {
 if (!b[i]) candidates.add(i);
 }

 return candidates;
}

/**

```

```

* Calculates the distance between two points using the Euclidean distance formula.
* @param x1 x-coordinate of pt. 1
* @param y1 y-coordinate of pt. 1
* @param x2 x-coordinate of pt. 2
* @param y2 y-coordinate of pt. 2
* @return the distance between two points
*/
public double pairwiseDistance(int x1, int y1, int x2, int y2)
{
 double distance = Math.sqrt(Math.pow(x2 - x1, 2) + (Math.pow(y2 - y1, 2)));
 return distance;
}

/**
 * Gets the maximum x-coordinate from a set of coordinates
 * @param xCoordinates the set of all x-coordinates received from the input
 * @return the largest value of x
 */
public int getMaxXCoordinate(int[] xCoordinates)
{
 int max = xCoordinates[0];

 for (int i = 0; i < xCoordinates.length; i++)
 {
 if (xCoordinates[i] > max)
 {
 max = xCoordinates[i];
 }
 }

 return max;
}

/**
 * Prints array a.
 */
private void printArray(int[] a)
{
 for (int v : a)
 {
 System.out.print(v + " ");
 }

 System.out.println("");
}

public String toString()
{
 String s = "";

```

```

s += "Vertices Number: \n";

s += verticesNumber + "\n\n";

s += "Coordinates: \n";

for (int i = 0; i < verticesNumber; i++)
{
 s += "(" + xCoordinates[i] + ") " + " " + "(" + yCoordinates[i] + ") " + "\n";
}

s += "\nAdjacency Matrix: \n";

for (int i=0; i<verticesNumber; i++)
{
 for (int j=0; j<verticesNumber; j++)
 {
 s += Math.round(matrix[i][j]) + " ";
 }
 s += "\n";
}

return s;
}
}

```

Michael Cassidy

## Michael Cassidy Coding and Diagram Examples

### **Capstone Discussion Page**

This was a quick project thrown together to demonstrate knowledge of Python and SQL usage.

User Stories:

- 1 - Allow users to upload references. References should be categorized with the name of the uploader, a link to the reference, a title, and a brief description. An ID should be generated for each reference.
- 2 - Allow users to comment on references. Comments should be displayed under reference information and be categorized by ID, commenter name, and the comment.
- 3 - All information should be hosted in a postgres database with table population done on the page itself.



Code:

```
Create a streamlit app to upload reference links to a postgres database.

import streamlit as st
import pandas as pd
import psycopg2
from collections import OrderedDict
import numpy
from psycopg2.extensions import register_adapter, AsIs

def addapt_numpy_float64(numpy_float64):
 return AsIs(numpy_float64)
def addapt_numpy_int64(numpy_int64):
 return AsIs(numpy_int64)
register_adapter(numpy.float64, addapt_numpy_float64)
register_adapter(numpy.int64, addapt_numpy_int64)

Initialize connection to local postgres database.
def init_connection():
 conn = psycopg2.connect(
 host="localhost",
 database="Capstone",
 user="postgres",
 password="xxx"
)
 cur = conn.cursor()
 return conn, cur

st.set_page_config(
 page_title="Capstone Discussion",
 layout="wide",
 menu_items={
 "Get Help": "https://docs.streamlit.io/en/stable/troubleshooting/clean-install.html",
 "About": "Easy discussion and storage for references."
 }
)

st.title("Capstone Discussion")
st.header("Upload and discuss references.")

add_selectbox = st.sidebar.selectbox(
 "Select a function",
 ["Upload", "Discuss"]
)

if add_selectbox == "Upload":
 # Allows user to upload references to a postgres database, as well as the name of whoever is uploading it and a title and description for the reference.
 st.subheader("Upload references")
 name = st.text_input("Name")
 title = st.text_input("Title")
 link = st.text_input("Link")
 description = st.text_input("Description")
 if st.button("Submit"):
 conn, cur = init_connection()
 cur.execute("SELECT * FROM refs")
 data = cur.fetchall()
 #Use ordered dict to preserve order of columns.
 df = pd.DataFrame(data, columns=OrderedDict([("reference_id", None), ("name", None), ("title", None), ("link", None), ("description", None)]))
```

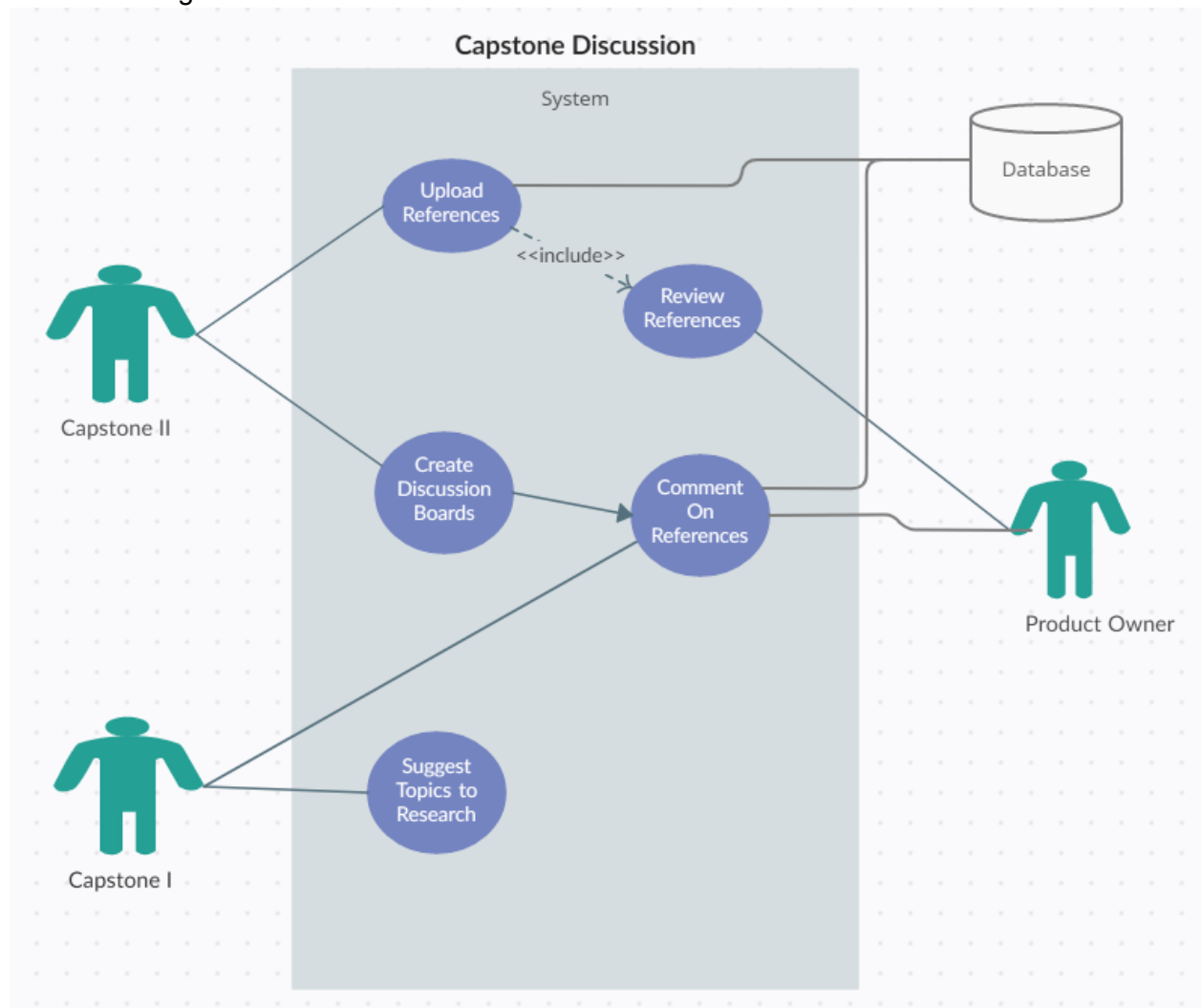
```

df = pd.DataFrame(data, columns=OrderedDict([("reference_id", None), ("name", None), ("title", None), ("link", None), ("description", None)]))
#Generate ID for reference as it's uploaded.
if len(df) == 0:
 reference_id = 1
else:
 reference_id = df["reference_id"].max() + 1
#Insert reference into database using ordered dict.
cur.execute(
 "INSERT INTO refs (reference_id, name, title, link, description) VALUES (%s, %s, %s, %s, %s)",
 (reference_id, name, title, link, description),
)
conn.commit()
st.success("Your reference has been added!")
conn.close()

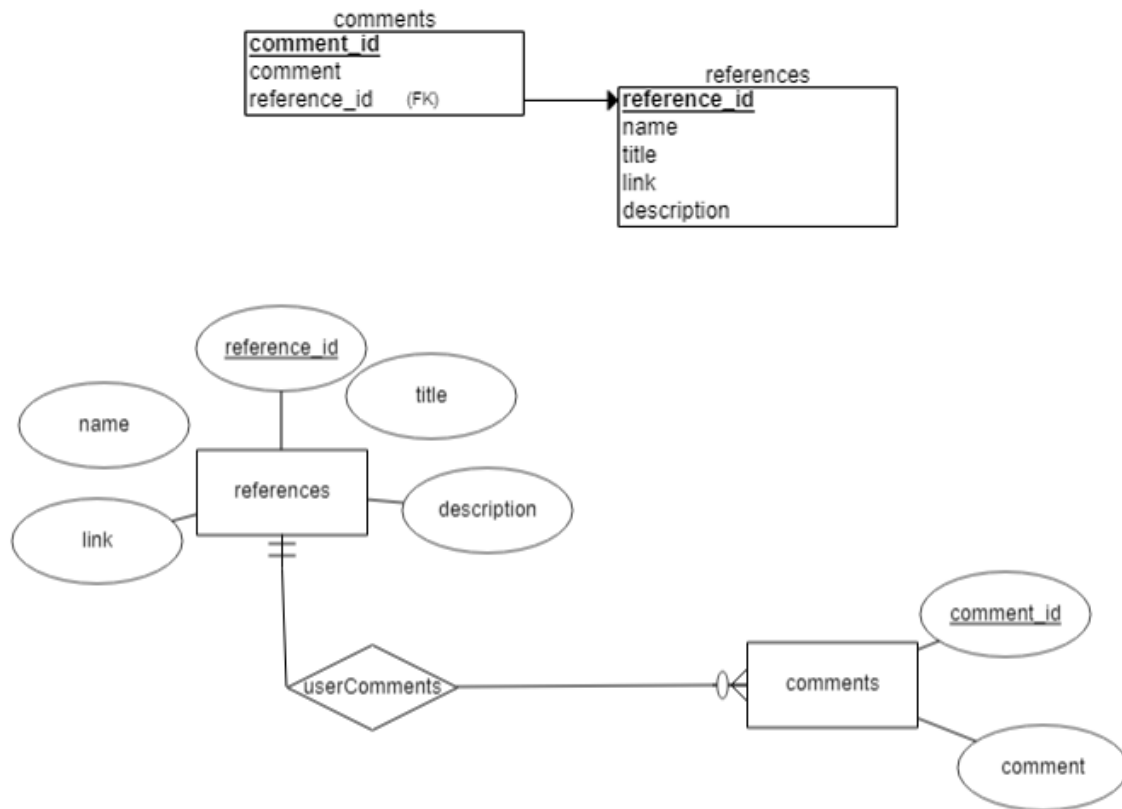
else:
Allows user to discuss references that have been uploaded to the database.
st.subheader("Discuss references")
conn, cur = init_connection()
If no references have been uploaded, display a message.
cur.execute("SELECT * FROM refs")
data = cur.fetchall()
if len(data) == 0:
 st.write("No references have been uploaded yet.")
else:
df = pd.DataFrame(data, columns=OrderedDict([("reference_id", None), ("name", None), ("title", None), ("link", None), ("description", None)]))
conn.close()
#Display all references in a table, hiding array number.
st.table(df[["reference_id", "name", "title", "link", "description"]])
#Allow user to select a reference to discuss.
selected_ref = st.selectbox("Select a reference", df["reference_id"])
#Display reference information, format link as a hyperlink.
st.write("Reference ID: ", selected_ref)
st.write("Name: ", df.loc[df["reference_id"] == selected_ref, "name"].values[0])
st.write("Title: ", df.loc[df["reference_id"] == selected_ref, "title"].values[0])
st.write("Link: ", df.loc[df["reference_id"] == selected_ref, "link"].values[0])
st.write("Description: ", df.loc[df["reference_id"] == selected_ref, "description"].values[0])
#Allow user to add a comment to the reference.
comment = st.text_input("Add a comment")
name = st.text_input("Name")
if st.button("Submit"):
 conn, cur = init_connection()
 cur.execute("SELECT * FROM coms")
 data = cur.fetchall()
 df = pd.DataFrame(data, columns=OrderedDict([("comment_id", None), ("reference_id", None), ("name", None), ("comment", None)]))
 if len(df) == 0:
 comment_id = 1
 else:
 comment_id = df["comment_id"].max() + 1
 cur.execute(
 "INSERT INTO coms (comment_id, reference_id, name, comment) VALUES (%s, %s, %s, %s)",
 (comment_id, selected_ref, name, comment),
)
 conn.commit()
 st.success("Your comment has been added!")
 conn.close()
#Display all comments for the selected reference.
conn, cur = init_connection()
cur.execute("SELECT * FROM coms")
data = cur.fetchall()
df = pd.DataFrame(data, columns=OrderedDict([("comment_id", None), ("reference_id", None), ("name", None), ("comment", None)]))
conn.close()
st.table(df.loc[df["reference_id"] == selected_ref, ["name", "comment"]])

```

Use Case Diagram:



## ER Diagram and Schema:



## Webpage Example:

The screenshot shows a web application interface for discussing references. The page title is "Capstone Discussion" and the subtitle is "Upload and discuss references." The main heading is "Discuss references".

Below the heading is a table with 5 columns: **reference\_id**, **name**, **title**, **link**, and **description**. The table contains 2 rows of data:

|   | reference_id | name    | title            | link                                                                                                                                                            | description                             |
|---|--------------|---------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| 0 | 1            | Michael | What is Edge ML? | <a href="https://www.ferriselectronics.com/electronics/what-edge-machine-learning">https://www.ferriselectronics.com/electronics/what-edge-machine-learning</a> | Introduction to edge ML for new people. |
| 1 | 2            | Mike    | Github Link      | <a href="https://github.com/MCassidy01/EdgeML-FIU-Research-Project">https://github.com/MCassidy01/EdgeML-FIU-Research-Project</a>                               | Read the full PDF here.                 |

Below the table is a form to "Select a reference". The selected reference is 1. The form fields are:

- Reference ID: 1
- Name: Michael
- Title: What is Edge ML?
- Link: <https://www.ferriselectronics.com/electronics/what-edge-machine-learning>
- Description: Introduction to edge ML for new people.

Below the form is a section to "Add a comment". The form fields are:

- Name: Mike
- Comment: Very helpful, I appreciate the introduction.

Below the form is a table with 3 columns: **name**, **comment**, and **comment**. The table contains 2 rows of data:

|   | name    | comment                                      |
|---|---------|----------------------------------------------|
| 0 | Jerry   | Very helpful, I appreciate the introduction. |
| 1 | Michael | No problem, hope it helps.                   |

## Automated Essay Grading:

User Stories worked on:

7 - As a developer, I want to be able to compare the semantic meaning of a sentence (semantic similarity)

8 - As a developer, I want to be able to score the readability of students' answers.

10 - As a professor, I want to be able to upload a question and answer set to the database. Then, I want to be able to specify how the answer should be graded, and any topics that should be identified in the student's answer

13 - As a student, I want to be able to view my essay feedback on a results page that is detailed, informative, and easy to navigate

14 - As a student, I want to know if I need to improve the academic or professional tone of my writing

16 - As a student, I want to know if my answer is concise, and see any improvements that I could make to make my answer more "to the point"

Code Examples:

```
332 338 @@ -338,7 +338,8 @@ def result(std_answer, prof_answer, prof_answer_data, question_id):
338 339 'structureScore': int(100 * (1 - (len(std_results['missing']) / len(prof_answer_data['topics'])))),
339 340 'readabilityScore': readability(std_answer),
340 341 'sentiment': sentiment_analyzer(std_answer),
341 342 'overallScore': round(0.4*results['structureScore'] + 0.15*results['similarityScore'] + 0.15*results['coherenceScore'] + 0.15*results['grammarScore'] + 0.15*results['readabilityScore'])
342 343 'concisenessScore': 100 - (conciseness(std_answer)*5),
343 344 'overallScore': round(0.4*results['structureScore'] + 0.15*results['similarityScore'] + 0.15*results['concisenessScore'] + 0.15*results['grammarScore'] + 0.15*results['readabilityScore'])
344 345 }}

452 453 @@ -452,8 +453,6 @@ def sentiment_analyzer(student_answer):
453 454 elif max(scores_list) == results['neu']:
454 455 return ('neu', results['neu'])
455 456
456 457 # calculate student readability using automated readability and store result and suggestions as json object
457 458 def readability(text):
458 459 readability = textstat.flesch_reading_ease(text)
459 460
460 461 @@ -471,4 +470,17 @@ def readability(text):
471 472)
472 473 return json.dumps(scoring)
473 474
474 475 # uses language tool to check for errors relating to conciseness, then returns the number of relevant errors
475 476 def conciseness(text):
476 477 api_url = 'https://api.languagetoolplus.com/v2/check'
477 478 # enabledOnly not set to true as it fails to return any issues
478 479 payload = dict(text = text, language = 'en-US', motherTongue = 'en-US', enabledRules = ['BE_A_X_ONE', 'DUE_TO_THE_FACT', 'EN_PLAIN_ENGLISH_REPLACE'], enabledCategories = ['REDUNDANCY'], enabledOnly = 'false', level = 'picky')
479 480 response = requests.post(api_url, payload)
480 481 json_response = json.loads(response.text)
481 482 matches = json_response['matches']
482 483 i = 0
483 484 for error in matches:
484 485 if error['rule']['id'] in ('EN_PLAIN_ENGLISH_REPLACE', 'IN_X_ORDER', 'DUE_TO_THE_FACT', 'TOO_LONG_SENTENCE') or error['rule']['category'] == 'REDUNDANCY':
485 486 i = i+1
486 487 return i
```

```
↑
....
@@ -15,6 +15,7 @@
15 15 import json
16 16 from database.repository_base import RepositoryBase
17 17 from nltk.sentiment.vader import SentimentIntensityAnalyzer
18 + import textstat
18 19
19 20 nlp = spacy.load("en_core_web_lg")
20 21 word_vectors = api.load("glove-wiki-gigaword-100")
↓
↑
....
@@ -434,3 +435,18 @@ def sentiment_analyzer(student_answer):
434 435 elif max(scores_list) == results['neu']:
435 436 return {'sentiment: ': results['neu']}
436 437
438 + #calculate student readability using automated readabillity and store result and suggestions as json object
439 + def readability(text):
440 + readability = textstat.automated_readability_index(text)
441 + suggestion = ""
442 + scoring = {
443 + 'readability': readability,
444 + 'suggestion': suggestion
445 + }
446 + if readability < 10.0:
447 + suggestion = "Your answer is too simple, please try to make it more professional."

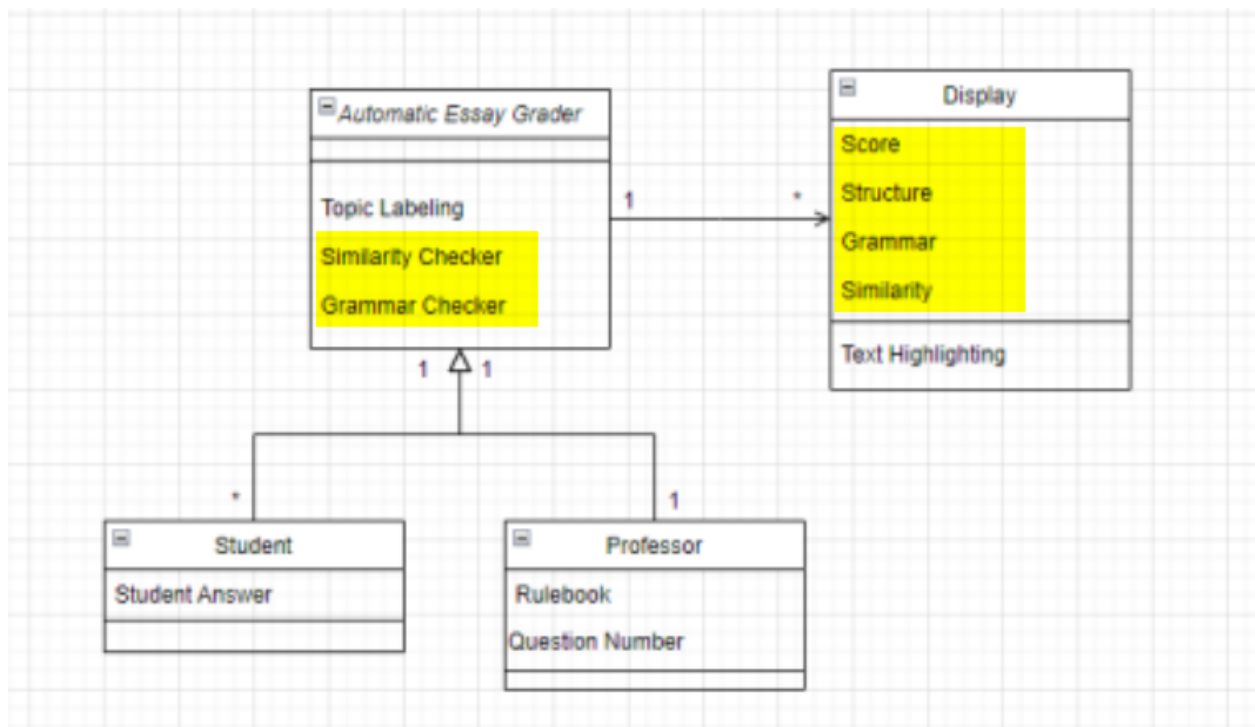
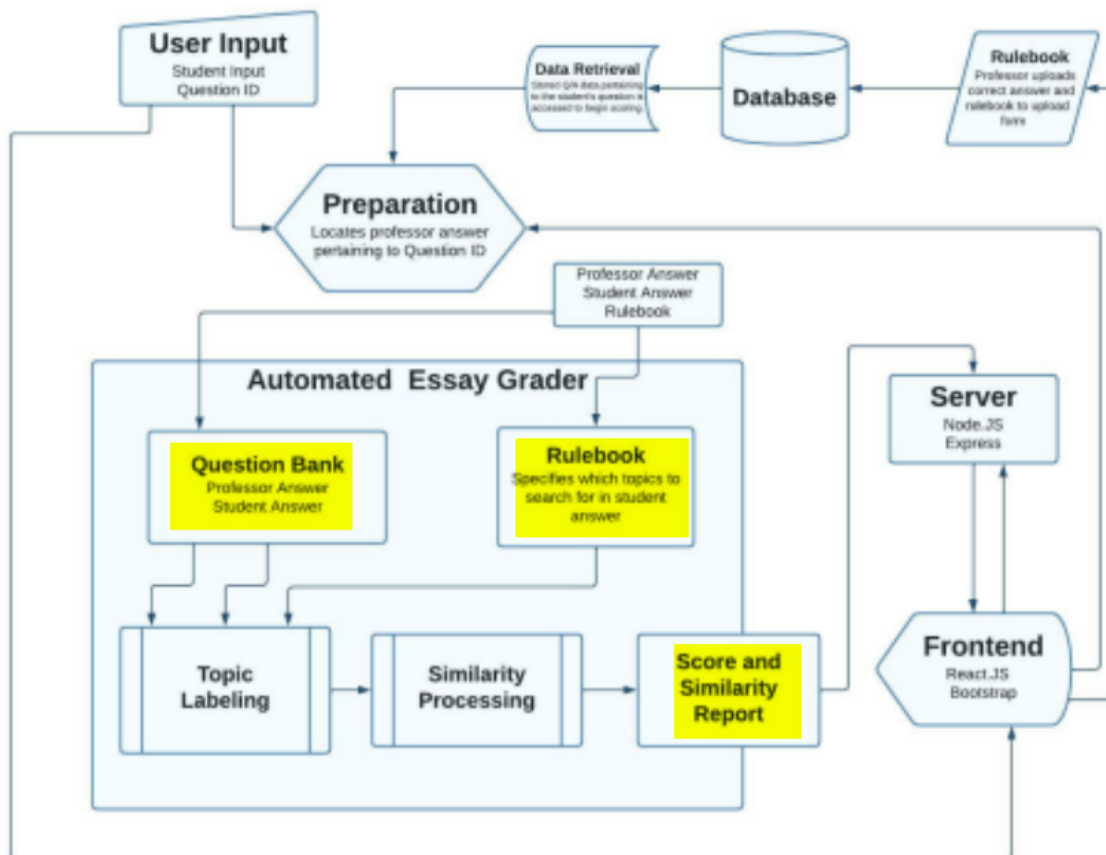
ghost on Jul 8
Avoid mentioning professionalism when talking about readability. Instead, suggest that the student improves their answer by raising the reading level of their answer.

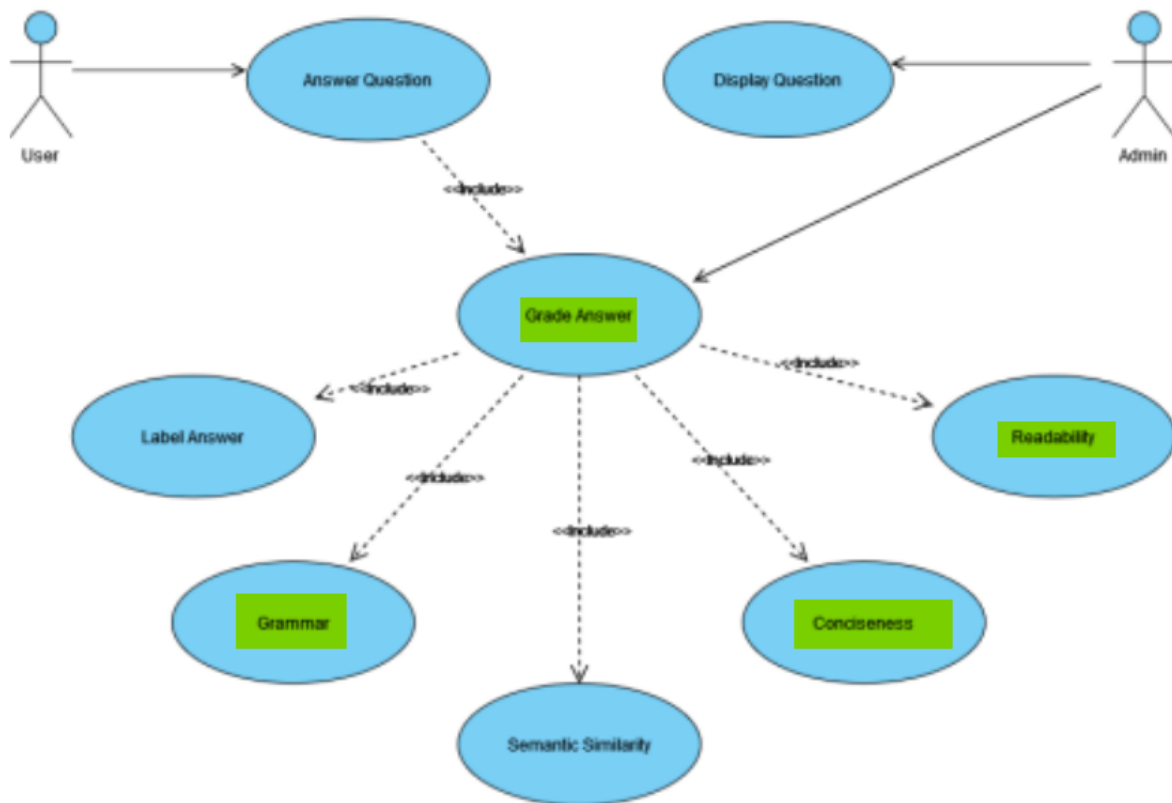
Reply...

Resolve conversation

448 + elif readability > 20.0:
449 + suggestion = "Your answer is too complex, please try to make it more readable."
450 + else:
451 + suggestion = "Your answer is readable."
452 + return json.dumps(scoring)
```

Diagrams with Contributions highlighted:





Jobin Kurian

Jobin Kurian

CIS4591

Capstone Documentation

Due to a lack of understanding of the project, this section will be dedicated to the professor in proving our abilities that we know how to code.

## 1). MYSQL (Database skills)

### Task 1: Create a database using PostgreSQL

```
CREATE TABLE Employees(
employee_id integer PRIMARY KEY,
```



```
name_ text NOT NULL,
hire_date date,
Salary numeric,
email char(30) UNIQUE,
level_ text NOT NULL
);

CREATE TABLE Companies(
company_id integer PRIMARY KEY,
name_ text NOT NULL,
industry text,
adress char(50),
num_employees integer NOT NULL,
description text
);

CREATE TABLE WorkAssignments(
assignment_id integer PRIMARY KEY,
due_date date NOT NULL,
difficulty text NOT NULL,
Company_id integer,
FOREIGN KEY (Company_id) REFERENCES Companies(company_id)
);

CREATE TABLE Performance(
employee_id integer,
assignment_id integer,
PRIMARY KEY(employee_id,assignment_id),
FOREIGN KEY (employee_id) references Employees(employee_id),
```

FOREIGN KEY (assignment\_id) references  
WorkAssignments(assignment\_id), score integer  
);

Insert Data

```
INSERT INTO Employees(employee_id, name_, hire_date,
Salary, email, level_) VALUES(1,'Michael
Keyes','06/15/2020','30000','mkeye001@netw.com','parttim
e'), (2,'Ellis
Carlisle','04/24/2017','50000','ecarl001@vers.com','fulltime')
,
(3,'Greg Garry','01/09/2018','55000','ggarr001@netw.com','parttime'),
(4,'Randy North','12/08/2019','28000','rnort001@
vers.com','parttime'), (5,'Barbara
Nunez','11/08/2010','71000','bnune001@jgreen.com ','fulltime'),
(6,'Shirley Reid','12/05/2007','89000','sreid001@itconnect.com
,','fulltime'), (7,'Braden
Robins','08/22/2009','77000','brobi001@jgreen.com ','fulltime'),
(8,'Brendon
Moffett','07/31/2015','62000','bmoff001@netw.com','parttime');
INSERT INTO companies(company_id, name_, industry, adress,
num_employees, description)
VALUES(1,'Net World','Internet Services','11345 SW 56 ST', '800',
'Internet Service provider working in
99.9% of areas');
(2,'Just Green','Grocers','4467 NW 8 ST', '10000', 'Sustainable agriculture for all'),

(3,'Verse','Research','8754 SW 134 TER', '150', 'Research Thinktank
specializing in the natural sciences'),

(4,'IT Connect','Internet Services','2105 NE 17 ST', '5400', 'Help solve
everyones IT problems!'); INSERT INTO
WorkAssignments(assignment_id, due_date, difficulty, Company_id)
VALUES(1,'02/18/2022','easy','1');
(1,'02/18/2022','easy','1')

(2,'03/10/2022','hard','2')

(3,'01/29/2022','medium','4');

(4,'05/30/2022','hard','1');

(5,'04/12/2022 ','easy','3');

(6,'09/17/2022 ','easy','3');
```

```
(7,'07/20/2022','medium','2');
```

```
INSERT INTO Performance(employee_id, assignment_id, score)
```

```
VALUES(1,'1','100'),
```

```
(1,'4','75'),
```

```
(2,'5','-1'),
```

```
(4,'5','59'),
```

```
(4,'6','29'),
```

```
(5,'7','-1'),
```

```
(5,'2','95'),
```

```
(6,'3','88');
```

Task 2. Manage a database using PostgreSQL

Query1

```
INSERT INTO Performance(employee_id, assignment_id, score)
```

```
VALUES(2,'6','77'),
```

```
(7,'2','-1'),
```

```
(7,'7','81'),
```

```
(8,'1','48'),
```

```
(8,'4','91');
```

**Query 2: Update the Performance table and set the grade to incomplete (0) where score is -1.** UPDATE Performance SET score = '0' WHERE score = '-1';

**Query 3: Add "NOT NULL" constraint to score column in the Performance table.** ALTER TABLE Performance  
ALTER COLUMN score SET NOT NULL

**Query 4: Delete the records from the table Performance where score is less than 50.** DELETE FROM Performance  
WHERE score < 50

**Query 5: Export the Performance table as Performance.csv by using SQL command "COPY". (Note: You need to be a root user to perform this action, so just provide the syntax of this query.)**

```
COPY Performance TO '/SQL/COUNTRY_DATA.CSV' WITH DELIMITER '|'
```

**Query 6: Add a new column called “company\_id” to the table Employees.** ALTER TABLE Employees  
ADD COLUMN company\_id integer;

**Query 7: Rename the column level in the table Employees to worker\_type.** ALTER TABLE Employees  
RENAME COLUMN level\_ TO worker\_type

**Query 8 : Update the Employees table by replacing previously null values in company\_id with the values as shown above.**

UPDATE employees

SET company\_id = 1

WHERE employee\_id = 1

UPDATE employees

SET company\_id = 3

WHERE employee\_id = 2

UPDATE employees

SET company\_id = 1

WHERE employee\_id = 3

UPDATE employees

SET company\_id = 3

WHERE employee\_id = 4

UPDATE employees

SET company\_id = 2

WHERE employee\_id = 5

UPDATE employees

SET company\_id = 4

WHERE employee\_id = 6

UPDATE employees

SET company\_id = 2

WHERE employee\_id = 7

UPDATE employees

SET company\_id = 1

**Query 9: Retrieve all the employee(s) (column: name) who work for Verse.** SELECT DISTINCT name\_  
FROM employees

WHERE email LIKE '%vers%';

**Query 10: List the difficulty of all work assignments for which “Michael Keyes” was scored.** SELECT difficulty  
FROM workassignments

WHERE company\_id = ANY(SELECT employee\_id  
FROM EMPLOYEES

WHERE name\_ = 'Michael Keyes');

**Query 11: Return counts grouped by difficulty for all work assignments assigned to employees who make more than \$50,000.**

SELECT difficulty, COUNT(\*)

FROM workassignments

WHERE company\_id = ANY(SELECT company\_id

FROM employees

WHERE salary > 50000)

GROUP BY difficulty

**Query 12: Return counts of the number of employees grouped by their worker\_type.** SELECT worker\_type, COUNT(\*)  
FROM employees

GROUP BY worker\_type

**Query 13: List the name(s) of all employee(s) who were assigned more than one work assignment.**

SELECT e.name\_, COUNT(assignment\_id) FROM employees e

INNER JOIN workassignments w ON e.employee\_id = w.company\_id

GROUP BY e.name\_

HAVING COUNT(assignment\_id) >= 2

**Query 14: List the name(s) and score(s) of employee(s) on easy assignments, ordered by score.**

SELECT name\_, score

FROM employees, performance, workassignments

WHERE employees.employee\_id = performance.employee\_id

AND performance.assignment\_id = workassignments.assignment\_id

AND workassignments.difficulty = 'easy'

ORDER BY score

**Query 15: List the name(s) of the employee(s) who were on at least one medium assignment and were hired after 2005.**

SELECT DISTINCT employees.name\_

FROM employees, performance

WHERE employees.employee\_id = performance.employee\_id

AND YEAR (hire\_date) > 2005

AND company\_id = ANY(SELECT company\_id

FROM workassignments

WHERE difficulty = 'medium')

**Query 16: List the name(s) of the “parttime” employee(s) who get(s) score greater than 50 for at least one work assignment.**

SELECT name\_

FROM employees

WHERE company\_id = ANY(SELECT company\_id

From performance

WHERE score > 50)

AND (worker\_type = 'parttime')

**Query 17: List the name(s) of the employees(s) and the highest work assignment scores for each employee.**

SELECT name\_, MAX(score)

```
FROM employees, performance, workassignments
WHERE employees.employee_id = performance.employee_id
AND performance.assignment_id = workassignments.assignment_id
GROUP BY name_
```

**Query 18: Group and list the scores for all the work assignments assigned to employees at “Net World”, ordered by scores.**

```
SELECT performance.score
FROM performance
WHERE performance.employee_id IN (
SELECT employees.employee_id
FROM Employees
WHERE employees.company_id = '1')
GROUP BY performance.score
```

**Query 19: List the names of the employee(s) ordered by the number of years since they have been hired. (Show the number of years in the output result)**

```
SELECT employees.name_,'(2022-03-14' - employees.hire_date) /
365 AS "years worked" FROM employees
order by "years worked"
```

**Query 20: Create the following table: title:**

```
CREATE TABLE titles(
name_ text PRIMARY KEY,
abbreviation text
);
```

**Query 21: Update the column values in “worker\_type” in the table Employees by replacing them with their corresponding abbreviation. For example, parttime should be updated to PTE.**

```
INSERT INTO titles(name_, abbreviation)
VALUES('parttime','PTE'),
('FULLTIME','FTE');
```

**Query 22: Update the Work\_Assignments table by adding another column, "requirement\_id," with the values as shown above.**

```
ALTER TABLE workassignments
```

```
add column requirement_id integer;
```

```
UPDATE workassignments SET requirement_id = 4 WHERE
assignment_id = 1; UPDATE workassignments SET
requirement_id = 7 WHERE assignment_id = 2; UPDATE
workassignments SET requirement_id = Null WHERE
assignment_id = 3; UPDATE workassignments SET
requirement_id = Null WHERE assignment_id = 4; UPDATE
workassignments SET requirement_id = 6 WHERE
assignment_id = 5;
UPDATE workassignments SET requirement_id = Null
WHERE assignment_id = 6; UPDATE workassignments SET
requirement_id = Null WHERE assignment_id = 7;
```

**Query 23: Add the foreign key constraint to the new "requirement\_id" column in the Work\_Assignments table to reference the "assignment\_id" column; that is, the table references itself.**

```
ALTER TABLE workassignments ADD FOREIGN KEY
(requirement_id) REFERENCES workassignments
```

**Query 24: List the IDs of all of the work assignments, along with the names and number of employees of the companies that manage them, and the ID of the required previous assignment, if any.**

```
SELECT
companies.name,companies.num_employees,workassignments.
assignment_id FROM companies, workassignments
WHERE companies.company_id = workassignments.company_id
```

**Query 25: List the name(s) of all the full time employees(s) and any assignments(s) that they have previously worked on, along with their final score.**

```
SELECT name,score,performance.assignment_id

FROM employees, workassignments, performance
```



WHERE employees.employee\_id = performance.employee\_id  
AND workassignments.assignment\_id = performance.assignment\_id  
AND (worker\_type = 'fulltime')

**Query 26: List the name(s) of the employees(s) that have worked on at least 2 assignments.**

```
SELECT e.name_, COUNT(assignment_id)
FROM employees e
INNER JOIN workassignments w ON e.employee_id = w.company_id
GROUP BY e.name_
HAVING COUNT(assignment_id) >= 2
```

**Query 27: List the name(s) of the part time employees(s) who have received a cumulative score across their assignment(s) greater than 80.**

```
SELECT employees.name_, SUM(performance.score)
FROM employees, performance
WHERE employees.employee_id = performance.employee_id
AND (worker_type = 'parttime')
GROUP BY name_
having SUM(score) >= 80
```

**Query 28: List the name(s) and scores(s) of all the employees(s) who received a better score in an assignment than in its requirement.**

```
SELECT E1.name, P1.score
FROM employees E1, employees E2, companies C1,
companies C2, performance P1, performance P2,
workassignments T1, workassignments T2
WHERE E1.employee_id = P1.employee_id
AND P1.assignment_id = T1.assignment_id
AND T1.company_id = C1.company_id
AND E2.employee_id = P2.employee_id
```

AND P2.assignment\_id = T2.assignment\_id  
AND T2.company\_id = C2.company\_id  
AND T1.requirement\_id = T2.assignment\_id  
AND P1.score > P2.score;

## 2) Java skills

a) designed a Banking application in Java using the principles of object-oriented programming including inheritance, abstraction and encapsulation

```
public abstract class Account {
 protected double balance;
 private String customerName;
 private String customerId;

 public Account(){
 balance =0;
 }

 public Account(String customerName, String
 customerId) { this.customerName =
 customerName;
 this.customerId = customerId;
 }

 public String getName() {
 return this.customerName;
 }

 public String getID() {
 return this.customerId;
 }

 public double getBalance() {
 return this.balance;
 }

 public void deposit() {

 }

 public void withdraw() {
```

```

 }

} //class

public class bankmain {

 public static void main(String[] args) {

 CheckingAccount customer1 = new
CheckingAccount("Dr. Waqas", "WCOP3337");
 System.out.println(customer1.type +
"\n*****");
 System.out.println(customer1);
 customer1.deposit(10);
 customer1.withdraw(50000);
 System.out.println("Current Balance: $" +
customer1.getBalance() + "\n\n");

 savingsAccount customer2 = new
savingsAccount("Dr. Waqas", "WCOP3337");
 System.out.println(customer2.type +
"\n*****");
 customer2.setInterest(0.04);
 System.out.println(customer2);
 customer2.deposit(500);
 customer2.showInterest();
 System.out.println("Current Balance: $" +
customer2.getBalance() + "\n\n");

 savingsAccount customer3 = new
savingsAccount("Dr. Waqas", "WCOP3337");
 System.out.println(customer3.type +
"\n*****");
 customer3.setInterest(0.5);
 System.out.println(customer3);
 customer3.deposit(100);
 customer3.showInterest();
 System.out.println("Current Balance: $" +
customer3.getBalance() + "\n\n");

 }
}

```

```

}

public class CheckingAccount extends Account{

 static double FEE = 2.0;
 public String type = "Checking Account";

 public CheckingAccount() {
 super();
 }

 public CheckingAccount(String customerName, String
 customerId) { super(customerName, customerId);
 }

 public CheckingAccount(double fee) {
 FEE = fee;
 }

 public void deposit(double amount) {

 if(amount > 10) {
 balance += amount;
 balance -= FEE;
 System.out.println("You have
successfully deposited $" + amount + "\nThere is a
transaction fees of $2 for this action"); }
 else if (amount > 0 && amount <= 10) {
 balance += amount;
 System.out.println("You have
successfully deposited $" + amount + "\nThere
isn't a transaction fee for $" + amount); }

 else {
 System.out.println("You must enter a positive
amount!");
 }
 }

 public void withdraw(double amount) {
 if (amount > 0) {
 if((amount+FEE) <= balance){
 balance -= amount;
 System.out.println("You successfully withdrawn $" +

```

```

amount + "\nThere is a transaction fees of $2 for
this action, your balance is: " + balance);
 }//nested
 else if (amount > balance) {
 System.out.println("You cannot
withdraw $" + amount + " due to insufficient
funds.");
 }
 }//outer
 else {
 System.out.println("You cannot withdraw negative
funds.");
 }
} //class
 public String toString() {
 return "Name: " + getName() + "\nID:
" + getID() + ", CurrentBalance: $" + balance;
 }
}

```

```

public class savingsAccount extends Account{

 public String type = "Savings Account";
 static double intRate = 0;

 //default constructor
 public savingsAccount() {
 super();
 balance = 0;
 }

 public savingsAccount(String customerName, String
customerId) { super(customerName,
customerId);

 }
 public savingsAccount(String type, double rate) {
 intRate = rate;
 }

 public void setInterest(double newRate) {
 if(newRate>=0) {

```

```

 intRate=newRate;
 }
 else {
System.out.println("Error: Negative value.");
 System.exit(0);
 }
}

 public double calcInterest(){
 return balance*intRate;
 }

 public void deposit(double amount) {
 if(amount > 0) {
 balance += amount;
System.out.println("You have successfully deposited $"
+ amount);
 }
 else {
System.out.println("You must enter a positive
amount!");
 }
 }

 public void withdraw(double amount) {
 if (amount > 0) {
 if(amount <= balance){
 balance -= amount;
 }//nested
 else if (amount > balance) {
System.out.println("You cannot withdraw this amount
due to insufficient funds.");
 }
 }//outer
 else {
System.out.println("You cannot withdraw negative
funds.");
 }
 }

 public String toString() {
 return "Name: " + getName() + "\nID: "
+ getID() + ", CurrentBalance: $" + balance;
 }

```

```

 }

 public void showInterest() {
 System.out.println("The interest rate set by
our bank is " + (intRate*10) + "% . Your projected
return of investment in one year is $" +
calcInterest());
 }

}

```

b) designed a gui quiz game application in java which uses two files one for reading questions and one for storing results and exception handling. When someone scores higher than previous player the score gets updated in the highScore.txt file

code:

```

import java.io.*;
import javax.swing.JOptionPane;
import java.util.Scanner;
public class Project_6 {
 //declare a global constant and set
 it's value to SIZE 10 final static int
 SIZE = 10;

 public static void main(String[] args) throws IOException {

 //declare and initialize variables and
 parallel arrays int menuChoice = 0;

 int Score = 0;
 String [] question1arr = new String[SIZE];
 String [] answer1Aarr = new String[SIZE];
 String [] answer2Aarr = new String[SIZE];
 String [] answer3Aarr = new String[SIZE];
 String [] answer4Aarr = new String[SIZE];
 int [] correctAnswer1arr = new int[SIZE];
 int [] pointValuearr = new int[SIZE];
 String [] blankarr = new String[SIZE];

 String [] highNames = new String[3];
 int [] highScores = new int[3];

 // initiate a do while loop for the main method
 do
 {
 //prompt for name and display the main menu

```

```

String userName = promptName();
menuChoice = displayMainMenu();

if(menuChoice == 1)
{
 //invoke dsisplayRules method
 displayRules();
}

// if user selects option 2, start the game
else if(menuChoice == 2)
{
 //declare scanner
 File infile = new File("questions.txt");
 Scanner in = new Scanner(infile);
 // Use a for loop to read the question and its answers from
 the text file and set up the accumulator
 for (int i = 0; i <10; i++)
 {
 question1arr[i] = in.nextLine();
 answer1Aarr[i] = in.nextLine();
 answer2Aarr[i] = in.nextLine();
 answer3Aarr[i] = in.nextLine();
 answer4Aarr[i] = in.nextLine();
 correctAnswer1arr[i] = in.nextInt();
 pointValuearr[i]= in.nextInt();
 blankarr[i] = in.nextLine();
 Score +=
 processQuestion(question1arr[i],answer1Aarr[i],answer2Aarr[i],
 answer3Aarr[i],answer4Aarr[i],pointValuearr[i],correctAnswer1arr[i]);
 displayScore(Score);
 }
 //close scanner
 in.close();
 //Invoke readHighScore method and store it inside highScore
 variable
 readInHighScore(highNames, highScores);

 //for testing purpose only, does not change the content of
 program in any way
 for(int i = 0; i < 3; i++) {
 System.out.println(highNames[i] + highScores[i]);
 }

 //Invoke compareScore methods

```



```

compareScore(Score, userName, highNames, highScores);
 UpdateHighScores(highNames, highScores);
 //Display goodbye message
 JOptionPane.showMessageDialog(null, "Thank
you for trying out this game.");

//reset the score only in the program not text file.
 Score = 0;
}

else if(menuChoice == 3)
{
 JOptionPane.showMessageDialog(null, "Goodbye!!", "Thank
you!", 1);
}
//end the do statement with the while
expression, make menuChoice variable not equal to 3 to end
the loop and close the program. }while(menuChoice != 3);
} //end the do while loop

//method to promptName
public static String promptName() {
 String userName = JOptionPane.showInputDialog(null,
 "Please enter your name : ", "Name", 3);
 return userName;
}

//Method #1)
// Return type: int
// Parameters: None
// Purpose: To display main menu and prompt
user for their choice. // This Method returns
menu choice
public static int displayMainMenu() {
 int menuChoice;
 String tempString;

 do {
 //Display an introduction to the game
 JOptionPane.showMessageDialog(null, "Welcome to The Code is
Right!!!", "Welcome", 1);

 //String userName =
 JOptionPane.showInputDialog(null, "Please enter your name :
", "Name", 3);

```

```

 tempString = JOptionPane.showInputDialog(null,
 "Please choose an option below. "
 + "\n1) See Rules"
 + "\n2) Play Game"
 + "\n3) Exit", "Menu", 1);

 //String to int
 menuChoice = Integer.parseInt(tempString);

 if (menuChoice < 1 || menuChoice > 3) {
 JOptionPane.showMessageDialog(null, "Invalid input! Please
 enter a number from 1-3 for your menu choice.", "Invalid",
 2); }

 }while(menuChoice < 1 || menuChoice > 3);
 return menuChoice;
 }

 //Method #2)
 // Return type: void
 // Parameters: None
 // Purpose: To display the rules of the game
 public static void displayRules() {

 JOptionPane.showMessageDialog(null, "Rule 1) In this quiz
 there will be 10 questions with multiple choice answers and a
 total score 1000 points. " + "\nRule 2) As you move through
 each question they will
 increase in difficulty."
 + "\nRule 3) Every time you get a question right you get 100
 points. If you get a question wrong you lose 100 points."
 + "\nRule 3) Have fun!!!", "Rules", 1);
 }

 //Method #3)
 // Return type: int
 // Parameters: 5 Strings and 2 ints
 // Purpose: To display the question and prompt for
 answer choice and validate.
 // Returns the points the user earned depending on if
 they were correct or incorrect

 public static int processQuestion(String question1,
 String answer1A, String answer1B, String answer1C, String
 answer1D, int pointValue, int correctAnswer1) throws
 IOException {

 int answerChoice;

```

```

 do {
 String tempString;
tempString = JOptionPane.showInputDialog(null, question1
 + "\n1) " + answer1A
 + "\n2) " + answer1B
 + "\n3) " + answer1C
 + "\n4) " + answer1D , "Question ", 3);

 answerChoice = Integer.parseInt(tempString);
//If the answerChoice is < 1 or > 4 display warning
 if (answerChoice < 1 || answerChoice > 4) {
 JOptionPane.showMessageDialog(null, "Invalid input! Please
enter a number from 1-3.", "Invalid", 2);
 }
 while (answerChoice < 1 || answerChoice > 4);

 // if answerChoice = correctAnswer return the pointValue
 // else return 0
 if (answerChoice == correctAnswer1) {
 JOptionPane.showMessageDialog(null, "correct!");
 return pointValue;
 } else {
 JOptionPane.showMessageDialog(null,
 "Incorrect!", "Incorrect", 2);
 return 0;
 }
 }

//Method# 4)
// Return type: void
// Parameters: 2 arrays(String and int)
// Purpose: To read the text file and store it
in the respective arrays.
 public static void readInHighScore(String[] name,
int[] Score) throws IOException {

 File infile1 = new File("highScore.txt");
 Scanner in1 = new Scanner(infile1);
 for(int i = 0; i < 3; i++) {
 name[i] = in1.next();
 Score[i] = in1.nextInt();
 }
 in1.close();
 }

//Method# 5)

```

```

// Return type: void
// Parameters: 2 arrays(int and string), int, and String
// Purpose: To compare the userscore with the pre
existing names and scores of the text files and setting
them equal to their respective arrays.

```

```

public static void compareScore(int userScore, String
userName, String[] name, int[] Score) throws IOException
{

```

```

 if(userScore > Score[0])
 {
 Score[2] = Score[1];
 name[2] = name[1];
 Score[1] = Score[0];
 name[1] = name[0];
 Score[0] = userScore;
 name[0] = userName;
 }
 else if (userScore > Score[1])
 {
 Score[2] = Score[1];
 name[2] = name[1];
 Score[1] = userScore;
 name[1] = userName;
 }
 else if (userScore > Score[2])
 {
 Score[2] = userScore;
 name[2] = userName;
 }
}

```

```

//Method# 6)

```

```

// Return type: void
// Parameters: 2 arrays(string and int)
// Purpose: To update the name and the highscore in
the txt file
public static void UpdateHighScores(String[]
highName, int[] highScore) throws IOException {

```

```

 PrintWriter output = new PrintWriter("highScore.txt");
 for(int i = 0; i < 3; i++) {
 output.println(highName[i] + " " + highScore[i]);
 }
}

```

```

 output.close();
 }

//Method# 7)
 // Return type: void
 // Parameters: 1 int
 // Purpose: To display the user's score in the end.
 public static void displayScore(int Score) {
 JOptionPane.showMessageDialog(null, "Your current Score
is: " + Score + " out of 1000.");
 }
}

```

### Read file:

I. How many bits are in a byte?

2  
7  
12  
8  
4  
100

II. Every complete statement end with a \_\_\_\_

colon  
comma  
semicolon  
period  
3  
100

III. What would be the outcome of this math expression:  $5 * 10 / 2$

35  
25  
30  
10  
2  
100

IV. The &&, ||, and ! are what types of operators?

Logical  
Ternary  
Relational  
Conditional  
1  
100

V. What would be the outcome of  $T || T \&\& F$ ?

True  
False  
||  
&&  
1  
100

VI. This is a variable that keeps a running total.

Sentinal

Sum

Total

Accumulator

4

100

VII. This type of loop always executes at least once. While

Do-while

For

All of the above

2

100

VIII. Which method can be defined only once in a program? main method

finalize method

static method

private method

1

100

IX. In Java language, an array index starts with \_\_\_\_.

[]

0

1

i

2

100

X. If an index of an element is N, what is its actual position in the array? N-1

N++

N+1

N+2

3

100

**Write file:**

jobin 900

900

John 600

### 3) Python Skills (Machine Learning)

```
In [9]: import pandas as pd
```

```
In [10]: df = pd.read_csv("Desktop/COP4612_HW1.csv") In [11]: df.head(10)
```

Out[11]:

```
salary sex pay_grade position age year_at_company 0 51945.03 M GS10 FinancialAnalyst 49 19 1 120282.27 M
GS15 Accountant 20 0 2 73131.81 F GS11 FinancialAnalyst 40 16 3 69548.85 M GS11 Developer 32 4 4 114453.95
M GS13 Accountant 57 21 5 100371.59 F GS10 Secretary 64 40 6 108991.56 M GS13 Developer 35 10 7 83882.52
M GS11 Developer 37 12 8 76271.75 M GS11 Developer 33 5 9 152449.05 M GS15 Accountant 56 14
```

```
In [12]: print(df.keys())
```

```
Index(['salary', 'sex', 'pay_grade', 'position', 'age', 'year_at_company'], dtype='object') In [13]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 3000 entries, 0 to 2999
```

```
Data columns (total 6 columns):
```

```
Column Non-Null Count Dtype
```

```

```

```
0 salary 3000 non-null float64
```

```
1 sex 3000 non-null object
```

```
2 pay_grade 3000 non-null object
```

```
3 position 3000 non-null object
```

```
4 age 3000 non-null int64
```

```
5 year_at_company 3000 non-null int64
```

```
dtypes: float64(1), int64(2), object(3)
```

```
memory usage: 140.8+ KB
```

```
In [14]: print(df.corr())
```

```
salary age year_at_company
```

```
salary 1.000000 0.052545 0.390425
```

```
age 0.052545 1.000000 0.665181
```

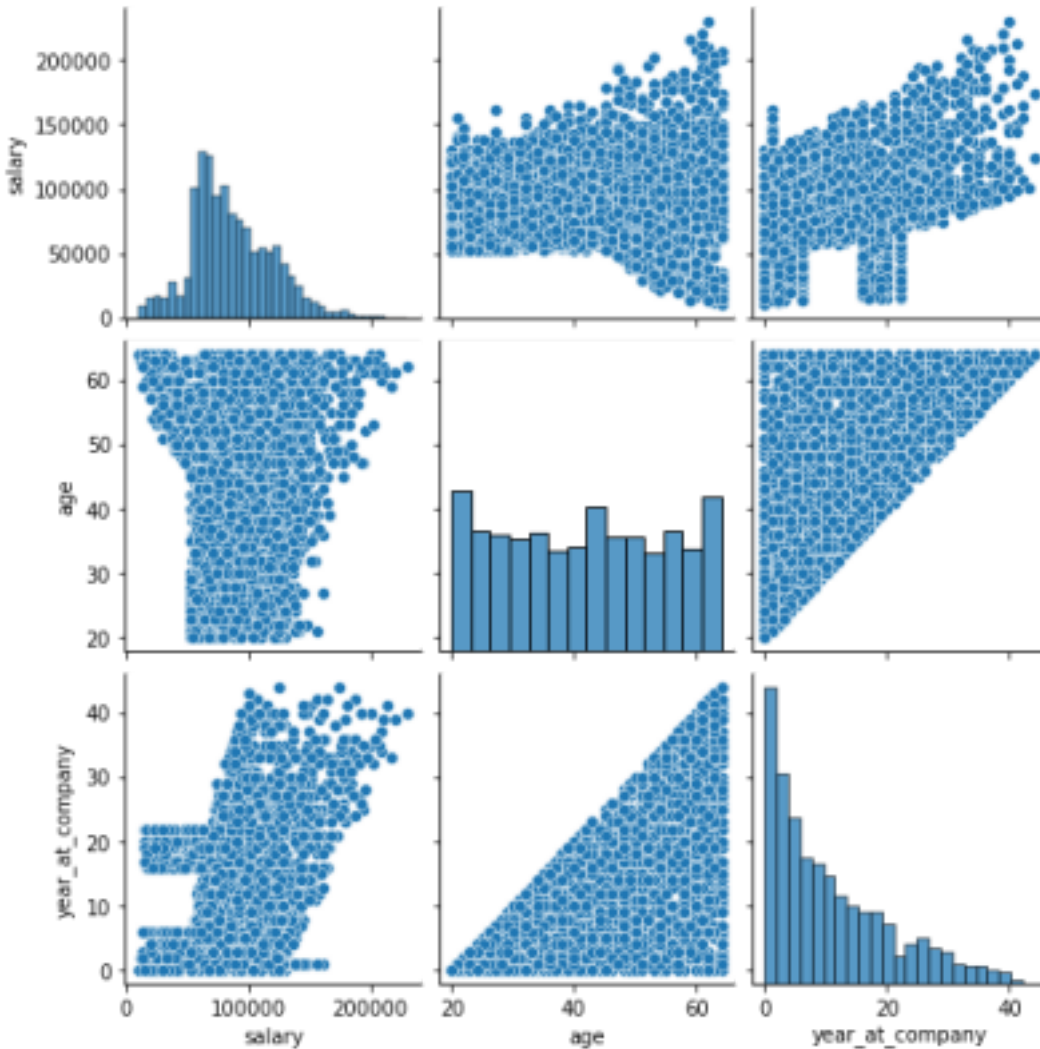
```
year_at_company 0.390425 0.665181 1.000000
```

```
In [15]: import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
sns.pairplot(df)
```

```
plt.show()
```



```
In [16]: dummy = pd.get_dummies(df, columns = ['sex', 'position', 'pay_grade']) print(dummy.keys())
dummy.head(5)
```

```
Index(['salary', 'age', 'year_at_company', 'sex_F', 'sex_M',
 'position_Accountant', 'position_Developer', 'position_Engineer',
 'position_FinancialAnalyst', 'position_Secretary', 'pay_grade_GS10',
 'pay_grade_GS11', 'pay_grade_GS12', 'pay_grade_GS13', 'pay_grade_GS14', 'pay_grade_GS15'],
 dtype='object')
```

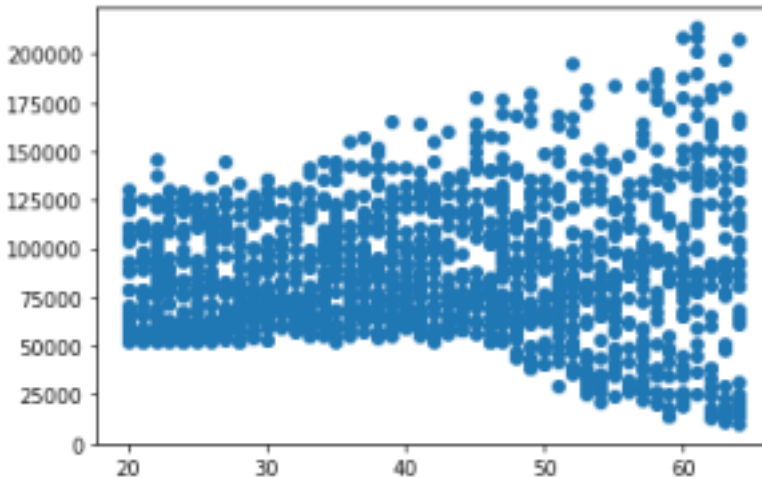
```
Out[16]:
salary age year_at_company sex_F sex_M position_Accountant position_Developer position_Engineer
position_FinancialAnalyst position_Secretary pay_grade_GS10 pay_grade_GS11 pay_grade_GS12
pay_grade_GS13 pay_grade_GS14 pay_grade_GS15 0 51945.03 49 19 0 1 0 0 0 1 0 1 0 0 0 0 0 1 120282.27 20 0
0 1 1 0 0 0 0 0 0 0 0 1 2 73131.81 40 16 1 0 0 0 0 1 0 0 1 0 0 0 0 3 69548.85 32 4 0 1 0 1 0 0 0 0 1 0 0 0 0 4
114453.95 57 21 0 1 1 0 0 0 0 0 0 0 1 0 0
```



```
In [17]: from matplotlib import pyplot as plt

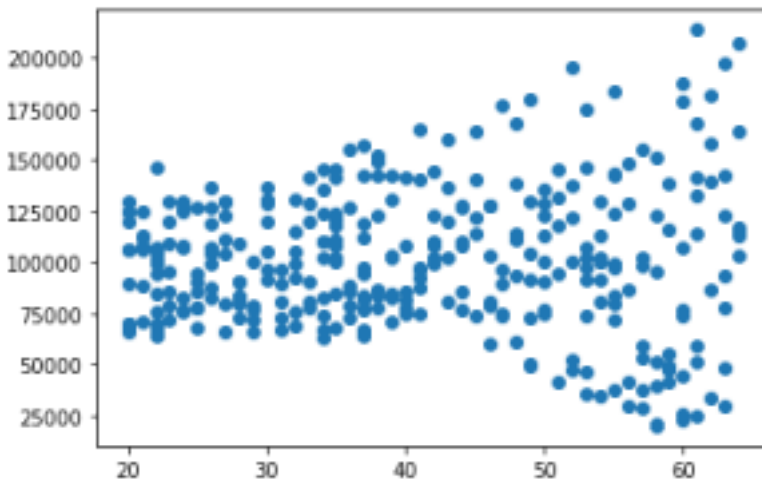
df_sex_F = dummy[dummy['sex_F'] == 1]
plt.scatter(df_sex_F['age'], df_sex_F['salary'])
print("\t\tWomen all positions")
plt.show()
```

Women all positions



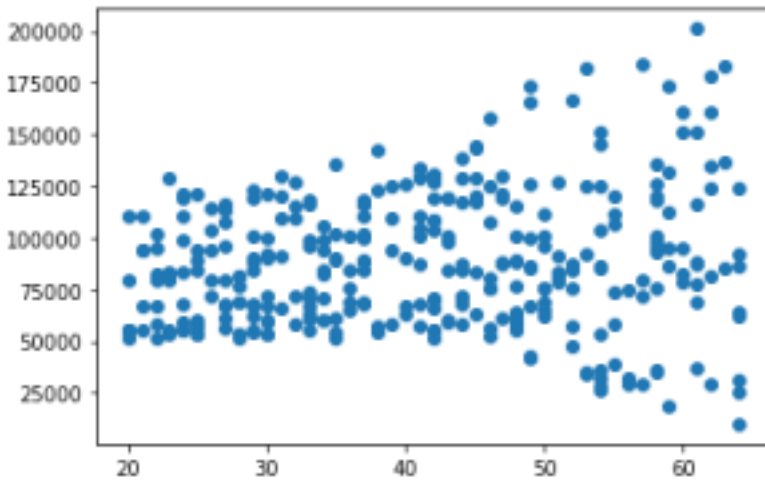
```
In [18]: f_developer = df_sex_F[df_sex_F['position_Developer'] == 1]
plt.scatter(f_developer['age'], f_developer['salary'])
print("\t\tWomen Developer")
plt.show()
```

Women Developer



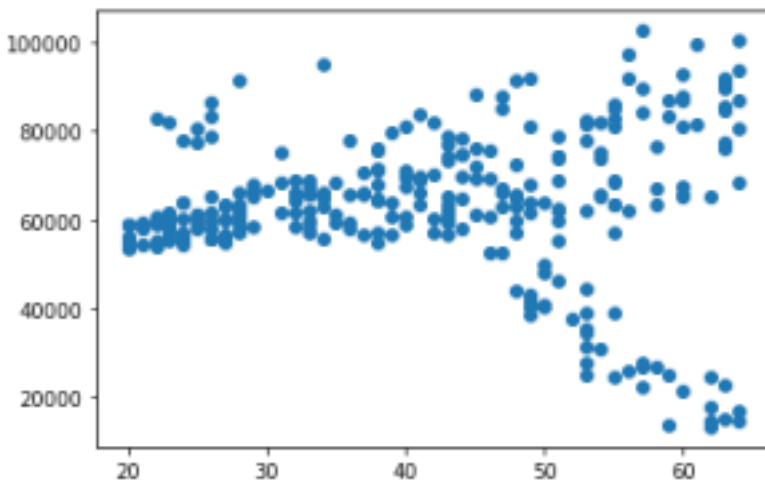
```
In [19]: f_engineer = df_sex_F[df_sex_F['position_Engineer'] == 1]
plt.scatter(f_engineer['age'], f_engineer['salary'])
print("\t\tWomen Engineer")
plt.show()
```

Women Engineer



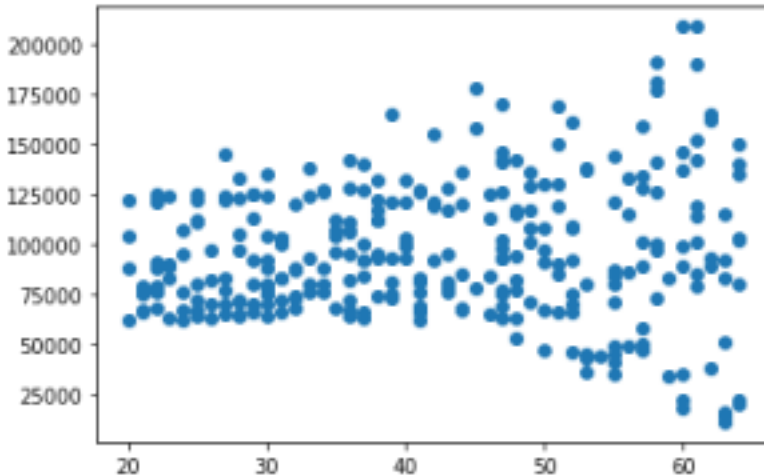
```
In [20]: f_Secretary = df_sex_F[df_sex_F['position_Secretary'] == 1]
plt.scatter(f_Secretary['age'], f_Secretary['salary'])
print("\t\tWomen Secretary")
plt.show()
```

Women Secretary



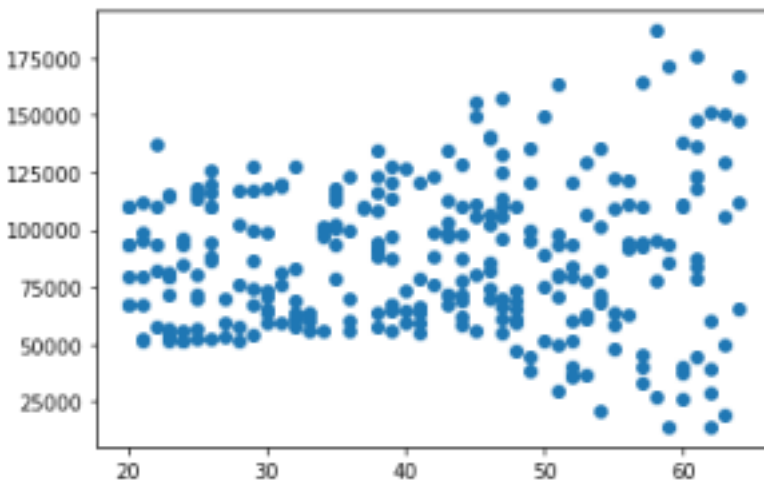
```
In [21]: f_Accountant = df_sex_F[df_sex_F['position_Accountant'] == 1]
plt.scatter(f_Accountant['age'], f_Accountant['salary'])
print("\t\tWomen Accountant")
plt.show()
```

Women Accountant



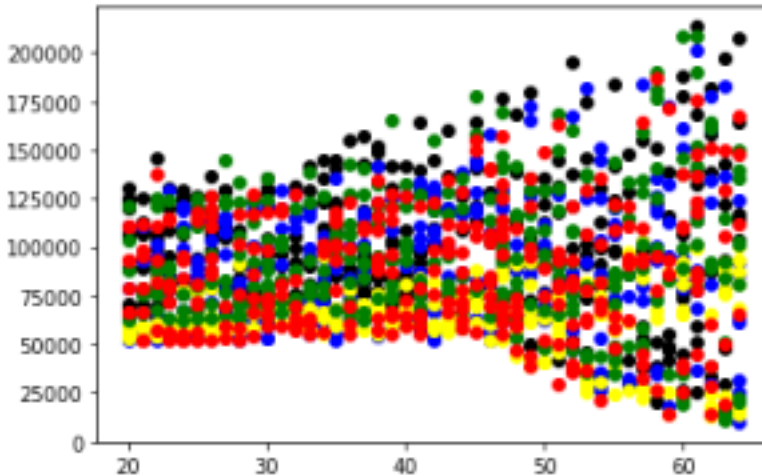
```
In [22]: f_FinancialAnalyst = df_sex_F[df_sex_F['position_FinancialAnalyst'] == 1]
plt.scatter(f_FinancialAnalyst['age'],f_FinancialAnalyst['salary'])
print("\t\tWomen FinancialAnalysts")
plt.show()
```

Women FinancialAnalysts



```
In [23]: plt.scatter(f_developer['age'],f_developer['salary'],color= 'black')
plt.scatter(f_engineer['age'],f_engineer['salary'],color= 'blue')
plt.scatter(f_Secretary['age'],f_Secretary['salary'],color= 'yellow')
plt.scatter(f_Accountant['age'],f_Accountant['salary'],color= 'green')
plt.scatter(f_FinancialAnalyst['age'],f_FinancialAnalyst['salary'],color= 'red')
print("\t\tWomen all positions\n\t\tdeveloper = black\n\t\tengineer = blue\n\t\tSecretary = yellow\n\t\tAccountant = green\n\t\tFinancialAnalyst = red") plt.show()
```

Women all positions  
 developer = black  
 engineer = blue  
 Secretary = yellow  
 Accountant = green  
 FinancialAnalyst = red



```
In [31]: women_stat_df = pd.DataFrame(columns=['Position','Count','Min Salary', 'Max Salary', 'Mean Salary','Std Dev Salary', 'Avg Age'])
```

```
In [34]: all_positions = [df_sex_F,f_developer,f_engineer,f_Secretary,f_Accountant,f_FinancialAnalyst]
all_position_titles = ['All', 'Developer', 'Engineer','FinancialAnalyst','Secretary','Accountant']
for i,data in enumerate(all_positions):
 position = all_position_titles[i]
 count = data.count()[0]
 min_salary = round(data['salary'].min(),4)
 max_salary = round(data['salary'].max(),4)
 mean_salary = round(data['salary'].mean(),4)
 std_dev_salary = data['salary'].std()
 avg_age = data['age'].mean()
 row = [position,count,min_salary,max_salary,mean_salary,std_dev_salary,avg_age]
 women_stat_df.loc[i] = row
women_stat_df
```

Out[34]:

|   | Position         | Count | Min Salary | Max Salary | Mean Salary | Std Dev Salary | Avg Age   |
|---|------------------|-------|------------|------------|-------------|----------------|-----------|
| 0 | All              | 1533  | 10086.06   | 213617.05  | 87549.9852  | 33495.5531     | 41.417482 |
| 1 | Developer        | 326   | 20152.53   | 213617.05  | 100963.2651 | 34717.4438     | 41.052147 |
| 2 | Engineer         | 329   | 10086.06   | 201232.58  | 88267.1785  | 32627.2545     | 41.145897 |
| 3 | FinancialAnalyst | 291   | 13411.31   | 102608.67  | 63296.3647  | 16671.1745     | 41.512027 |
| 4 | Secretary        | 300   | 11410.88   | 208665.88  | 96069.2654  | 34256.2358     | 41.923333 |
| 5 | Accountant       | 287   | 13991.54   | 186865.19  | 87178.3256  | 32097.6774     | 41.519164 |

```
In [39]: def remove_outliers(df,columns,n_std):
for col in columns:
 print('column: {}'.format(col))

 mean = df[col].mean()
 sd = df[col].std()

 df = df[(df[col] <= mean+(n_std*sd))]
```

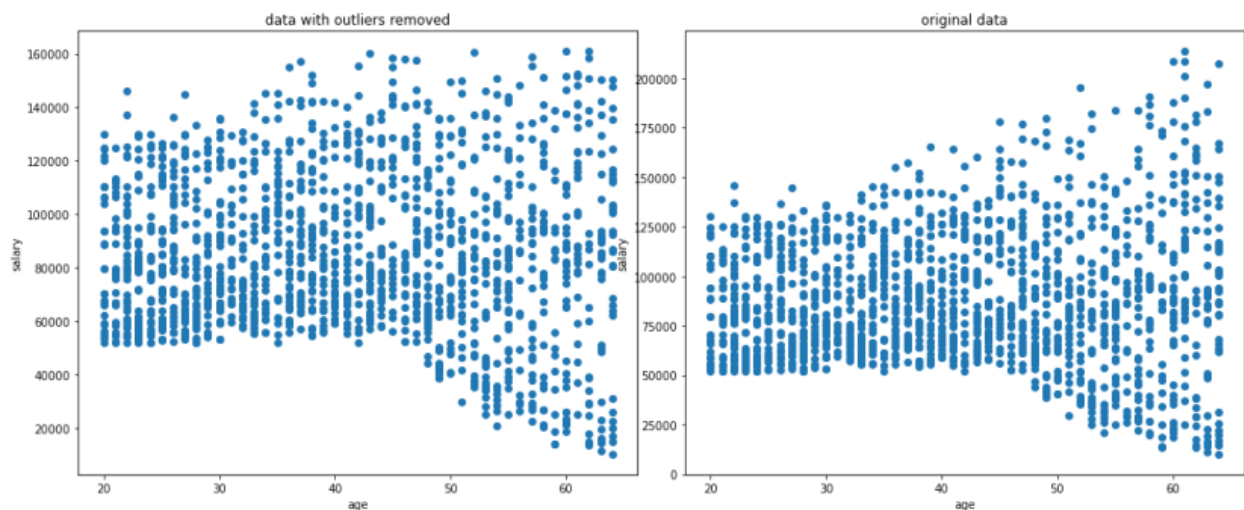
```
return df
```

```
In [42]: df = remove_outliers(df_sex_F,['salary','age'],2.2)
pos = [df_sex_F,f_developer,f_engineer,f_Secretary,f_Accountant,f_FinancialAnalyst] without = []
for data in pos:
 without.append(remove_outliers(data,['salary'],2.2))
```

```
column: salary
column: age
column: salary
column: salary
column: salary
column: salary
column: salary
column: salary
```

```
In [43]: fig,axes = plt.subplots(nrows = 1,ncols = 2, figsize = (15,6)) fig.tight_layout()
axes[0].scatter(df['age'],df['salary'])
axes[1].scatter(df_sex_F['age'],df_sex_F['salary'])
axes[0].set_title('data with outliers removed')
axes[1].set_title('original data')
axes[0].set_xlabel('age')
axes[1].set_xlabel('age')
axes[0].set_ylabel('salary')
axes[1].set_ylabel('salary')
```

```
Out[43]:
Text(564.545454545444, 0.5, 'salary')
```



```
In [74]: import numpy as np
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
def training_automation(data_source):
```

```

report_df = pd.DataFrame(columns =
['Train','Test','Test_R_Score','Test_RMSE','Train_R_Score','Train_RMSE','Model_Var','Model_Error','Avg_
New_Salary'])
df

model = LinearRegression()

x_data = data_source[['age']]
y_data = data_source['salary']

sizes = np.arange(0.2, 1.0, 0.2)

for size in sizes:
 x_train, x_test, y_train, y_test = train_test_split(x_data,y_data,shuffle = True, train_size = size)

 model.fit(x_train,y_train)

 predictions_test = model.predict(x_test)

 error_test = y_test - predictions_test

 r2_test = r2_score(y_test, predictions_test)
 rmse_test = mean_squared_error(y_test, predictions_test, squared = True)

 predictions_train = model.predict(x_train)
 r2_train = r2_score(y_train, predictions_train)
 rmse_train = mean_squared_error(y_train, predictions_train, squared = True)

 error_train = y_train - predictions_train

 model_variance = (rmse_test / y_test.size) / (rmse_train / y_train.size)

 error_mean = error_test.mean()
 average_salary = round(model.intercept_,2)
 report_df.loc[len(report_df.index)] = {'Train': size,
'Test': 1-size,
'Test_R_Score':r2_test,
'Test_RMSE': rmse_test,
'Train_R_Score': r2_train,
'Train_RMSE': rmse_train,
'Model_Var': model_variance,
'Model_Error': error_mean,
'Avg_New_Salary': average_salary}

return report_df

```

```

In [78]: report = training_automation(df)
print(f"Women Positions: ALL")
report

```

Women Positions: ALL

Out[78]:

| Train | Test | Test_R_Score | Test_RMSE    | Train_R_Score | Train_RMSE   | Model_Var | Model_Error | Avg_New_Salary | 0 |
|-------|------|--------------|--------------|---------------|--------------|-----------|-------------|----------------|---|
| 0.2   | 0.8  | -0.022137    | 9.228750e+08 | 0.022704      | 8.609070e+08 | 0.267995  | 1306.339947 | 98140.03       |   |

```
In [79]: report = training_automation(without[0]) print(f"Women Positions: {all_position_titles[0]}") report
```

Women Positions: All

Out[79]:

| Train | Test | Test_R_Score | Test_RMSE    | Train_R_Score | Train_RMSE   | Model_Var | Model_Error  | Avg_New_Salary | 0 |
|-------|------|--------------|--------------|---------------|--------------|-----------|--------------|----------------|---|
| 0.2   | 0.8  | -0.0085      | 8.945794e+08 | 0.000087      | 9.389193e+08 | 0.238194  | -2853.486418 | 88084.9        |   |

```
In [81]: report = training_automation(without[1]) print(f"Women Positions: {all_position_titles[1]}") report
```

Women Positions: Developer

Out[81]:

| Train | Test | Test_R_Score | Test_RMSE    | Train_R_Score | Train_RMSE   | Model_Var | Model_Error | Avg_New_Salary | 0 |
|-------|------|--------------|--------------|---------------|--------------|-----------|-------------|----------------|---|
| 0.2   | 0.8  | 0.007357     | 1.035211e+09 | 0.004833      | 7.814620e+08 | 0.32857   | 630.689695  | 103968.27      |   |

```
In [82]: report = training_automation(without[2]) print(f"Women Positions: {all_position_titles[2]}") report
```

Women Positions: Engineer

Out[82]:

| Train | Test | Test_R_Score | Test_RMSE    | Train_R_Score | Train_RMSE   | Model_Var | Model_Error  | Avg_New_Salary | 0 |
|-------|------|--------------|--------------|---------------|--------------|-----------|--------------|----------------|---|
| 0.2   | 0.8  | -0.015492    | 8.370375e+08 | 0.01347       | 7.964561e+08 | 0.259647  | -2443.276054 | 76551.8        |   |

```
In [83]: report = training_automation(without[3]) print(f"Women Positions: {all_position_titles[3]}") report
```

Women Positions: FinancialAnalayst

Out[83]:

| Train | Test | Test_R_Score | Test_RMSE    | Train_R_Score | Train_RMSE   | Model_Var | Model_Error | Avg_New_Salary | 0 |
|-------|------|--------------|--------------|---------------|--------------|-----------|-------------|----------------|---|
| 0.2   | 0.8  | -0.001464    | 2.901505e+08 | 0.000502      | 1.831081e+08 | 0.389316  | 10.82996    | 62010.22       |   |

```
In [84]: report = training_automation(without[4]) print(f"Women Positions: {all_position_titles[4]}") report
```

Women Positions: Secretary

Out[84]:

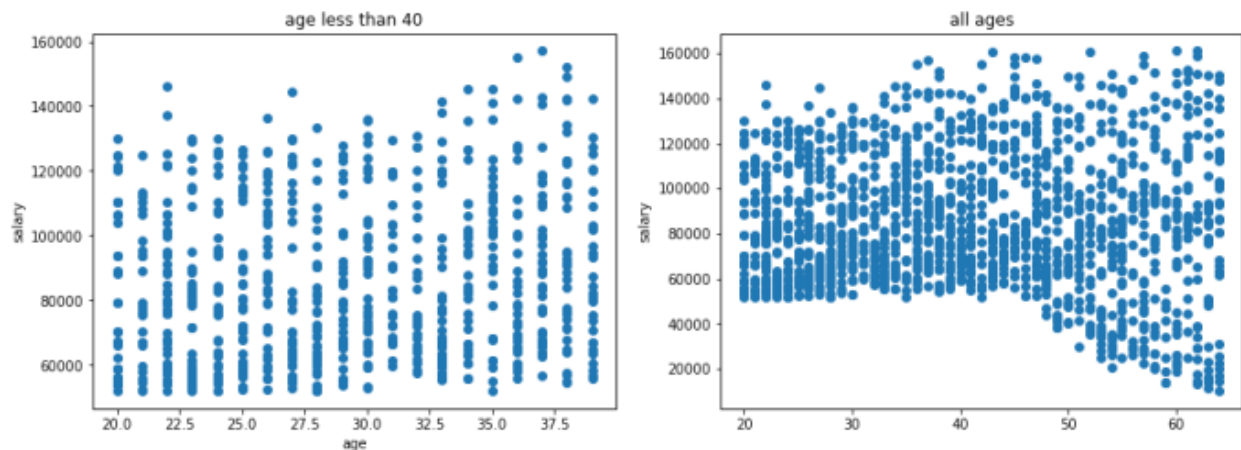
| Train | Test | Test_R_Score | Test_RMSE    | Train_R_Score | Train_RMSE   | Model_Var | Model_Error  | Avg_New_Salary | 0 |
|-------|------|--------------|--------------|---------------|--------------|-----------|--------------|----------------|---|
| 0.2   | 0.8  | -0.026481    | 1.054587e+09 | 0.005048      | 7.445240e+08 | 0.349594  | -4464.534781 | 91157.82       |   |

```
In [85]: less_40 = df[df['age'] < 40]
```

```
In [87]: fig, axes = plt.subplots(1, 2, figsize = (15, 5)) axes[0].scatter(less_40['age'], less_40['salary'])
axes[0].set_title('age less than 40')
axes[0].set_xlabel('age')
axes[0].set_ylabel('salary')

axes[1].scatter(df['age'], df['salary'])
axes[1].set_title('all ages')
axes[1].set_xlabel('age')
axes[1].set_ylabel('salary')
```

Out[87]:  
Text(0, 0.5, 'salary')



```
In [88]: report = training_automation(less_40) print(f"Womden Positions: All")
report
```

Womden Positions: All

```
Out[88]:
Train Test Test_R_Score Test_RMSE Train_R_Score Train_RMSE Model_Var Model_Error Avg_New_Salary 0
0.2 0.8 0.020059 6.109484e+08 0.02131 5.818316e+08 0.260649 1473.824834 66235.63
```

```
In [91]: from sklearn.linear_model import LinearRegression
```

```
from sklearn.model_selection import train_test_split
```

```
model = LinearRegression()
```

```
x_data = less_40[['age']]
y_data = less_40['salary']
```

```
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, shuffle = True, train_size = 0.4)
model.fit(x_train, y_train)
```



```

from sklearn.metrics import mean_squared_error, r2_score
predictions_test = model.predict(x_test)
error_test = y_test - predictions_test

r2_test = r2_score(y_test, predictions_test)
rmse_test = mean_squared_error(y_test, predictions_test, squared = True)

predictions_train = model.predict(x_train)
r2_train = r2_score(y_train, predictions_train)
rmse_train = mean_squared_error(y_train, predictions_train, squared = True)

error_train = y_train - predictions_train

model_variance = (rmse_test / y_test.size) / (rmse_train / y_train.size)

print(f"price = {model.intercept_} + {model.coef_[0]} mileage + error", end = "\n\n")

print('Test r_score is:\t', r2_test)
print('Test rmse is:\t\t', rmse_test)
print()
print(f"Training-Test Split:\t 0.7 training 0.3 test")
print()
print('Training r_score is:\t', r2_train)
print('Training rmse is:\t', rmse_train)
print('Model Variance:\t\t', model_variance)
print()
print('Mean of Test Error:\t', error_test.mean())
print('Mean of Train Error:\t', error_train.mean())

print()

print('Average salary of women < 40 years of age: ', round(model.intercept_, 2))

plt.subplot(1,2, 1)
plt.title('Test Error Density')

remove scientific notation from axis
plt.ticklabel_format(style='plain')

my_kde = sns.kdeplot(error_test)

x,y = my_kde.lines[0].get_data()

vlines(x, ymin, ymax, colors, linestyle)
plt.vlines(error_test.mean(), 0, y.max(), color='Black', ls=':')

plt.subplot(1,2, 2)
plt.title('Test Error')
plt.scatter(range(error_test.size), error_test)

```

```
plt.show()
```

price = 77989.08534645132 + 216.50859539742655 mileage + error

Test r\_score is: 0.010776854219104015

Test rmse is: 642648058.2053282

Training-Test Split: 0.7 training 0.3 test

Training r\_score is: 0.002702411732220078

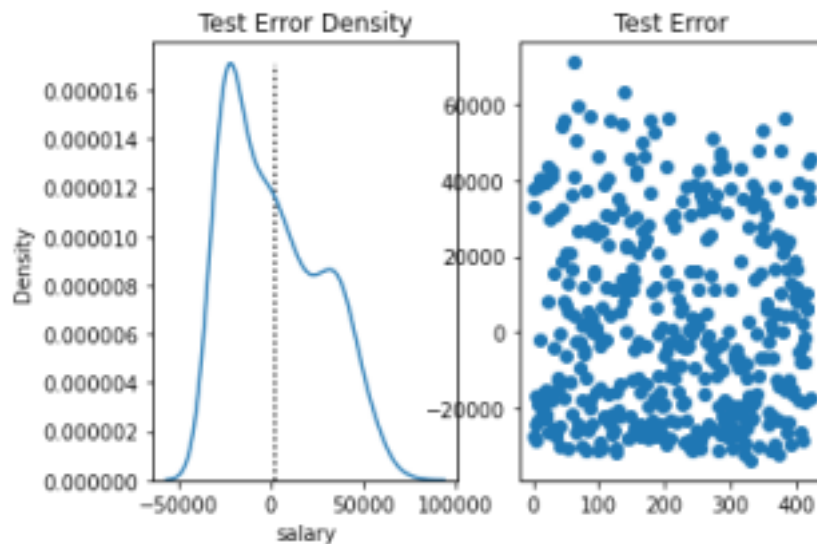
Training rmse is: 565853966.4475682

Model Variance: 0.7544575130130522

Mean of Test Error: 2276.7539488075454

Mean of Train Error: -1.3982267301277757e-12

Average salary of women < 40 years of age: 77989.09



```
In []:
```

```
In [1]: import pandas as pd
```

```
In [2]: Cancer_Data = pd.read_csv("Desktop/Cancer_Data.csv") Cancer_Data.head(5)
```

```
Out[2]:
```

```
id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
points_mean ... radius_worst texture_worst perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst concave
points_worst sy
0 842302 M 17.99 10.38 122.8 1001.0 0.11840 0.27760 0.3001 0.14710 ... 25.38 17.33 184.60 2019.0 0.1622 0.6656
0.7119 0.2654 1 842517 M 20.57 17.77 132.9 1326.0 0.08474 0.07864 0.0869 0.07017 ... 24.99 23.41 158.80 1956.0
0.1238 0.1866 0.2416 0.1860 2 84300903 M 19.69 21.25 130.0 1203.0 0.10960 0.15990 0.1974 0.12790 ... 23.57
25.53 152.50 1709.0 0.1444 0.4245 0.4504 0.2430 3 84348301 M 11.42 20.38 NaN 386.1 0.14250 NaN 0.2414
```

```
0.10520 ... 14.91 26.50 98.87 567.7 0.2098 0.8663 0.6869 0.2575 4 84358402 M 20.29 14.34 135.1 1297.0 0.10030
0.13280 0.1980 0.10430 ... 22.54 16.67 152.20 1575.0 0.1374 0.2050 0.4000 0.1625
```

5 rows × 32 columns

```
In [3]: Cancer_Data.describe()
```

Out[3]:

```
id radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
points_mean symmetry_mean ... radius_worst texture_worst perimeter_worst area_worst smoothness_worst
compactness_worst concavity_worst

count 5.690000e+02 565.000000 564.000000 565.000000 565.000000 568.000000 564.000000 556.000000
558.000000 567.000000 ... 567.000000 568.000000 567.000000 567.000000 434.000000 567.000000 556.000000

mean 3.037183e+07 14.123581 19.286436 92.012319 655.822655 0.096316 0.103982 0.089747 0.049576
0.181207 ... 16.259310 25.666585 107.299083 878.223633 0.132375 0.254414 0.276891 std 1.250206e+08
3.515269 4.313091 24.219501 352.848451 0.014037 0.052455 0.079679 0.038692 0.027430 ... 4.837704 6.146429
33.643536 568.959490 0.023267 0.157471 0.207402 min 8.670000e+03 6.981000 9.710000 43.790000 143.500000
0.052630 0.019380 0.000000 0.000000 0.106000 ... 7.930000 12.020000 50.410000 185.200000 0.071170 0.027290
0.000000 25% 8.692180e+05 11.700000 16.167500 75.210000 420.300000 0.086290 0.064315 0.029930 0.020692
0.161950 ... 13.010000 21.075000 84.135000 514.650000 0.115400 0.147450 0.120975 50% 9.060240e+05
13.340000 18.835000 86.340000 551.100000 0.095865 0.092350 0.061880 0.033950 0.179200 ... 14.960000
25.405000 97.660000 684.600000 0.131350 0.211900 0.230050 75% 8.813129e+06 15.780000 21.802500
104.100000 788.500000 0.105300 0.130400 0.131950 0.074122 0.195700 ... 18.775000 29.675000 125.650000
1060.000000 0.147175 0.339500 0.385500 max 9.113205e+08 28.110000 39.280000 188.500000 2501.000000
0.163400 0.345400 0.426800 0.201200 0.304000 ... 36.040000 49.540000 251.200000 4254.000000 0.218400
1.058000 1.252000
```

8 rows × 31 columns

```
In [4]: Cancer_Data.keys()
```

Out[4]:

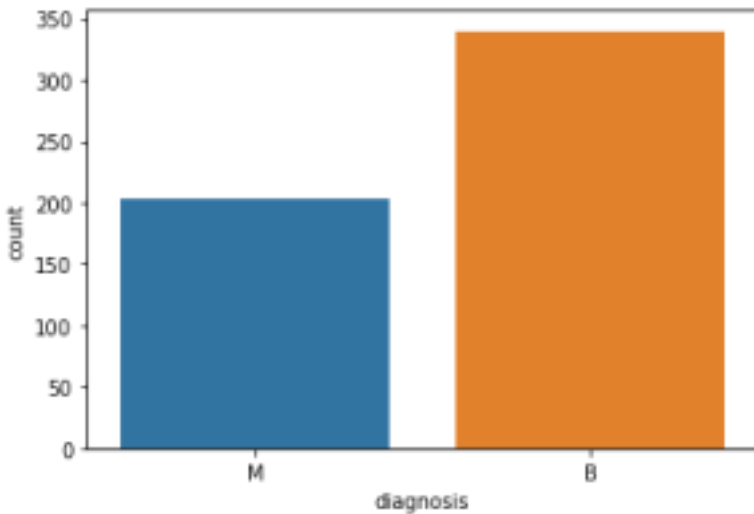
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean',
'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se',
'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst',
'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst',
'fractal_dimension_worst'],
```

```
dtype='object')
```

```
In [5]: nRow, nCol = Cancer_Data.shape
print(f'There are {nRow} rows and {nCol} columns')
```

There are 569 rows and 32 columns

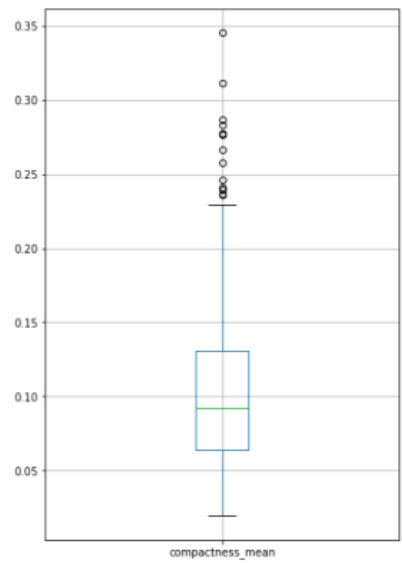
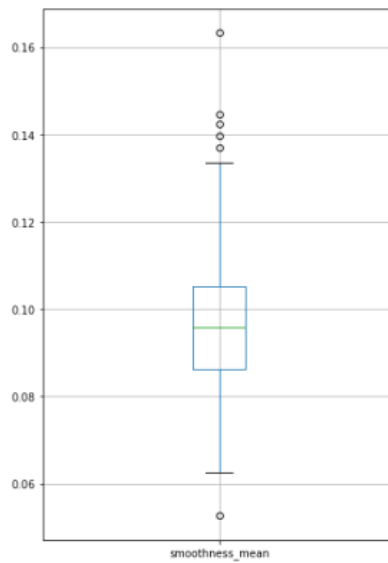
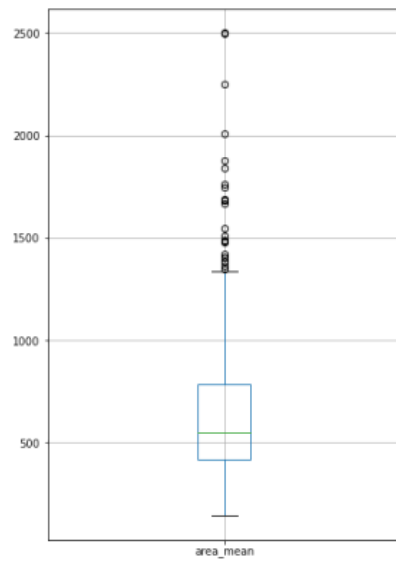
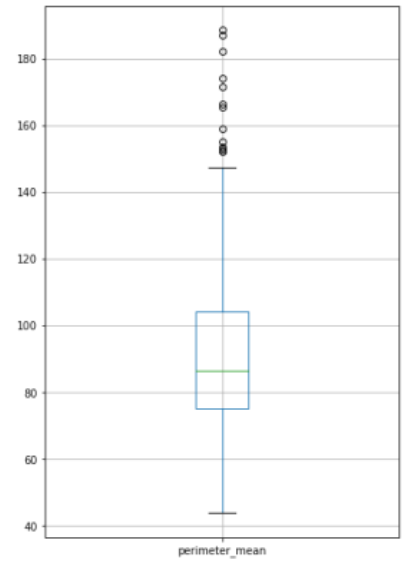
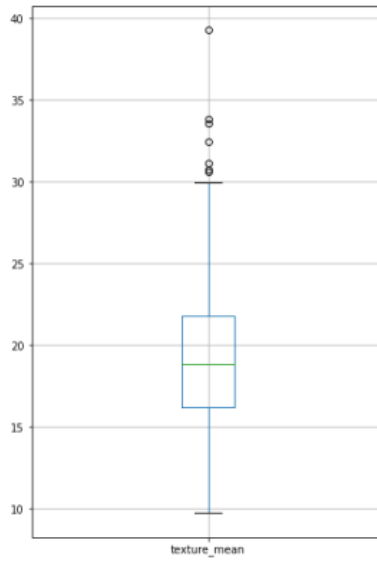
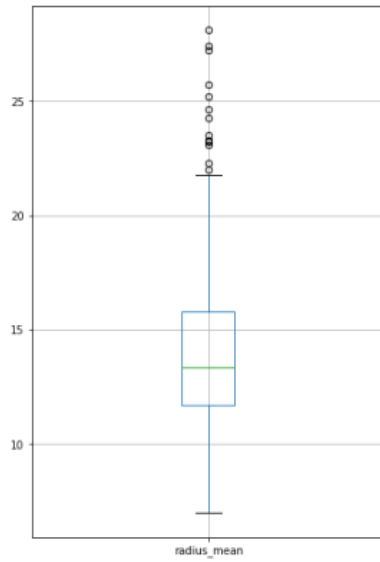
```
In [6]: import seaborn as sns
import matplotlib.pyplot as plt
df2 = sns.countplot(x = 'diagnosis', data = Cancer_Data)
```



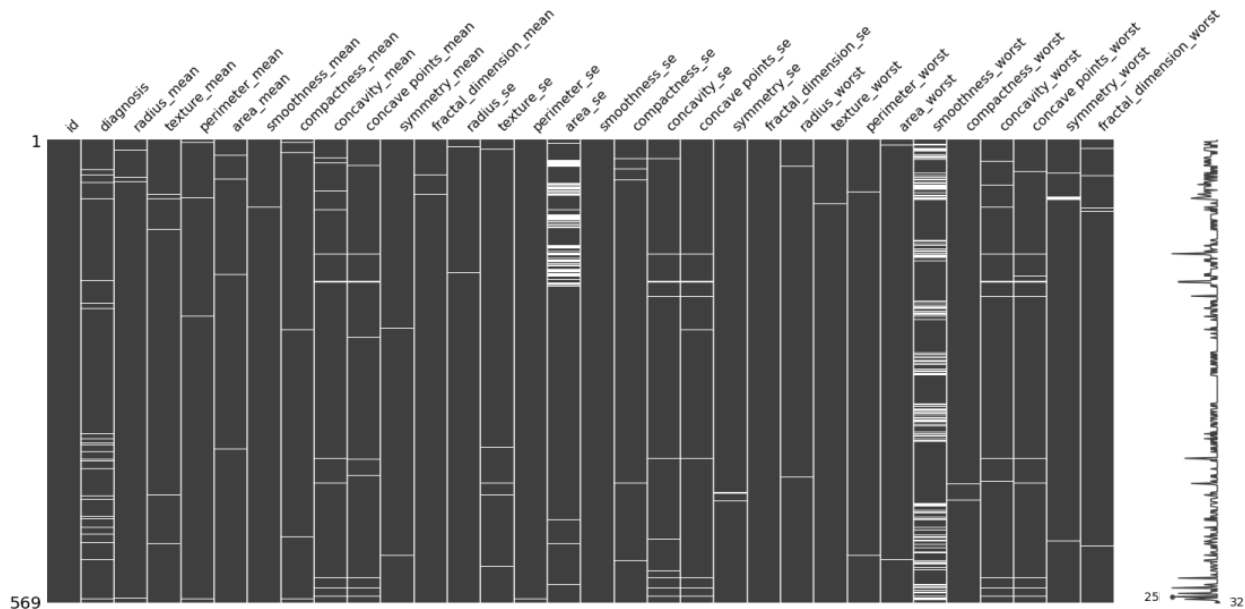
```
In [7]: fig, axes = plt.subplots(2, 3, figsize=(20, 20)) Cancer_Data.boxplot(ax=axes[0, 0], column=
'radius_mean') Cancer_Data.boxplot(ax=axes[0, 1], column='texture_mean')
Cancer_Data.boxplot(ax=axes[0, 2], column='perimeter_mean') Cancer_Data.boxplot(ax=axes[1, 0],
column='area_mean') Cancer_Data.boxplot(ax=axes[1, 1], column='smoothness_mean')
Cancer_Data.boxplot(ax=axes[1, 2], column='compactness_mean')
```

<AxesSubplot:>

Out[7]:



```
In [8]: import missingno as msno
msno.matrix(Cancer_Data)
plt.show()
```



```
In [9]: columns_with_missing_data = [col for col in Cancer_Data.columns if Cancer_Data[col].isnull().sum()
> 0]
columns_with_missing_data
```

```
Out[9]:
['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean', 'symmetry_mean',
 'fractal_dimension_mean', 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'compactness_se',
 'concavity_se',
 'concave points_se', 'symmetry_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst', 'concavity_worst',
 'concave points_worst', 'symmetry_worst',
 'fractal_dimension_worst']
```

```
In [10]: Cancer_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
```

Data columns (total 32 columns):

# Column Non-Null Count Dtype

```

0 id 569 non-null int64
1 diagnosis 543 non-null object
2 radius_mean 565 non-null float64
3 texture_mean 564 non-null float64
4 perimeter_mean 565 non-null float64
5 area_mean 565 non-null float64
6 smoothness_mean 568 non-null float64
7 compactness_mean 564 non-null float64
8 concavity_mean 556 non-null float64
9 concave points_mean 558 non-null float64
10 symmetry_mean 567 non-null float64
11 fractal_dimension_mean 567 non-null float64
12 radius_se 567 non-null float64
13 texture_se 564 non-null float64
14 perimeter_se 568 non-null float64
15 area_se 501 non-null float64
16 smoothness_se 569 non-null float64
17 compactness_se 564 non-null float64
18 concavity_se 557 non-null float64
19 concave points_se 559 non-null float64
20 symmetry_se 565 non-null float64
21 fractal_dimension_se 569 non-null float64
22 radius_worst 567 non-null float64
23 texture_worst 568 non-null float64
24 perimeter_worst 567 non-null float64
25 area_worst 567 non-null float64
26 smoothness_worst 434 non-null float64
27 compactness_worst 567 non-null float64
28 concavity_worst 556 non-null float64
29 concave points_worst 557 non-null float64
30 symmetry_worst 563 non-null float64
31 fractal_dimension_worst 562 non-null float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

```
In [11]: def get_numerical_summary(Cancer_Data):
total = Cancer_Data.shape[0]
columns_with_missing_data = [col for col in Cancer_Data.columns if Cancer_Data[col].isnull().sum() >
0] missing_percent = {}
```

```
for col in columns_with_missing_data:
 null_count = Cancer_Data[col].isnull().sum()
 per = (null_count/total) * 100
 missing_percent[col] = per
 print("{} : {} ({}%)".format(col, null_count, round(per, 3)))
return missing_percent
```

```
missing_percent = get_numerical_summary(Cancer_Data)
```

```
diagnosis : 26 (4.569%)
radius_mean : 4 (0.703%)
texture_mean : 5 (0.879%)
```

```
perimeter_mean : 4 (0.703%)
area_mean : 4 (0.703%)
smoothness_mean : 1 (0.176%)
compactness_mean : 5 (0.879%)
concavity_mean : 13 (2.285%)
concave points_mean : 11 (1.933%)
symmetry_mean : 2 (0.351%)
fractal_dimension_mean : 2 (0.351%)
radius_se : 2 (0.351%)
texture_se : 5 (0.879%)
perimeter_se : 1 (0.176%)
area_se : 68 (11.951%)
compactness_se : 5 (0.879%)
concavity_se : 12 (2.109%)
concave points_se : 10 (1.757%)
symmetry_se : 4 (0.703%)
radius_worst : 2 (0.351%)
texture_worst : 1 (0.176%)
perimeter_worst : 2 (0.351%)
area_worst : 2 (0.351%)
smoothness_worst : 135 (23.726%)
compactness_worst : 2 (0.351%)
concavity_worst : 13 (2.285%)
concave points_worst : 12 (2.109%)
symmetry_worst : 6 (1.054%)
fractal_dimension_worst : 7 (1.23%)
```

In [12]: #11

```
def drop_feature(data_frame, threshold):

 for col, per in missing_percent.items():
 if per > threshold:
 data_frame.drop(col, axis = 1, inplace = True)

 return data_frame

df3 = drop_feature(Cancer_Data.copy(), 22)

_ = get_numerical_summary(df3)
```

```
diagnosis : 26 (4.569%)
radius_mean : 4 (0.703%)
texture_mean : 5 (0.879%)
perimeter_mean : 4 (0.703%)
area_mean : 4 (0.703%)
smoothness_mean : 1 (0.176%)
compactness_mean : 5 (0.879%)
concavity_mean : 13 (2.285%)
concave points_mean : 11 (1.933%)
symmetry_mean : 2 (0.351%)
fractal_dimension_mean : 2 (0.351%)
radius_se : 2 (0.351%)
texture_se : 5 (0.879%)
perimeter_se : 1 (0.176%)
area_se : 68 (11.951%)
```



```
compactness_se : 5 (0.879%)
concavity_se : 12 (2.109%)
concave points_se : 10 (1.757%)
symmetry_se : 4 (0.703%)
radius_worst : 2 (0.351%)
texture_worst : 1 (0.176%)
perimeter_worst : 2 (0.351%)
area_worst : 2 (0.351%)
compactness_worst : 2 (0.351%)
concavity_worst : 13 (2.285%)
concave points_worst : 12 (2.109%)
symmetry_worst : 6 (1.054%)
fractal_dimension_worst : 7 (1.23%)
```

In [13]: #11

```
def drop_row(Cancer_Data, threshold):
 # Getting Missing count of each sample
 for idx in range(Cancer_Data.shape[0]):
 Cancer_Data.loc[idx, 'missing_count'] = Cancer_Data.iloc[idx, :].isnull().sum()
```

```
Cancer_Data.drop(Cancer_Data[Cancer_Data['missing_count'] > threshold].index, axis = 0, inplace =
True)
return Cancer_Data
```

```
print("Samples Before Removal : {}".format(df3.shape[0]))
```

```
this can take some time... depend on the shape of the dataframe
df3 = drop_row(df3, 1)
```

```
print("Samples After Removal : {}".format(df3.shape[0]))
```

```
Samples Before Removal : 569
Samples After Removal : 413
```

In [14]: df3.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 568
Data columns (total 32 columns):
Column Non-Null Count Dtype

0 id 413 non-null int64
1 diagnosis 413 non-null object
2 radius_mean 413 non-null float64
3 texture_mean 413 non-null float64
4 perimeter_mean 413 non-null float64
5 area_mean 413 non-null float64
6 smoothness_mean 413 non-null float64
7 compactness_mean 413 non-null float64
8 concavity_mean 413 non-null float64
9 concave points_mean 413 non-null float64
10 symmetry_mean 413 non-null float64
11 fractal_dimension_mean 413 non-null float64
```

```

12 radius_se 413 non-null float64
13 texture_se 413 non-null float64
14 perimeter_se 413 non-null float64
15 area_se 413 non-null float64
16 smoothness_se 413 non-null float64
17 compactness_se 413 non-null float64
18 concavity_se 413 non-null float64
19 concave points_se 413 non-null float64
20 symmetry_se 413 non-null float64
21 fractal_dimension_se 413 non-null float64
22 radius_worst 413 non-null float64
23 texture_worst 413 non-null float64
24 perimeter_worst 413 non-null float64
25 area_worst 413 non-null float64
26 compactness_worst 413 non-null float64
27 concavity_worst 413 non-null float64
28 concave points_worst 413 non-null float64
29 symmetry_worst 413 non-null float64
30 fractal_dimension_worst 413 non-null float64
31 missing_count 413 non-null float64
dtypes: float64(30), int64(1), object(1)
memory usage: 106.5+ KB

```

In [15]: #13

```

from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import LabelEncoder

```

```
lb = LabelEncoder()
```

```

cat_cols = [col for col in df3.columns if df3[col].dtype == 'object']
for col in cat_cols:
 df3[col] = lb.fit_transform(df3[col])
df3.head()

```

Out[15]:

```

id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
points_mean ... radius_worst texture_worst perimeter_worst area_worst compactness_worst concavity_worst concave
points_worst symmetry_worst fract

0 842302 1 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.30010 0.14710 ... 25.38 17.33 184.6 2019.0 0.6656
0.7119 0.2654 0.4601 1 842517 1 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.08690 0.07017 ... 24.99 23.41
158.8 1956.0 0.1866 0.2416 0.1860 0.2750 2 84300903 1 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.19740
0.12790 ... 23.57 25.53 152.5 1709.0 0.4245 0.4504 0.2430 0.3613 5 843786 1 12.45 15.70 82.57 477.1 0.12780
0.17000 0.15780 0.08089 ... 15.47 23.75 103.4 741.6 0.5249 0.5355 0.1741 0.3985 7 84458202 1 13.71 20.83 90.20
577.9 0.11890 0.16450 0.09366 0.05985 ... 17.06 28.14 110.6 897.0 0.3682 0.2678 0.1556 0.3196

```

5 rows × 32 columns

```
In [16]: df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 568
Data columns (total 32 columns):
Column Non-Null Count Dtype

0 id 413 non-null int64
1 diagnosis 413 non-null int32
2 radius_mean 413 non-null float64
3 texture_mean 413 non-null float64
4 perimeter_mean 413 non-null float64
5 area_mean 413 non-null float64
6 smoothness_mean 413 non-null float64
7 compactness_mean 413 non-null float64
8 concavity_mean 413 non-null float64
9 concave points_mean 413 non-null float64
10 symmetry_mean 413 non-null float64
11 fractal_dimension_mean 413 non-null float64
12 radius_se 413 non-null float64
13 texture_se 413 non-null float64
14 perimeter_se 413 non-null float64
15 area_se 413 non-null float64
16 smoothness_se 413 non-null float64
17 compactness_se 413 non-null float64
18 concavity_se 413 non-null float64
19 concave points_se 413 non-null float64
20 symmetry_se 413 non-null float64
21 fractal_dimension_se 413 non-null float64
22 radius_worst 413 non-null float64
23 texture_worst 413 non-null float64
24 perimeter_worst 413 non-null float64
25 area_worst 413 non-null float64
26 compactness_worst 413 non-null float64
27 concavity_worst 413 non-null float64
28 concave points_worst 413 non-null float64
29 symmetry_worst 413 non-null float64
30 fractal_dimension_worst 413 non-null float64
31 missing_count 413 non-null float64
dtypes: float64(30), int32(1), int64(1)
memory usage: 104.9 KB
```

```
In [17]: #15/16
```

```
import numpy as np
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

```
y = df3.iloc[:, 1]
x = df3.iloc[:, 2:31]
```

```
def lr_automation(x,y):
 report_dataframe = pd.DataFrame(columns = ['Test_Size',
 'Train_Accuracy_Score',
```

```

'Test_Accuracy_Score',
'Normalized_Score'])
lr_model = LogisticRegression(max_iter = 10000)

sizes = np.arange(0.1,1.0,0.1)

for size in sizes:

 x_train, x_test, y_train, y_test = train_test_split(x, y, shuffle = True, test_size = size)

 lr_model.fit(x_train, y_train)

 lr_train_predictions = lr_model.predict(x_train)
 lr_test_predictions = lr_model.predict(x_test)

 train_accuracy_score = metrics.accuracy_score(lr_train_predictions, y_train)
 test_accuracy_score = metrics.accuracy_score(lr_test_predictions, y_test)
 normalized_score = test_accuracy_score / train_accuracy_score

 report_dataframe.loc[len(report_dataframe.index)] = {'Test_Size': size,
'Train_Accuracy_Score': train_accuracy_score,
'Test_Accuracy_Score': test_accuracy_score,
'Normalized_Score': normalized_score }

return report_dataframe

report_dataframe = lr_automation(x, y)
print(report_dataframe)

```

```

Test_Size Train_Accuracy_Score Test_Accuracy_Score Normalized_Score
0 0.1 0.970350 0.904762 0.932407
1 0.2 0.957576 0.975904 1.019140
2 0.3 0.972318 0.943548 0.970411
3 0.4 0.971660 0.915663 0.942369
4 0.5 0.956311 0.966184 1.010324
5 0.6 0.963636 0.947581 0.983338
6 0.7 0.983740 0.948276 0.963950
7 0.8 0.987805 0.945619 0.957294
8 0.9 1.000000 0.924731 0.924731

```

```

In [18]: from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

```

```

In [19]: y = df3.iloc[:,1]
x = df3.iloc[:,2:31]

```

```

In [20]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 3)

```

```

In [21]: lr_model = LogisticRegression()

```

```
lr_model.fit(x_train, y_train)
```

```
lr_train_predictions = lr_model.predict(x_train)
```

```
lr_test_predictions = lr_model.predict(x_test)
```

```
print('Train Accurary Score = ', metrics.accuracy_score(lr_train_predictions, y_train))
```

```
print('Test Accurary Score = ', metrics.accuracy_score(lr_test_predictions, y_test))
```

```
Train Accurary Score = 0.9579288025889967
```

```
Test Accurary Score = 0.9423076923076923
```

```
C:\Users\Jobin\anaconda3\lib\site-packages\sklearn\linear_model_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1): STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

```
n_iter_i = _check_optimize_result(
```

```
In [22]: df3.drop([0,1,2,5,7,9,14,15,17,18],axis = 0, inplace = True)
```

```
df3.head(500)
```

Out[22]:

```
id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
points_mean ... radius_worst texture_worst perimeter_worst area_worst compactness_worst concavity_worst concave
points_worst symmetry_worst frac
```

```
20 8510653 0 13.080 15.71 85.63 520.0 0.10750 0.12700 0.04568 0.03110 ... 14.500 20.49 96.09 630.5 0.27760
```

```
0.18900 0.07283 0.3184 21 8510824 0 9.504 12.44 60.34 273.9 0.10240 0.06492 0.02956 0.02076 ... 10.230 15.66
```

```
65.13 314.9 0.11480 0.08867 0.06227 0.2450 24 852552 1 16.650 21.38 110.00 904.6 0.11210 0.14570 0.15250
```

```
0.09170 ... 26.460 31.56 177.00 2215.0 0.35780 0.46950 0.20950 0.3613 33 854002 1 19.270 26.47 127.90 1162.0
```

```
0.09401 0.17190 0.16570 0.07593 ... 24.150 30.90 161.40 1813.0 0.65900 0.60910 0.17850 0.3672 35 854253 1
```

```
16.740 21.59 110.10 869.5 0.09610 0.13360 0.13480 0.06018 ... 20.010 29.02 133.50 1229.0 0.38350 0.54090
```

```
0.18130 0.4863
```

```
... .. 562 925622 1 15.220 30.62 103.40 716.9 0.10480 0.20870
```

```
0.25500 0.09429 ... 17.520 42.79 128.70 915.0 0.79170 1.17000 0.23560 0.4089 565 926682 1 20.130 28.25 131.20
```

```
1261.0 0.09780 0.10340 0.14400 0.09791 ... 23.690 38.25 155.00 1731.0 0.19220 0.32150 0.16280 0.2572 566
```

```
926954 1 16.600 28.08 108.30 858.1 0.08455 0.10230 0.09251 0.05302 ... 18.980 34.12 126.70 1124.0 0.30940
```

```
0.34030 0.14180 0.2218 567 927241 1 20.600 29.33 140.10 1265.0 0.11780 0.27700 0.35140 0.15200 ... 25.740
```

```
39.42 184.60 1821.0 0.86810 0.93870 0.26500 0.4087 568 92751 0 7.760 24.54 47.92 181.0 0.05263 0.04362
```

```
0.00000 0.00000 ... 9.456 30.37 59.16 268.6 0.06444 0.00000 0.00000 0.2871
```

403 rows × 32 columns

```
In [23]: y = df3.iloc[:, 1]
x = df3.iloc[:, 2:31]
```

```
In [24]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 3)
```

```
In [25]: lr_model = LogisticRegression()
lr_model.fit(x_train, y_train)
```

```
lr_train_predictions = lr_model.predict(x_train)
lr_test_predictions = lr_model.predict(x_test)
```

```
print('Train Accuracy Score = ', metrics.accuracy_score(lr_train_predictions, y_train))
print('Test Accuracy Score = ', metrics.accuracy_score(lr_test_predictions, y_test))
```

Train Accuracy Score = 0.9635761589403974  
Test Accuracy Score = 0.9504950495049505

C:\Users\Jobin\anaconda3\lib\site-packages\sklearn\linear\_model\\_logistic.py:763: ConvergenceWarning:  
lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

n\_iter\_i = \_check\_optimize\_result(

```
In [26]: #20
```

```
df3 = Cancer_Data.copy()
num_cols = [col for col in df3.columns if df3[col].dtype != 'object']
print(num_cols)
df4 = df3[num_cols]
```

```
['id', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean',
'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean',
'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se',
'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst', 'compactness_worst',
'concavity_worst', 'concave points_worst', 'symmetry_worst', 'fractal_dimension_worst']
```

```
In [27]: df4.head()
```

Out[27]:

```
id radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean concave
points_mean symmetry_mean ... radius_worst texture_worst perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst conca
points_worst
```

```
0 842302 17.99 10.38 122.8 1001.0 0.11840 0.27760 0.3001 0.14710 0.2419 ... 25.38 17.33 184.60 2019.0 0.1622
0.6656 0.7119 0.26 1 842517 20.57 17.77 132.9 1326.0 0.08474 0.07864 0.0869 0.07017 0.1812 ... 24.99 23.41
158.80 1956.0 0.1238 0.1866 0.2416 0.18 2 84300903 19.69 21.25 130.0 1203.0 0.10960 0.15990 0.1974 0.12790
0.2069 ... 23.57 25.53 152.50 1709.0 0.1444 0.4245 0.4504 0.24 3 84348301 11.42 20.38 NaN 386.1 0.14250 NaN
0.2414 0.10520 0.2597 ... 14.91 26.50 98.87 567.7 0.2098 0.8663 0.6869 0.25 4 84358402 20.29 14.34 135.1 1297.0
0.10030 0.13280 0.1980 0.10430 0.1809 ... 22.54 16.67 152.20 1575.0 0.1374 0.2050 0.4000 0.16
```

5 rows × 31 columns

```
In [28]: def get_numerical_summary(df4):
total = df4.shape[0]
columns_with_missing_data = [col for col in df4.columns if df4[col].isnull().sum() > 0] missing_percent = {}

for col in columns_with_missing_data:
 null_count = df4[col].isnull().sum()
 per = (null_count/total) * 100
 missing_percent[col] = per
 print("{} : {} ({}%)".format(col, null_count, round(per, 3)))
return missing_percent
```

```
missing_percent = get_numerical_summary(df4)
```

```
radius_mean : 4 (0.703%)
texture_mean : 5 (0.879%)
perimeter_mean : 4 (0.703%)
area_mean : 4 (0.703%)
smoothness_mean : 1 (0.176%)
compactness_mean : 5 (0.879%)
concavity_mean : 13 (2.285%)
concave points_mean : 11 (1.933%)
symmetry_mean : 2 (0.351%)
fractal_dimension_mean : 2 (0.351%)
radius_se : 2 (0.351%)
texture_se : 5 (0.879%)
perimeter_se : 1 (0.176%)
area_se : 68 (11.951%)
compactness_se : 5 (0.879%)
concavity_se : 12 (2.109%)
concave points_se : 10 (1.757%)
symmetry_se : 4 (0.703%)
radius_worst : 2 (0.351%)
texture_worst : 1 (0.176%)
perimeter_worst : 2 (0.351%)
area_worst : 2 (0.351%)
smoothness_worst : 135 (23.726%)
compactness_worst : 2 (0.351%)
concavity_worst : 13 (2.285%)
concave points_worst : 12 (2.109%)
```

```
symmetry_worst : 6 (1.054%)
fractal_dimension_worst : 7 (1.23%)
```

In [29]: #21

```
def drop_feature(data_frame, threshold):

 for col, per in missing_percent.items():
 if per > threshold:
 data_frame.drop(col, axis = 1, inplace = True)

 return data_frame

knn_data_cleaned = drop_feature(df4.copy(), 22)

_ = get_numerical_summary(df4)
```

```
radius_mean : 4 (0.703%)
texture_mean : 5 (0.879%)
perimeter_mean : 4 (0.703%)
area_mean : 4 (0.703%)
smoothness_mean : 1 (0.176%)
compactness_mean : 5 (0.879%)
concavity_mean : 13 (2.285%)
concave points_mean : 11 (1.933%)
symmetry_mean : 2 (0.351%)
fractal_dimension_mean : 2 (0.351%)
radius_se : 2 (0.351%)
texture_se : 5 (0.879%)
perimeter_se : 1 (0.176%)
area_se : 68 (11.951%)
compactness_se : 5 (0.879%)
concavity_se : 12 (2.109%)
concave points_se : 10 (1.757%)
symmetry_se : 4 (0.703%)
radius_worst : 2 (0.351%)
texture_worst : 1 (0.176%)
perimeter_worst : 2 (0.351%)
area_worst : 2 (0.351%)
smoothness_worst : 135 (23.726%)
compactness_worst : 2 (0.351%)
concavity_worst : 13 (2.285%)
concave points_worst : 12 (2.109%)
symmetry_worst : 6 (1.054%)
fractal_dimension_worst : 7 (1.23%)
```

In [30]: #22

```
def drop_row(Cancer_Data, threshold):
 # Getting Missing count of each sample
 for idx in range(Cancer_Data.shape[0]):
 df4.loc[idx, 'missing_count'] = df4.iloc[idx, :].isnull().sum()

 df4.drop(df4[df4['missing_count'] > threshold].index, axis = 0, inplace = True)
 return df4
```



```
print("Samples Before Removal : {}".format(knn_data_cleaned.shape[0]))
```

```
this can take some time... depend on the shape of the dataframe
```

```
knn_data_cleaned = drop_row(knn_data_cleaned, 1)
```

```
print("Samples After Removal : {}".format(knn_data_cleaned.shape[0]))
```

```
Samples Before Removal : 569
```

```
Samples After Removal : 335
```

```
In []:
```

Ernesto Vilches

Ernesto Vilches

CIS4591

Capstone Documentation

Due to a lack of understanding of the project, this section will be dedicated to the professor in proving our abilities that we know how to code.

Goal was to predict customer churn as accurately as possible, we cleaned our data by dropping some columns and joining columns from other csv's. Also, create a python script capable of handling a data frame and to then predict in the most efficient manner which users would move out ("churn"). Progressively refine the processing of our UI through trial and error to get the fastest and most efficient running version possible. Create a visually attractive bar graph that portrays the results and the aspects that affect customer churn.

User Stories:

- As a user, I want to have a homepage, so that I can view important information at a glance
- As a developer, I want a single data frame, to make creating a machine learning model more efficient
- As a user, I want to have multiple graphs that consume data from our model, in order to visualize customer churn.

**Data cleaning code:**

```

data_cleaning.py > -
1 import pandas as pd
2 import numpy as np
3
4 #Read CSV files
5 product_data_df = pd.read_csv("JUNO - Cleaned Product Data - FINAL.csv")
6 subscription_df = pd.read_csv("JUNO - Subscriptions.csv")
7 sfdc_acc_export_df = pd.read_csv("JUNO - SFDC Account Export.csv")
8
9 #Merge data
10 temp_df = sfdc_acc_export_df.merge(subscription_df, left_on='Full Account ID', right_on='Row Labels')
11 combined_df = temp_df.merge(product_data_df, left_on='JUNO Account ID', right_on='account_id')
12
13 #This will fill the "region" column not sure why it messes up
14 #when we do the merge... but we might want to use region for our model
15 account_currency_map = {'USD': 'NAM', 'CAD': 'NAM', 'EUR': 'EU', 'GBP': 'EU', 'AUD': 'AU', 'NZD': 'AU', 'ZAR': 'SA'}
16 combined_df['Account Currency'] = combined_df['Account Currency'].map(account_currency_map)
17
18 #remove invalid durations due to faulty data
19 combined_df = combined_df[combined_df['Duration'] != '#NUM!']
20 combined_df = combined_df[combined_df['Duration'] < '44602']
21
22 #csv for displaydata for UI
23 combined_df.to_csv("display.csv", index=False)
24
25 ##todo:
26 ***
27 Calculate Churn within Python
28 Data cleaning
29 ***
30
31 #Drop repeating columns/stuff we don't need
32 drop = ["Account ID", "Full User ID", "JUNO Account ID", "Row Labels", "Full Account ID", "account_id", "account_user_id", "Current State", "created_at", "Last Activity", "Last Modified Date", ""]
33 combined_df.drop(columns=drop, inplace=True)
34
35 #make indicator variables
36 combined_df = pd.get_dummies(combined_df, columns=["Kaseya Market Segment", "Account Currency"])

```

---

```

data_cleaning.py > -
14 #when we do the merge... but we might want to use region for our model
15 account_currency_map = {'USD': 'NAM', 'CAD': 'NAM', 'EUR': 'EU', 'GBP': 'EU', 'AUD': 'AU', 'NZD': 'AU', 'ZAR': 'SA'}
16 combined_df['Account Currency'] = combined_df['Account Currency'].map(account_currency_map)
17
18 #remove invalid durations due to faulty data
19 combined_df = combined_df[combined_df['Duration'] != '#NUM!']
20 combined_df = combined_df[combined_df['Duration'] < '44602']
21
22 #csv for displaydata for UI
23 combined_df.to_csv("display.csv", index=False)
24
25 ##todo:
26 ***
27 Calculate Churn within Python
28 Data cleaning
29 ***
30
31 #Drop repeating columns/stuff we don't need
32 drop = ["Account ID", "Full User ID", "JUNO Account ID", "Row Labels", "Full Account ID", "account_id", "account_user_id", "Current State", "created_at", "Last Activity", "Last Modified Date", ""]
33 combined_df.drop(columns=drop, inplace=True)
34
35 #make indicator variables
36 combined_df = pd.get_dummies(combined_df, columns=["Kaseya Market Segment", "Account Currency"])
37
38 #remove empty rows
39 combined_df = combined_df.dropna(how='any')
40
41 #Make it a CSV
42 combined_df.to_csv('combined.csv', index=False)
43
44 print(combined_df.head())
45
46 print("Done")

```

**Model Code:**

```

model.py > k fold
1 from prometheus.client import Counter
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import train_test_split, StratifiedKFold
4 from sklearn.metrics import precision_score, roc_curve, roc_auc_score, classification_report, accuracy_score, confusion_matrix
5 import pandas as pd
6 from imblearn.over_sampling import SMOTE
7 from sklearn.ensemble import ExtraTreesClassifier
8 import matplotlib.pyplot as plt
9
10 def k_fold(file):
11 data = pd.read_csv(file)
12 drops = ["Churn", "Kaseya Market Segment - None -",
13 "Kaseya Market Segment End User", "Kaseya Market Segment Financial Services", "Kaseya Market Segment General Business",
14 "Kaseya Market Segment Government", "Kaseya Market Segment Healthcare", "Kaseya Market Segment Hospitality",
15 "Kaseya Market Segment Retail", "Kaseya Market Segment Software Vendor", "Account Currency_AU",
16 "Account Currency_SA", "reputation_to_date", "Connect 2019"]
17 y = data["Churn"]
18 x = data.drop(columns=drops)
19 scores = []
20
21 oversample = SMOTE()
22 x, y = oversample.fit_resample(x, y)
23
24 kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=45)
25 conf_matrix = None
26 i = 1
27 for train_index, test_index in kf.split(x, y):
28 print("{} of Kfold {}".format(i, kf.n_splits))
29 x_train, x_test = x.loc[train_index], x.loc[test_index]
30 y_train, y_test = y.loc[train_index], y.loc[test_index]
31
32 model = LogisticRegression(max_iter=1000)
33 print("Shape: ", x_train.shape, x_test.shape, y_train.shape, y_test.shape)
34 model.fit(x_train, y_train)
35
36 y_prediction = model.predict(x_test)

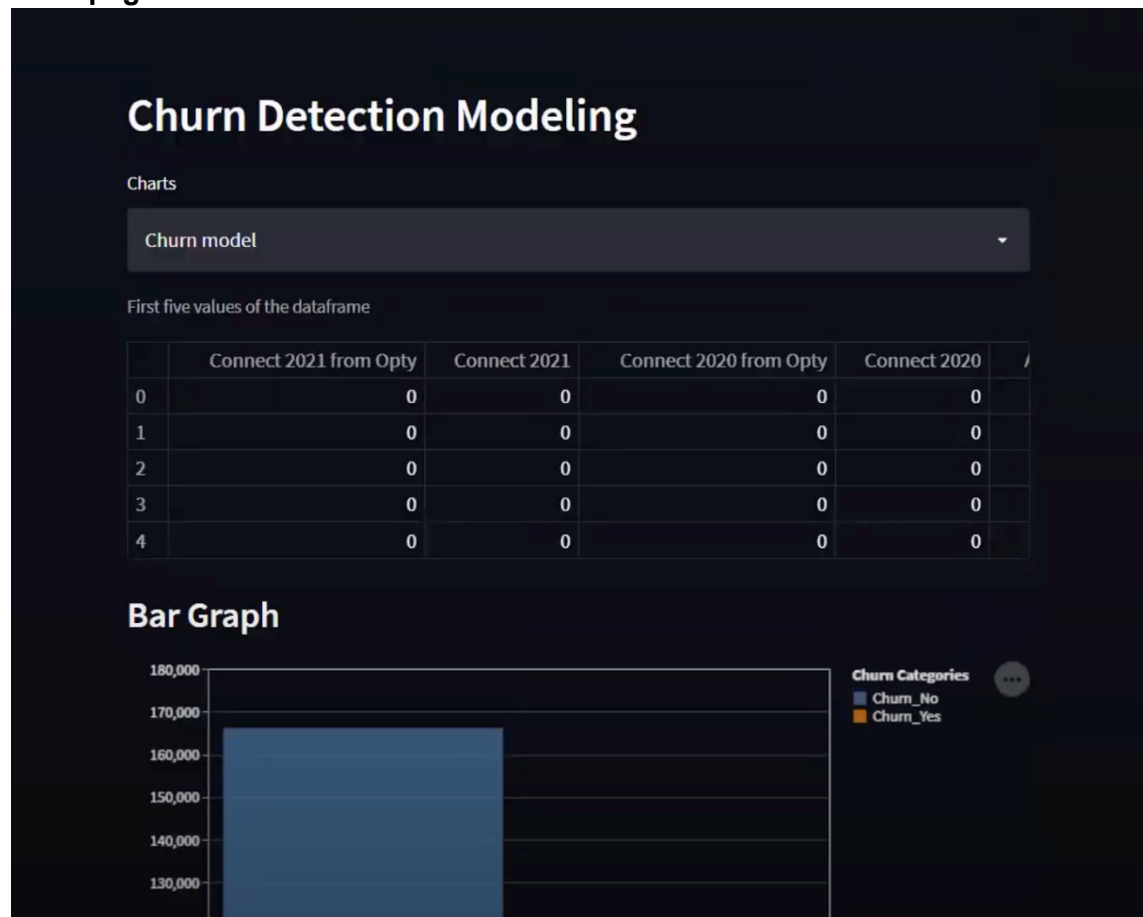
```

```

model.py > k fold > score
20 scores = []
21
22 oversample = SMOTE()
23 x, y = oversample.fit_resample(x, y)
24
25 kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=45)
26 conf_matrix = None
27 i = 1
28 for train_index, test_index in kf.split(x, y):
29 print("{} of Kfold {}".format(i, kf.n_splits))
30 x_train, x_test = x.loc[train_index], x.loc[test_index]
31 y_train, y_test = y.loc[train_index], y.loc[test_index]
32
33 model = LogisticRegression(max_iter=1000)
34 print("Shape: ", x_train.shape, x_test.shape, y_train.shape, y_test.shape)
35 model.fit(x_train, y_train)
36
37 y_prediction = model.predict(x_test)
38 print("Prediction: ", y_prediction)
39 score = roc_auc_score(y_test, y_prediction)
40 scores.append(score)
41 # print('ROC AUC score: ', score)
42 # print()
43 # print(classification_report(y_test, y_prediction, digits=6))
44 report = pd.DataFrame(classification_report(y_test, y_prediction, digits=6, output_dict=True)).transpose()
45 conf_matrix = confusion_matrix(y_test, y_prediction)
46 print(conf_matrix)
47 print()
48 i += 1
49 avg = sum(scores)/len(scores)
50 print(classification_report(y_test, y_prediction, digits=6))
51 print("Average ROC AUC Score: ", avg)
52 print("Precision: ", precision_score(y_true=y_test, y_pred=y_prediction, average="binary"))
53
54 return [report, conf_matrix]

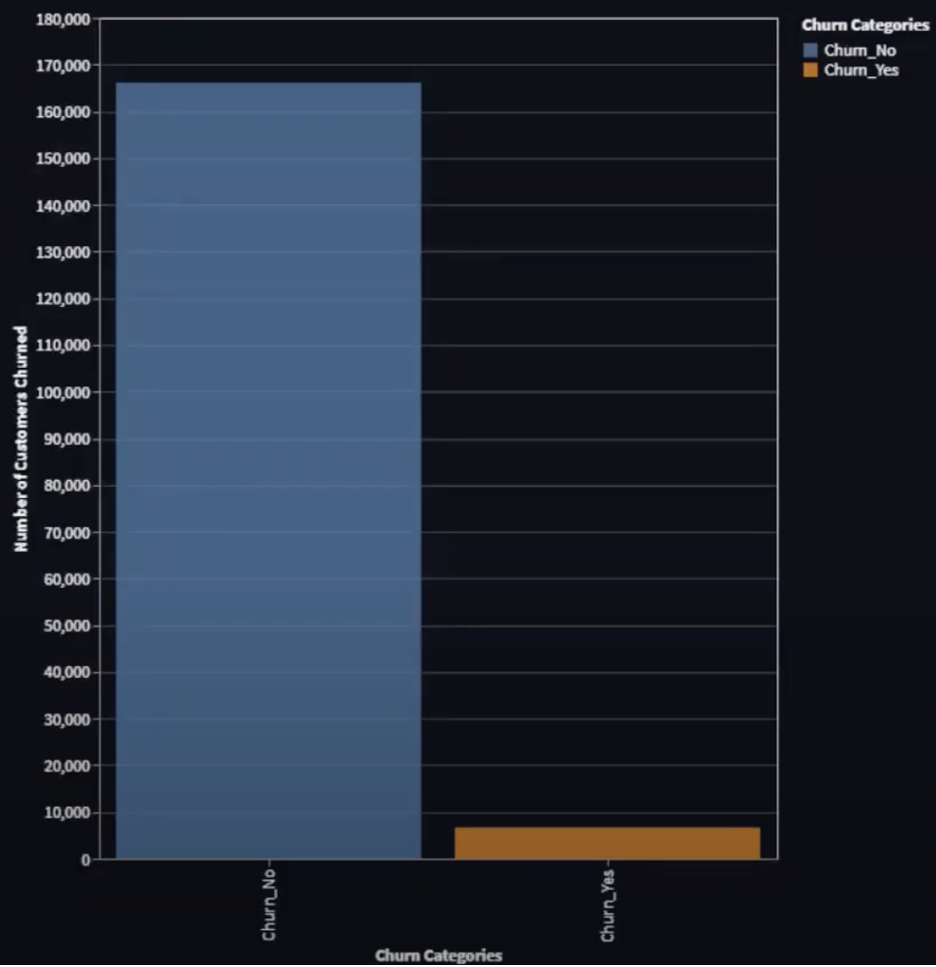
```

## Homepage Model:



Amount of people who churned vs amount who did not example:

## Bar Graph



Sequence Diagram:

