README for CSE 464
Project Part 1

Shantanu Shishodia
1225590054

GitHub Repository link: https://github.com/shantanushishodia/cse-464-2023-sshishod

Instructions to Run

- Download the cse-464-sshishod.zip file from this repository
- Run mvn package
- This should run all tests for the project
- This command will build the project in the target folder as well
- Alternatively, unzip cse-464-sshishod.zip and then open the GraphHandler folder in IntelliJ

APIs

- void graphImporter(String filePath) - import a directed graph in a dot file
- String toString() - Graph information like number of nodes, edges and their directions
- void saveGraphToFile(String filePath) - Write the graph information to a file
- void addOneNode(String label) - Adds a new node to the graph with the given label if it does not exist

- void addMultipleNodes(ArrayList<String> labels) - Add multiple nodes to the graph

- boolean addEdge(String initialNode, String targetNode) - Returns true if edge is added otherwise returns false if edge exists

- void saveGraphDOT(String filePath) - Outputs the modified graph in DOT format to the specified file

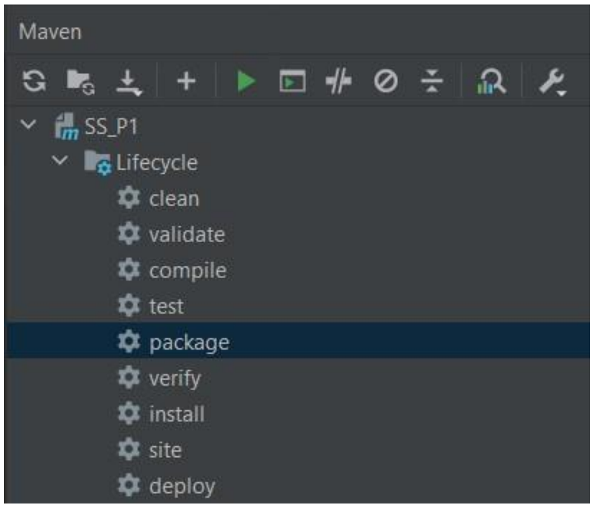
- void saveGraphPNG(String filePath) - Output the modified graph to a PNG file (Graph Visualization)

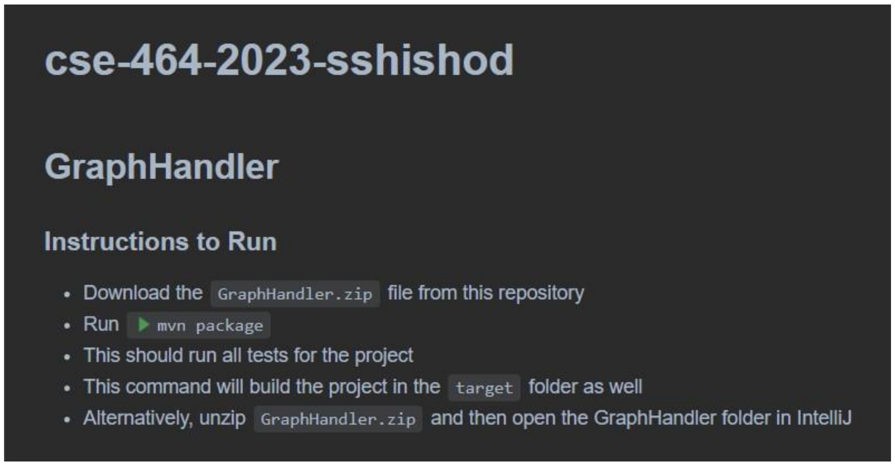
1. Adding Maven support to the project

- I was following the standard directory layout from when I had started my project by following the guide at https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html
- I created a pom.xml and added all my project dependencies with the feature 1 implemented. The commit which contains this change: https://github.com/shantanushishodia/cse-464-2023sshishod/commit/02bfb1b77b5f9b57182503ffa81ad2fad3011a4e
- A later commit fixed the tests not running while executing the mvn package command in which the maven-surefire-plugin needed to be changed to 2.22.0: https://github.com/shantanushishodia/cse-464-2023sshishod/commit/4775c744f1294f2ddfae27a2b43b71611f7085b9

2. Output for mvn package command (test performed using test1.dot as initial input)

Can use both way to initiate maven package



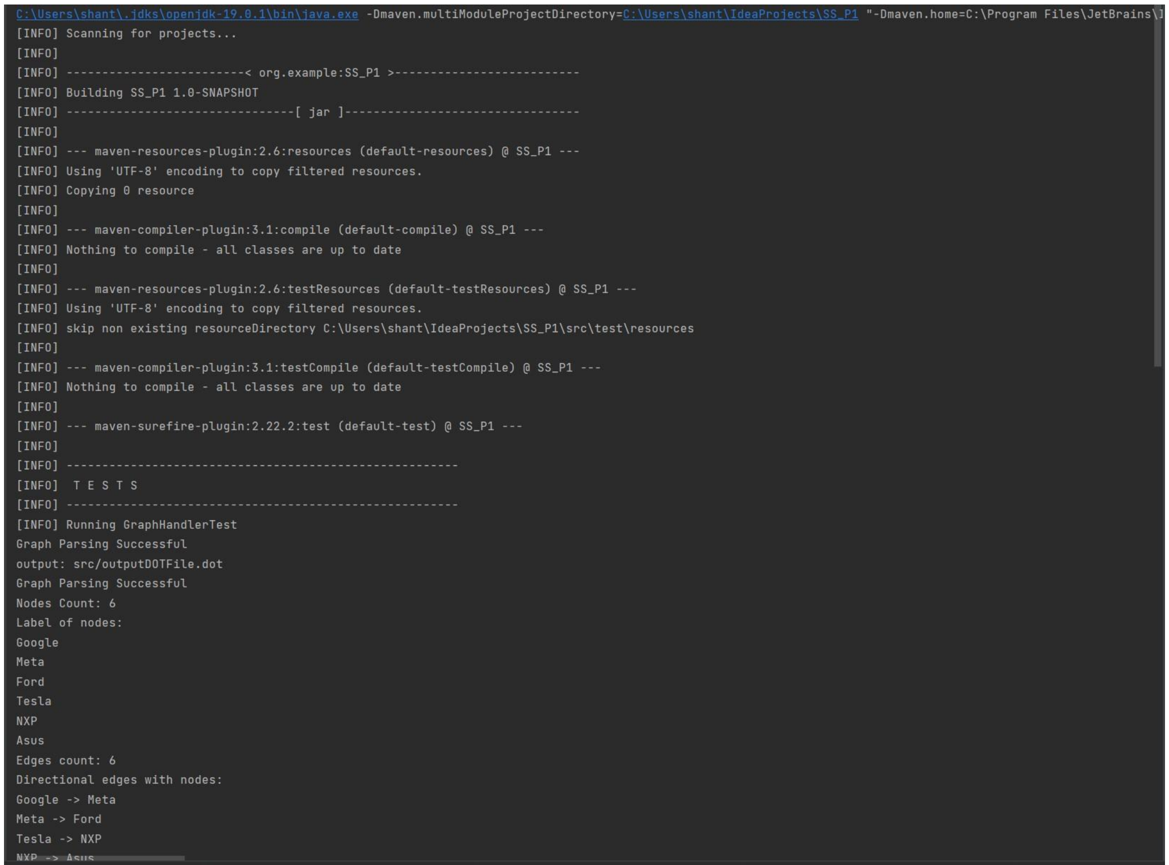


cse-464-2023-sshishod

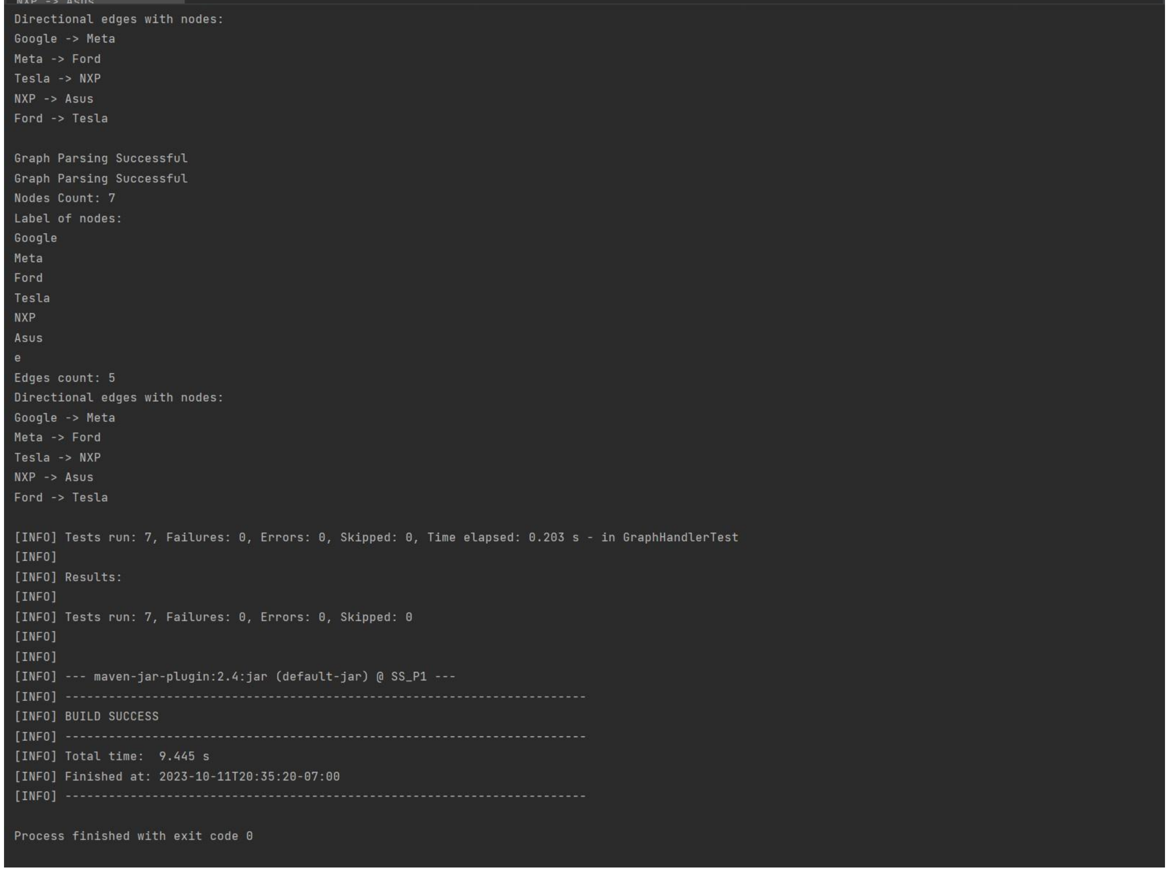
GraphHandler

Instructions to Run

- Download the `GraphHandler.zip` file from this repository
- Run `mvn package`
- This should run all tests for the project
- This command will build the project in the `target` folder as well
- Alternatively, unzip `GraphHandler.zip` and then open the GraphHandler folder in IntelliJ



```
C:\Users\shant\.jdks\openjdk-19.0.1\bin\java.exe -Dmaven.multiModuleProjectDirectory=C:\Users\shant\IdeaProjects\SS_P1 "-Dmaven.home=C:\Program Files\JetBrains\
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< org.example:SS_P1 >-------------------------
[INFO] Building SS_P1 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ SS_P1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ SS_P1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ SS_P1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\shant\IdeaProjects\SS_P1\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ SS_P1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ SS_P1 ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running GraphHandlerTest
Graph Parsing Successful
output: src/outputDOTFile.dot
Graph Parsing Successful
Nodes Count: 6
Label of nodes:
Google
Meta
Ford
Tesla
NXP
Asus
Edges count: 6
Directional edges with nodes:
Google -> Meta
Meta -> Ford
Tesla -> NXP
NXP -> Asus
```

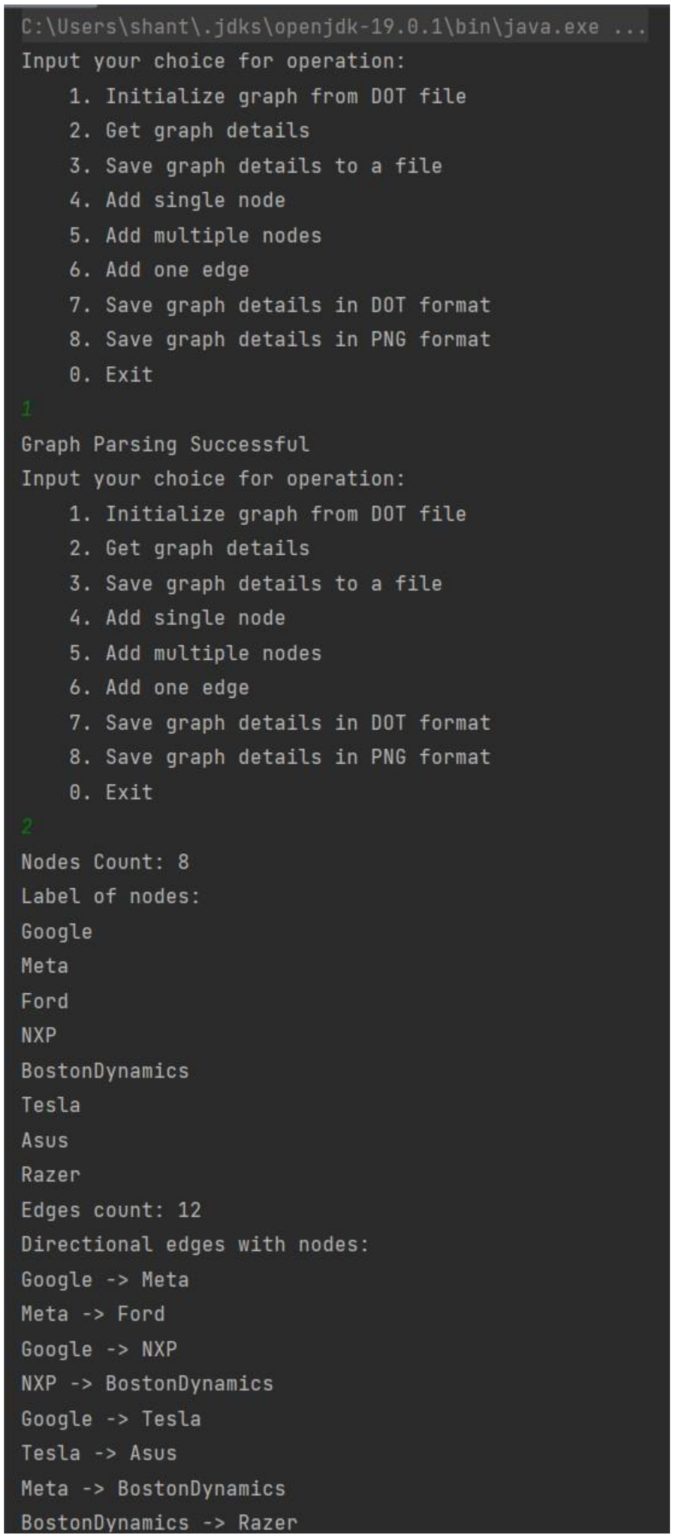


```
Directional edges with nodes:
Google -> Meta
Meta -> Ford
Tesla -> NXP
NXP -> Asus
Ford -> Tesla

Graph Parsing Successful
Graph Parsing Successful
Nodes Count: 7
Label of nodes:
Google
Meta
Ford
Tesla
NXP
Asus
e
Edges count: 5
Directional edges with nodes:
Google -> Meta
Meta -> Ford
Tesla -> NXP
NXP -> Asus
Ford -> Tesla

[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.203 s - in GraphHandlerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ SS_P1 ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.445 s
[INFO] Finished at: 2023-10-11T20:35:20-07:00
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

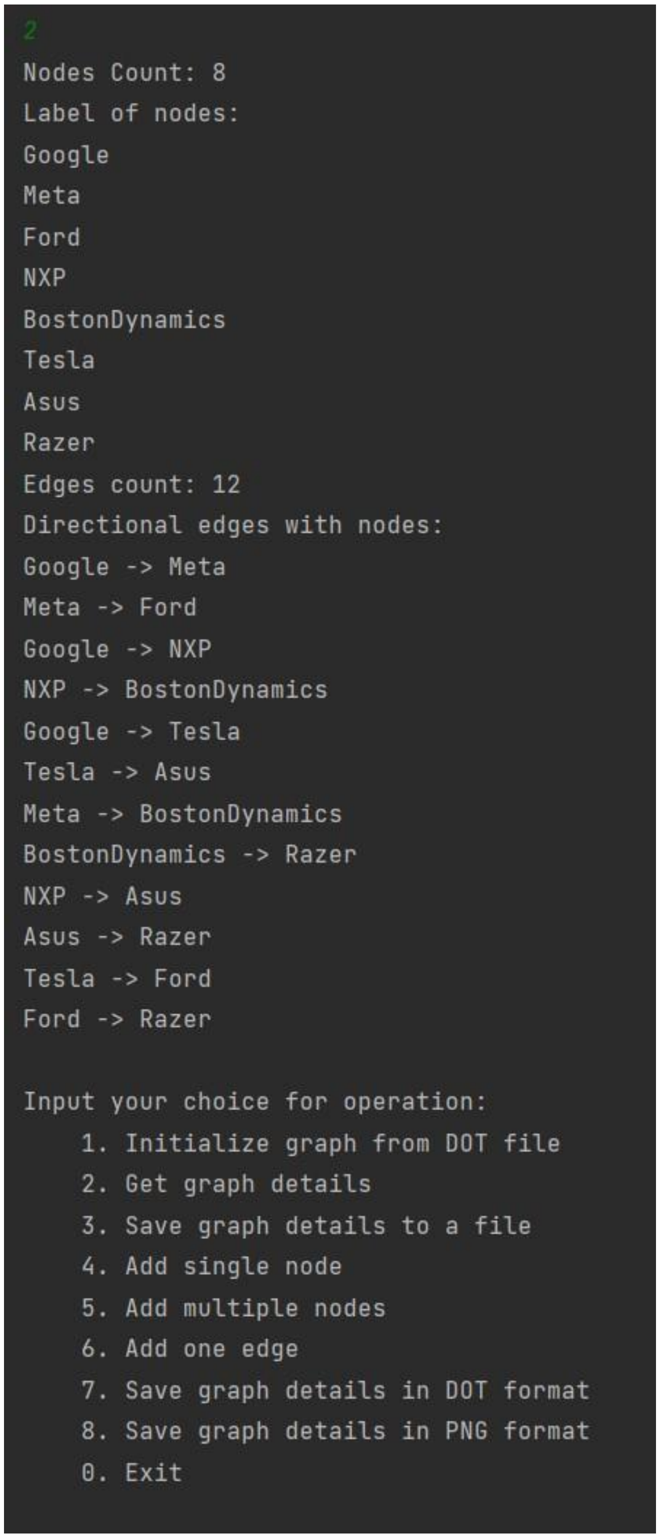
3. **Output for Feature 1 (Following outputs are using companies.dot file)**

Commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/02bfb1b77b5f9b57182503ffa81ad2fad3011a4e

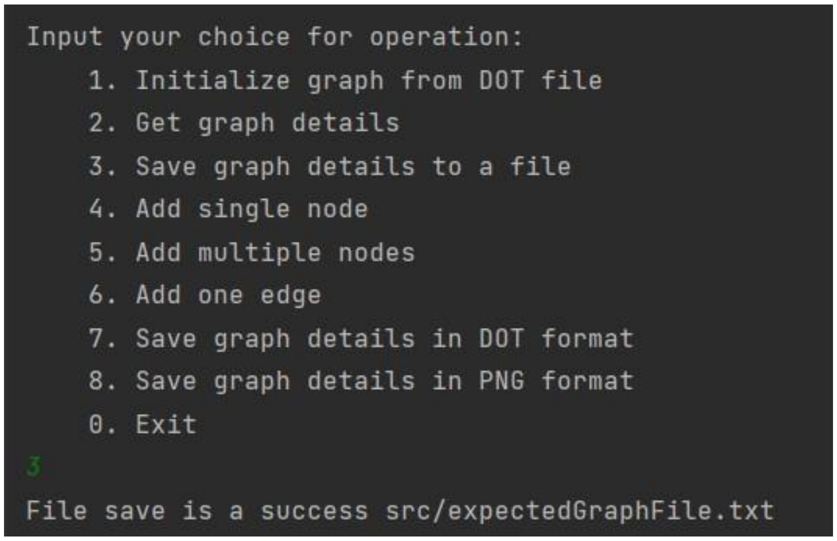
Test commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/232dbbe23a68ab180afcfac79579c5acc5fa7eb5



```
C:\Users\shant\.jdks\openjdk-19.0.1\bin\java.exe ...
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
1
Graph Parsing Successful
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
2
Nodes Count: 8
Label of nodes:
Google
Meta
Ford
NXP
BostonDynamics
Tesla
Asus
Razer
Edges count: 12
Directional edges with nodes:
Google -> Meta
Meta -> Ford
Google -> NXP
NXP -> BostonDynamics
Google -> Tesla
Tesla -> Asus
Meta -> BostonDynamics
BostonDynamics -> Razer
```



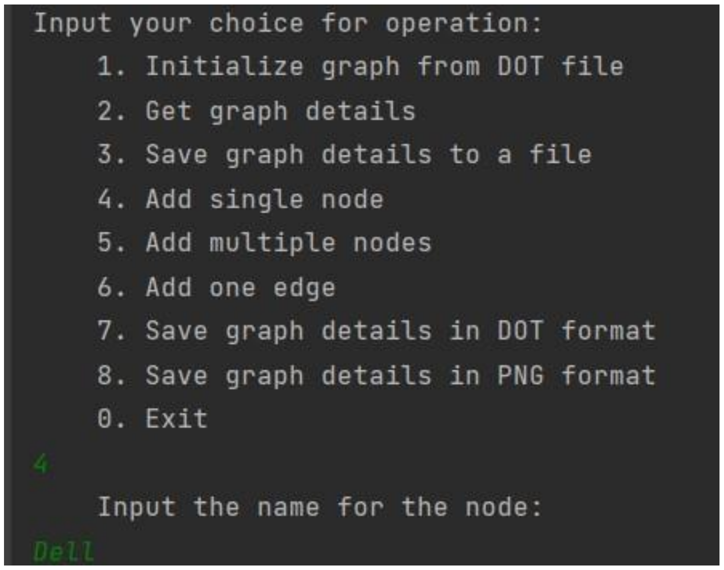
```
2
Nodes Count: 8
Label of nodes:
Google
Meta
Ford
NXP
BostonDynamics
Tesla
Asus
Razer
Edges count: 12
Directional edges with nodes:
Google -> Meta
Meta -> Ford
Google -> NXP
NXP -> BostonDynamics
Google -> Tesla
Tesla -> Asus
Meta -> BostonDynamics
BostonDynamics -> Razer
NXP -> Asus
Asus -> Razer
Tesla -> Ford
Ford -> Razer

Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
```



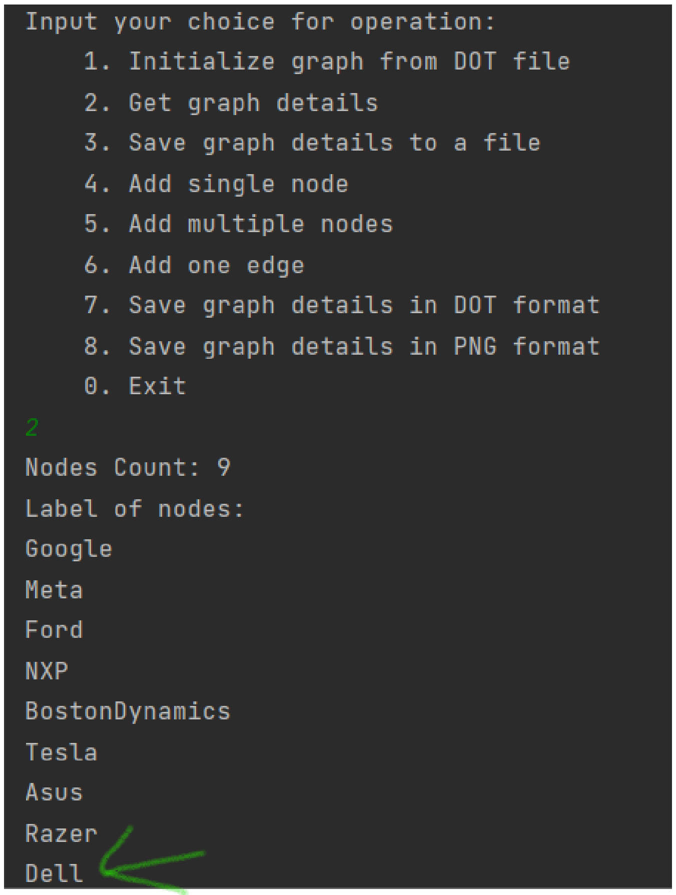
```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
3
File save is a success src/expectedGraphFile.txt
```

4. **Output for Feature 2**

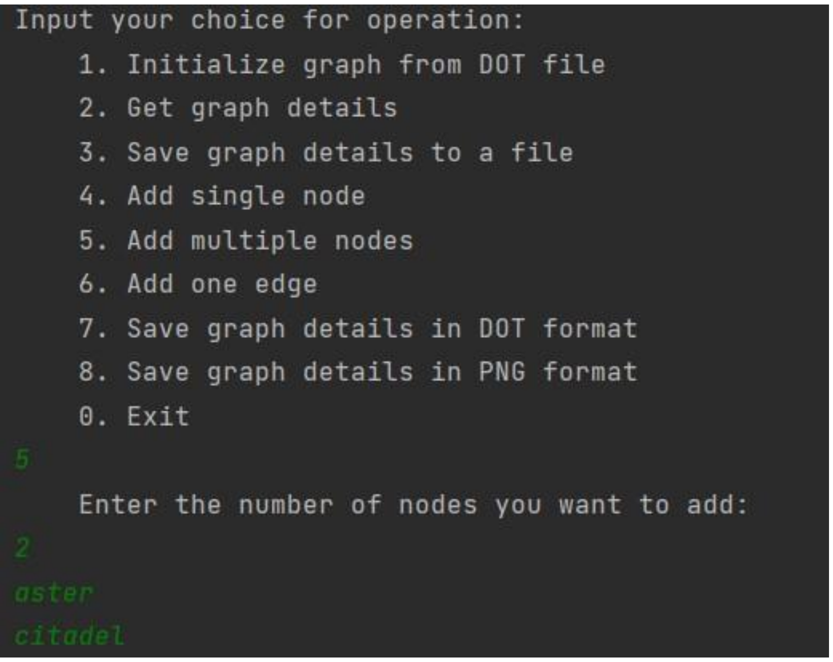
Commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/746bbca009cb58e6b2ecc2cd0a4c6646ef0c3128
Test Commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/756e0007926730e6afac23955a90dcf552c9d6c4



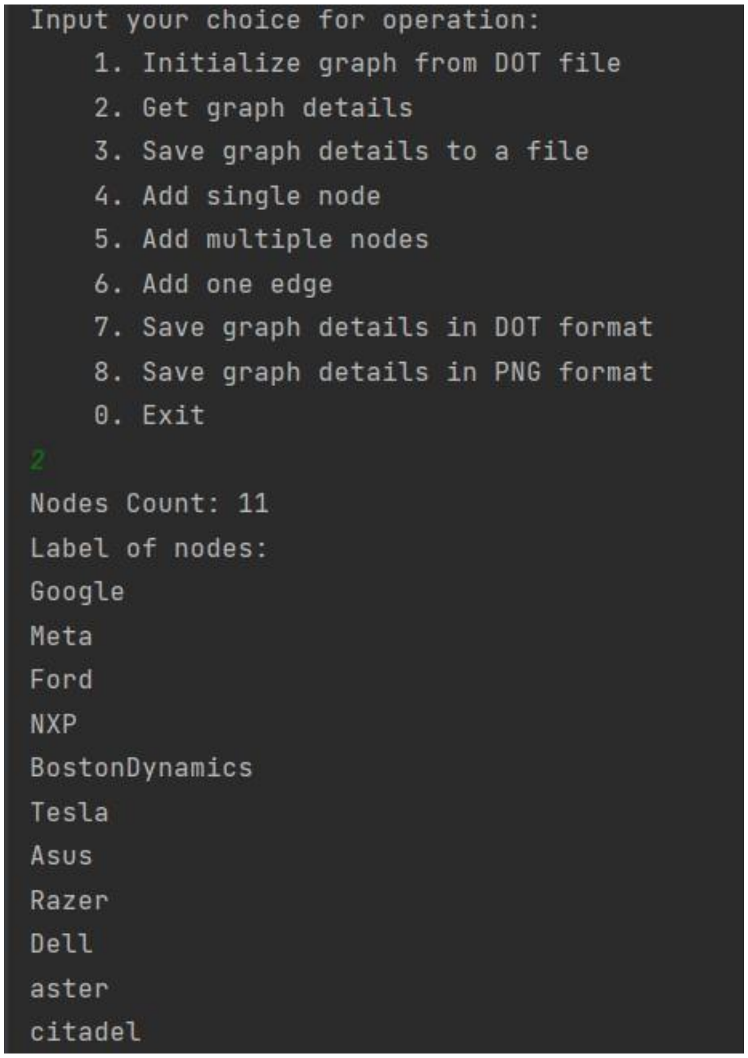
```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
4

    Input the name for the node:
Dell
```



```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
2
Nodes Count: 9
Label of nodes:
Google
Meta
Ford
NXP
BostonDynamics
Tesla
Asus
Razer
Dell
```



```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
5
    Enter the number of nodes you want to add:
2
aster
citadel
```

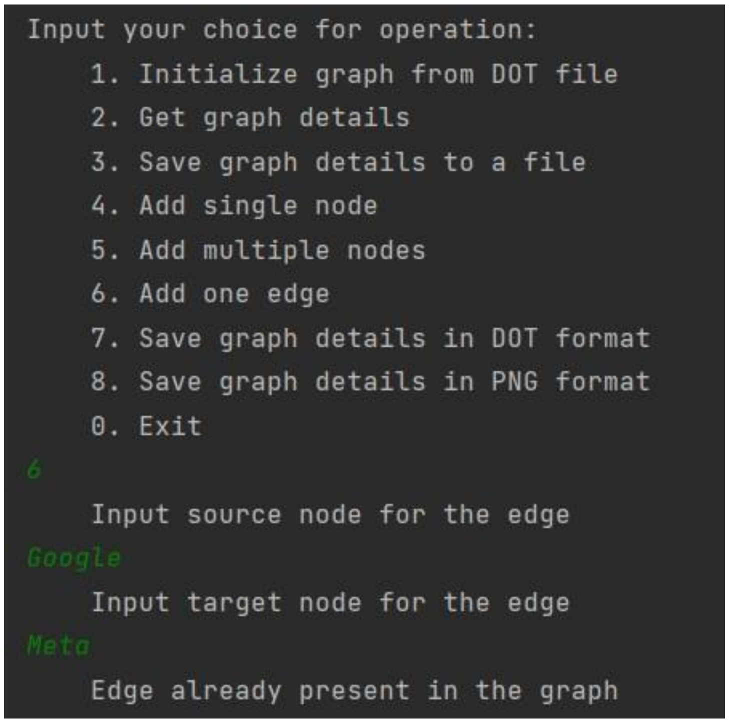


```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
2
Nodes Count: 11
Label of nodes:
Google
Meta
Ford
NXP
BostonDynamics
Tesla
Asus
Razer
Dell
aster
citadel
```

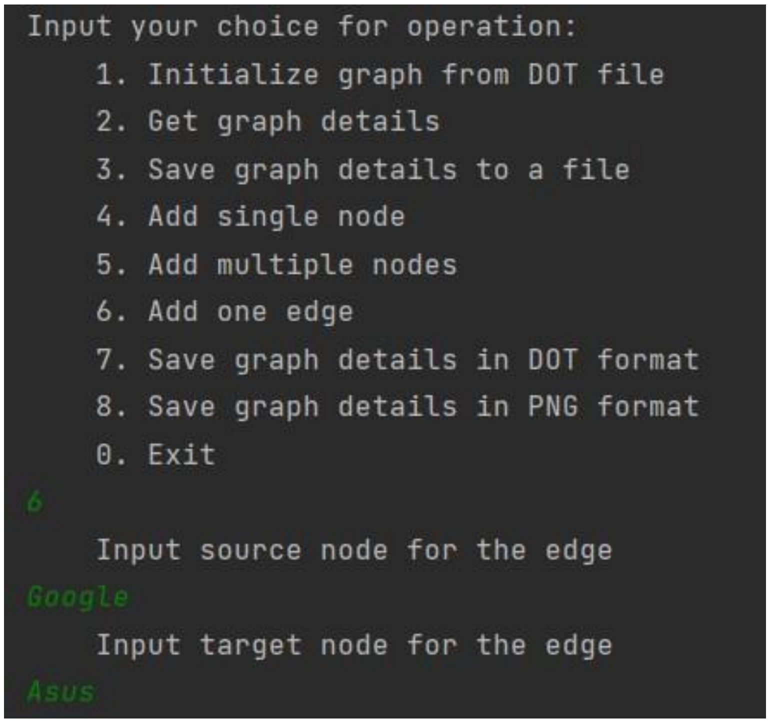
5. Output for Feature 3

Commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/6112f068e5f919476bbaa828f9accfdc9ec7ba99

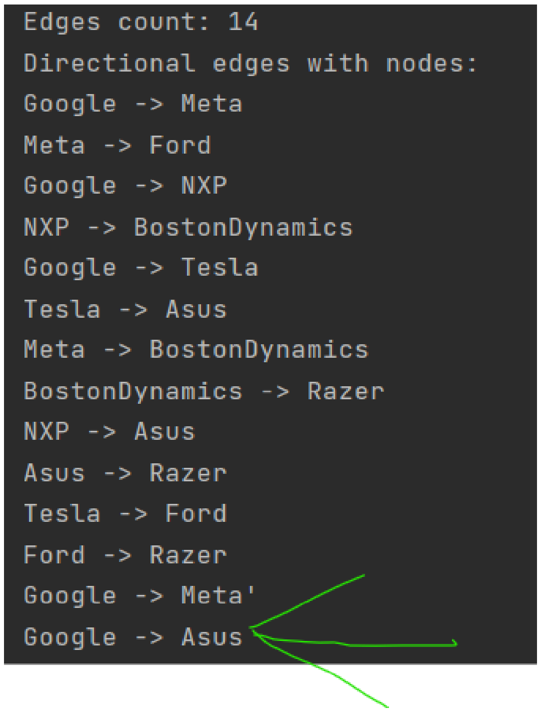
Test Commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/4775c744f1294f2ddfae27a2b43b71611f7085b9



```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
6
    Input source node for the edge
Google
    Input target node for the edge
Meta
    Edge already present in the graph
```



```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
6
    Input source node for the edge
Google
    Input target node for the edge
Asus
```

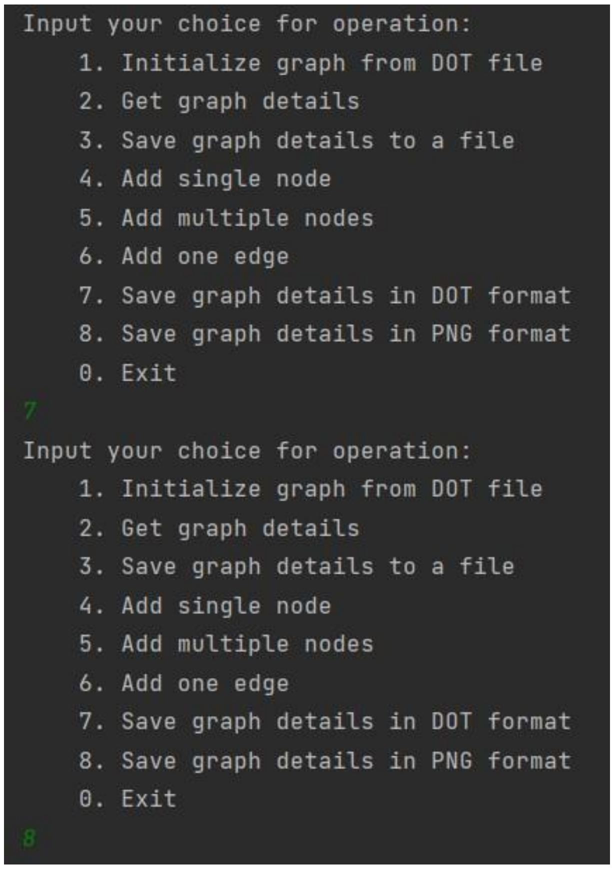


```
Edges count: 14
Directional edges with nodes:
Google -> Meta
Meta -> Ford
Google -> NXP
NXP -> BostonDynamics
Google -> Tesla
Tesla -> Asus
Meta -> BostonDynamics
BostonDynamics -> Razer
NXP -> Asus
Asus -> Razer
Tesla -> Ford
Ford -> Razer
Google -> Meta'
Google -> Asus
```

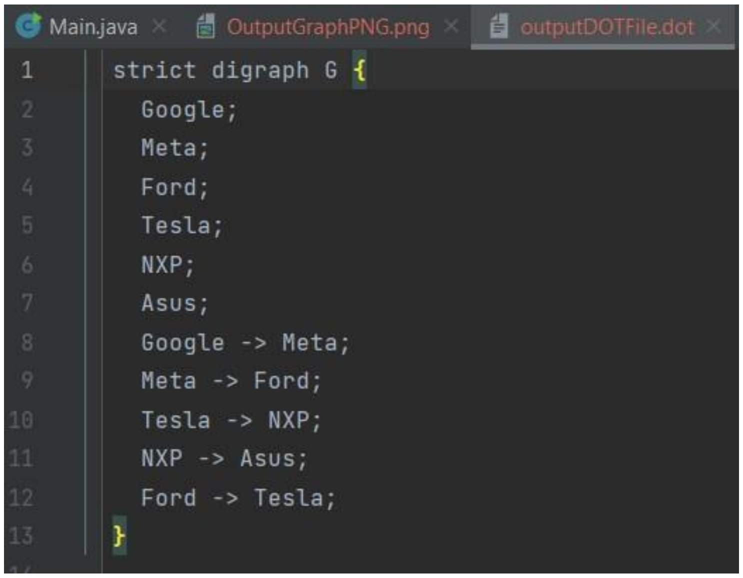
6. Output for Feature 4

Commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/f9876cfc90d0cdcf42880ec0dadc92d609493929

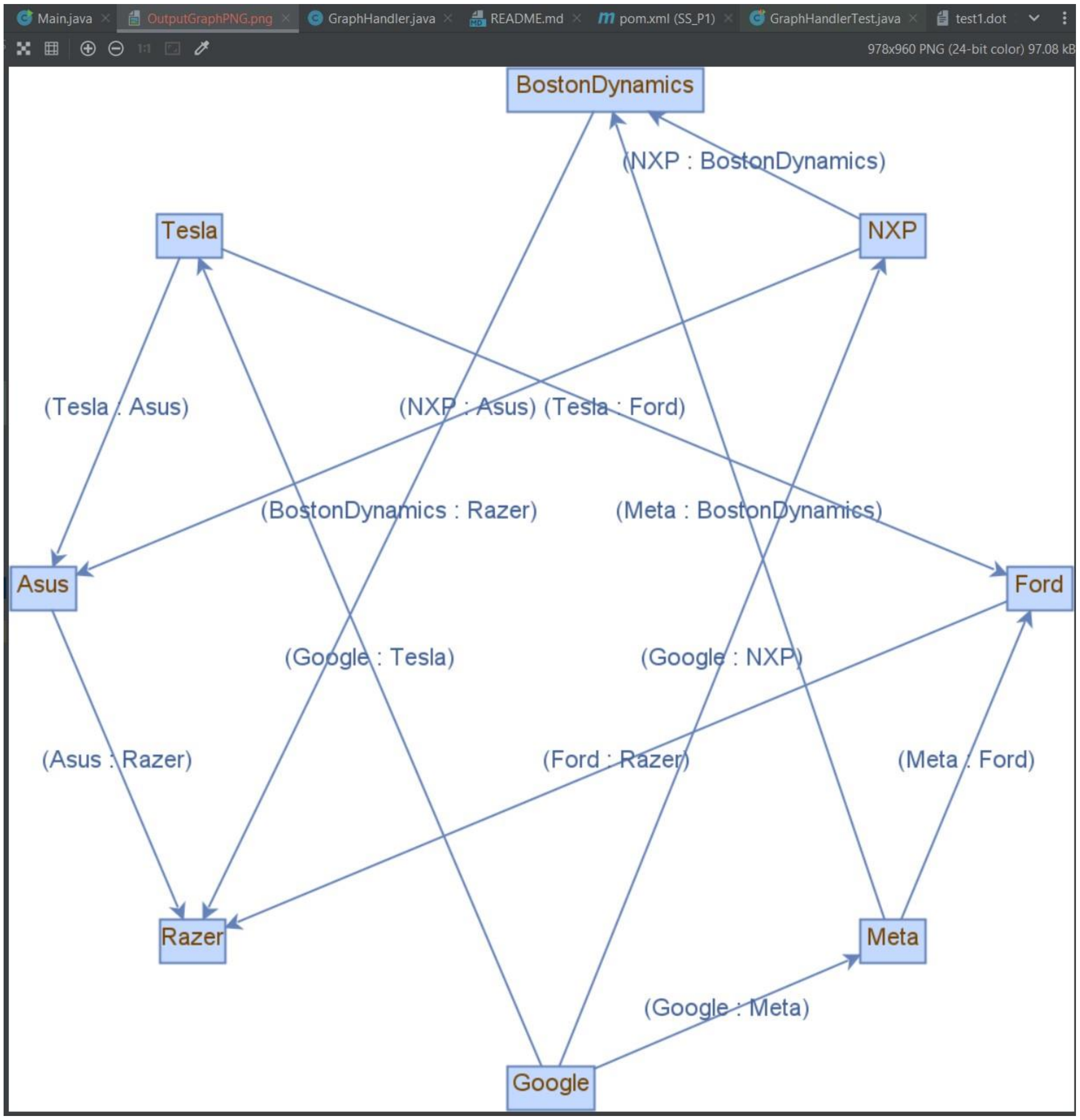
Test Commit: https://github.com/shantanushishodia/cse-464-2023-sshishod/commit/4775c744f1294f2ddfae27a2b43b71611f7085b9



```
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
7
Input your choice for operation:
    1. Initialize graph from DOT file
    2. Get graph details
    3. Save graph details to a file
    4. Add single node
    5. Add multiple nodes
    6. Add one edge
    7. Save graph details in DOT format
    8. Save graph details in PNG format
    0. Exit
8
```



```
strict digraph G {
  Google;
  Meta;
  Ford;
  Tesla;
  NXP;
  Asus;
  Google -> Meta;
  Meta -> Ford;
  Tesla -> NXP;
  NXP -> Asus;
  Ford -> Tesla;
}
```


Main.java
OutputGraphPNG.png
GraphHandler.java
README.md
pom.xml (SS_P1)
GraphHandlerTest.java
test1.dot
978x960 PNG (24-bit color) 97.08 kB
BostonDynamics
Tesla
NXP
Asus
Ford
Razer
Meta
Google
(NXP : BostonDynamics)
(Tesla : Asus)
(NXP : Asus)
(Tesla : Ford)
(BostonDynamics : Razer)
(Meta : BostonDynamics)
(Google : Tesla)
(Google : NXP)
(Asus : Razer)
(Ford : Razer)
(Meta : Ford)
(Google : Meta)