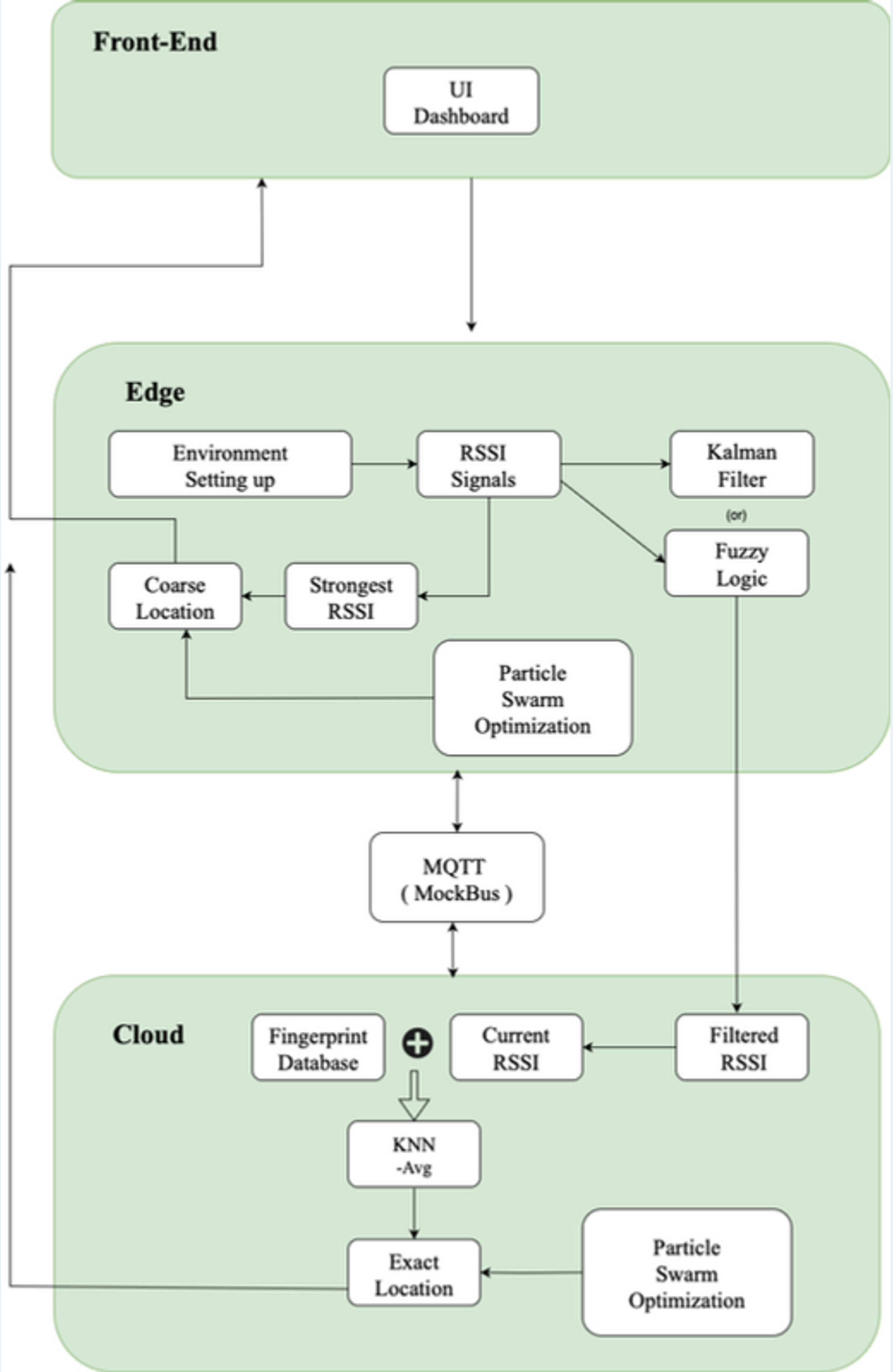
SIMULATED EDGE-IOT SYSTEM FOR INDUSTRIAL RESOURCE TRACKING

- Zhiheng Zhao, Peng Lin,
Leidi Shen, Mengdi Zhang

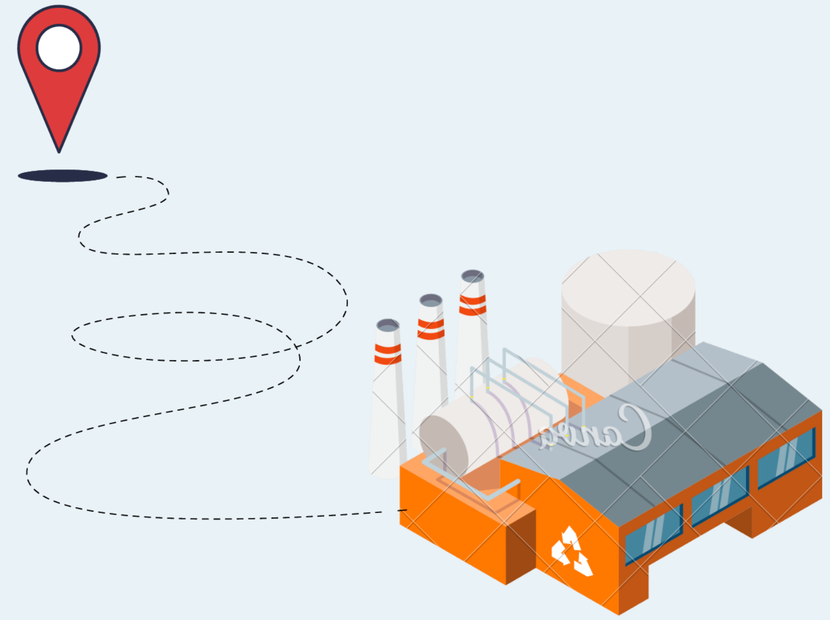
Group No : 3

PROBLEM STATEMENT

Large manufacturing floors hold thousands of items, making it hard for workers to quickly find materials. Edge computing processes data locally to give instant coarse location, ensuring reliability for production. This reduces dependency on cloud, saving computation and storage while speeding up search efficiency.


Front-End
UI Dashboard
Edge
Environment Setting up
RSSI Signals
Kalman Filter
(or)
Fuzzy Logic
Coarse Location
Strongest RSSI
Particle Swarm Optimization
MQTT (MockBus)
Cloud
Fingerprint Database
Current RSSI
Filtered RSSI
KNN -Avg
Exact Location
Particle Swarm Optimization

Environment & Signal Simulation



- Platform: Matplotlib, NumPy, SciPy
- Create 2D grid layout with gateways & WIFI tags.
- Implement RSSI generation (log-normal path loss + Gaussian noise).
- Integrate Kalman filtering (filterpy).
- Deliver a function that, given tag positions, returns filtered RSSI values.

Processing Flow

Implementation(using simpy) :

- **Edge Processing (Coarse Location)**: RSSI from gateways via log-distance model with noise; strongest gateway chosen.

- **Cloud Processing (Refined Location):** Built fingerprinting dataset; applied k-NN on tag RSSI; refined location computed by weighted averaging of neighbors.

- Integrated with Dataset(https://github.com/jinyi-yoon/CampusRSSI/tree/main)

- Selected Environment - Medium-Obs (Office)

```
../debugpy/launcher 54002 -- /Users/kanishkadhanasekar/Downloads/capstone-project-03_cloud9/src/proce
ssing/material_locator.py
------------------------------------------------------------
Enter Material IDs separated by commas : mat115,mat009

Simulating requests for: MAT115, MAT009

============================================================
[0.00s] NEW REQUEST: MAT115
============================================================

[0.00s] Starting coarse location for MAT115...

============================================================
[0.00s] NEW REQUEST: MAT009
============================================================

[0.00s] Starting coarse location for MAT009...


------------------------------------------------------------
[0.50s] Material ID: MAT115
RSSI values:
 AP0: -69.76 dBm
 AP1: -70.88 dBm
 AP2: -75.40 dBm
 AP3: -67.20 dBm
 AP4: -68.80 dBm
 AP5: -67.88 dBm
 AP6: -65.56 dBm
 AP7: -60.56 dBm
 AP8: -68.40 dBm
 AP9: -59.08 dBm
 AP10: -70.68 dBm
 AP11: -63.28 dBm
 AP12: -54.76 dBm
 AP13: -63.76 dBm
 AP14: -65.28 dBm
 AP15: -52.48 dBm

Max RSSI: -52.48 dBm
Coarse Location Zone: AP15
```

```
------------------------------------------------------------

[0.50s] Sending data to cloud for fine location...


------------------------------------------------------------
[0.50s] Material ID: MAT009
RSSI values:
 AP0: -47.64 dBm
 AP1: -55.55 dBm
 AP2: -57.77 dBm
 AP3: -51.73 dBm
 AP4: -56.64 dBm
 AP5: -60.05 dBm
 AP6: -61.18 dBm
 AP7: -65.00 dBm
 AP8: -50.41 dBm
 AP9: -52.09 dBm
 AP10: -59.18 dBm
 AP11: -55.00 dBm
 AP12: -60.64 dBm
 AP13: -55.86 dBm
 AP14: -59.32 dBm
 AP15: -64.59 dBm

Max RSSI: -47.64 dBm
Coarse Location Zone: AP0

------------------------------------------------------------

[0.50s] Sending data to cloud for fine location...
Computing the exact location....waiting for cloud response...

KNN → Predicted: [0.9 9.9], Error: 0.00 meters

Random Forest → Predicted: [1.39 9.68], Error: 0.54 meters

MLP (NN) → Predicted: [1.5  9.67], Error: 0.64 meters

 ==>> Best Model: KNN with error 0.00 meters
[2.50s] Refined location received: {'KNN': array([0.9, 9.9])}
```

```
ssing/material_locator.py
Computing the exact location....waiting for cloud response...

KNN → Predicted: [0.9 9.9], Error: 0.00 meters

Random Forest → Predicted: [1.39 9.68], Error: 0.54 meters

MLP (NN) → Predicted: [1.5  9.67], Error: 0.64 meters

 ==>> Best Model: KNN with error 0.00 meters
[2.50s] Refined location received: {'KNN': array([0.9, 9.9])}


------------------------------------------------------------

[2.50s] Computing optimal path...

------------------------------------------------------------
Computing the exact location....waiting for cloud response...

KNN → Predicted: [0.  2.4], Error: 0.00 meters

Random Forest → Predicted: [0.81 1.87], Error: 0.97 meters

MLP (NN) → Predicted: [0.71 1.03], Error: 1.54 meters

 ==>> Best Model: KNN with error 0.00 meters
[2.50s] Refined location received: {'KNN': array([0. , 2.4])}


------------------------------------------------------------

[2.50s] Computing optimal path...

------------------------------------------------------------

A* Path       --> Nodes Explored: 411, Path Length: 6.30, Time: 1.28ms
Dijkstra Path --> Nodes Explored: 5664, Path Length: 7.20, Time: 14.92ms

A* Path       --> Nodes Explored: 159, Path Length: 3.10, Time: 0.47ms
Dijkstra Path --> Nodes Explored: 1677, Path Length: 3.10, Time: 4.28ms

[5.90s] Has material MAT009 been picked up? (yes/no):
```

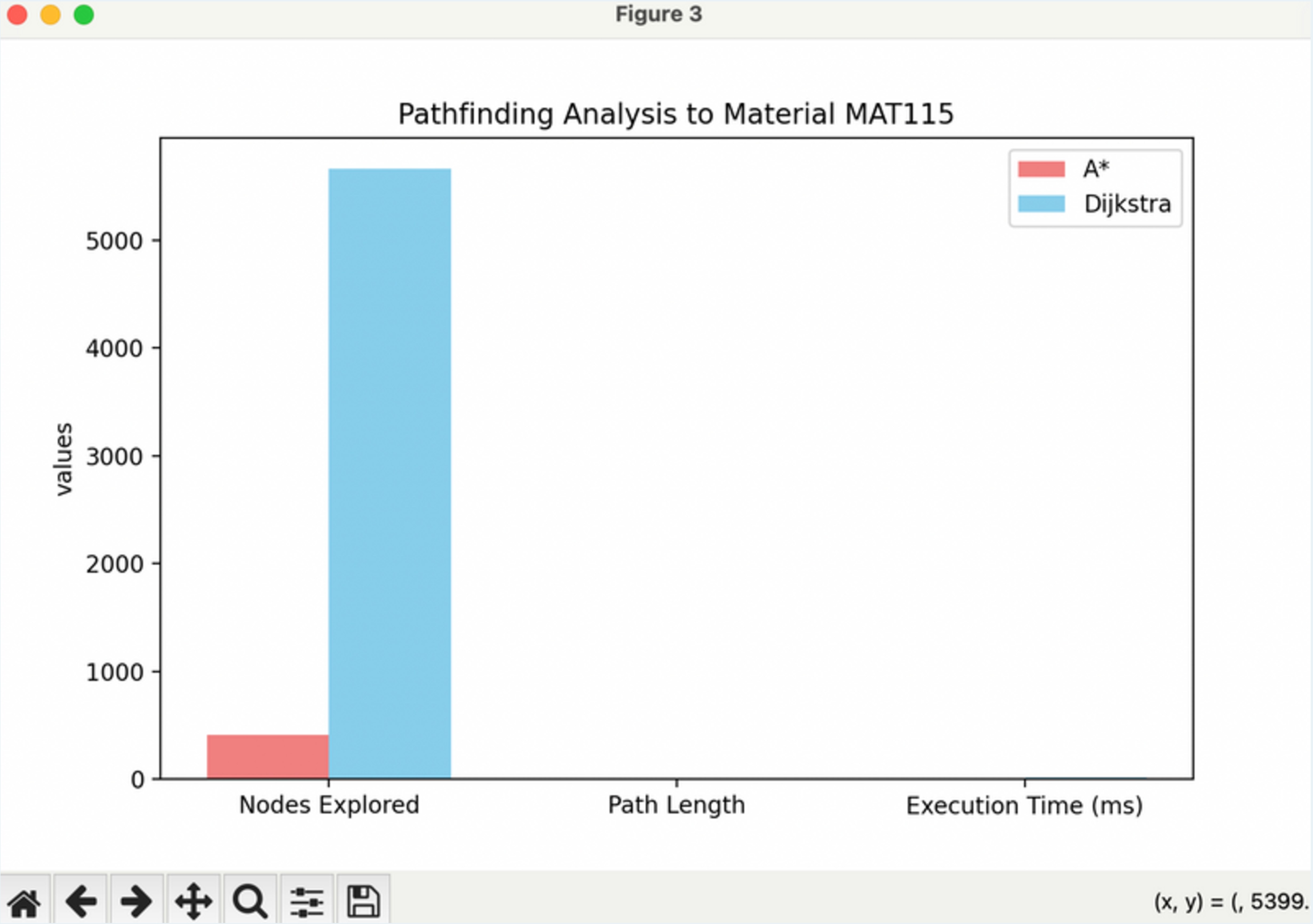


Figure 3
Pathfinding Analysis to Material MAT115
A*
Dijkstra
values
5000
4000
3000
2000
1000
0
Nodes Explored
Path Length
Execution Time (ms)
(x, y) = (, 5399.

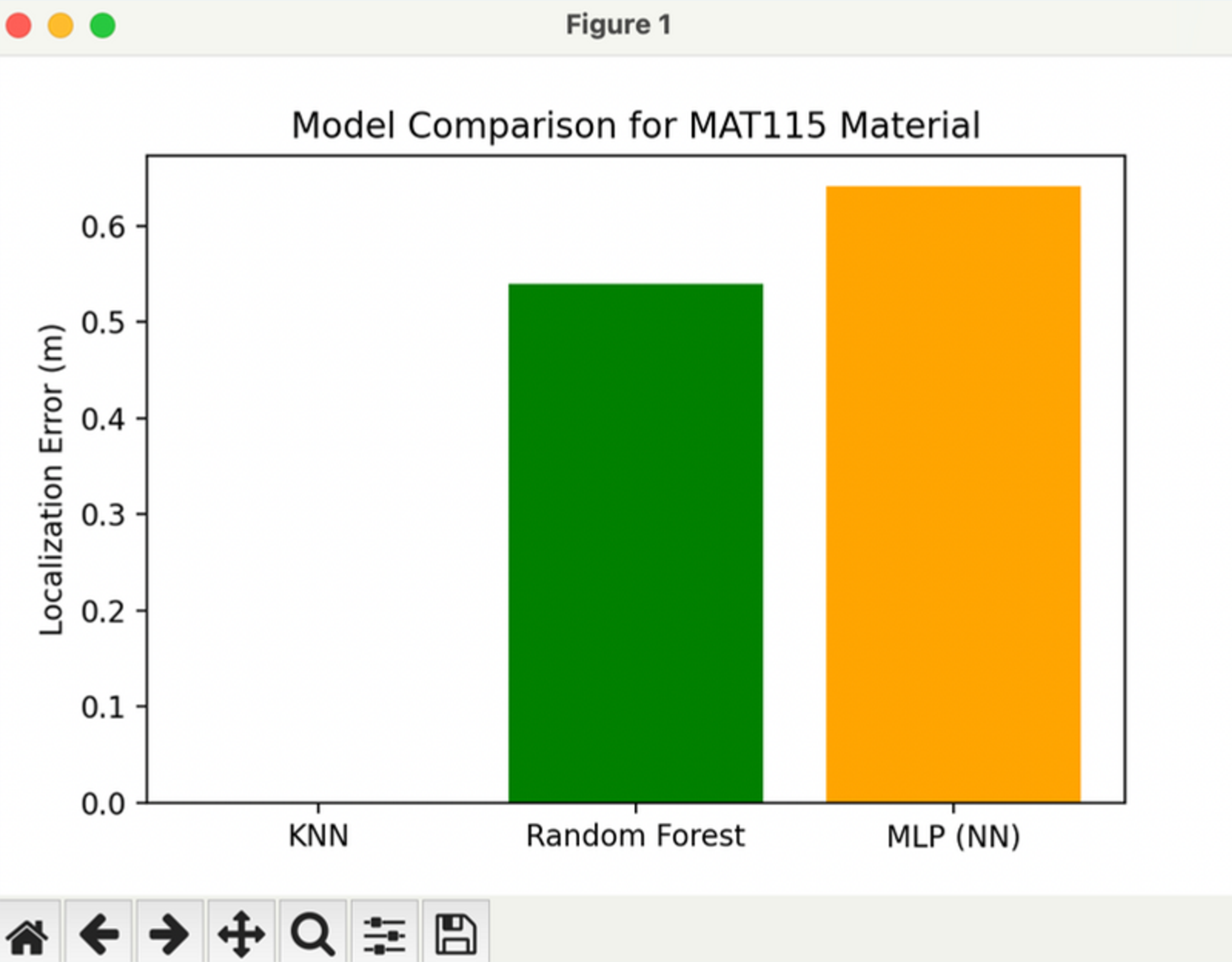
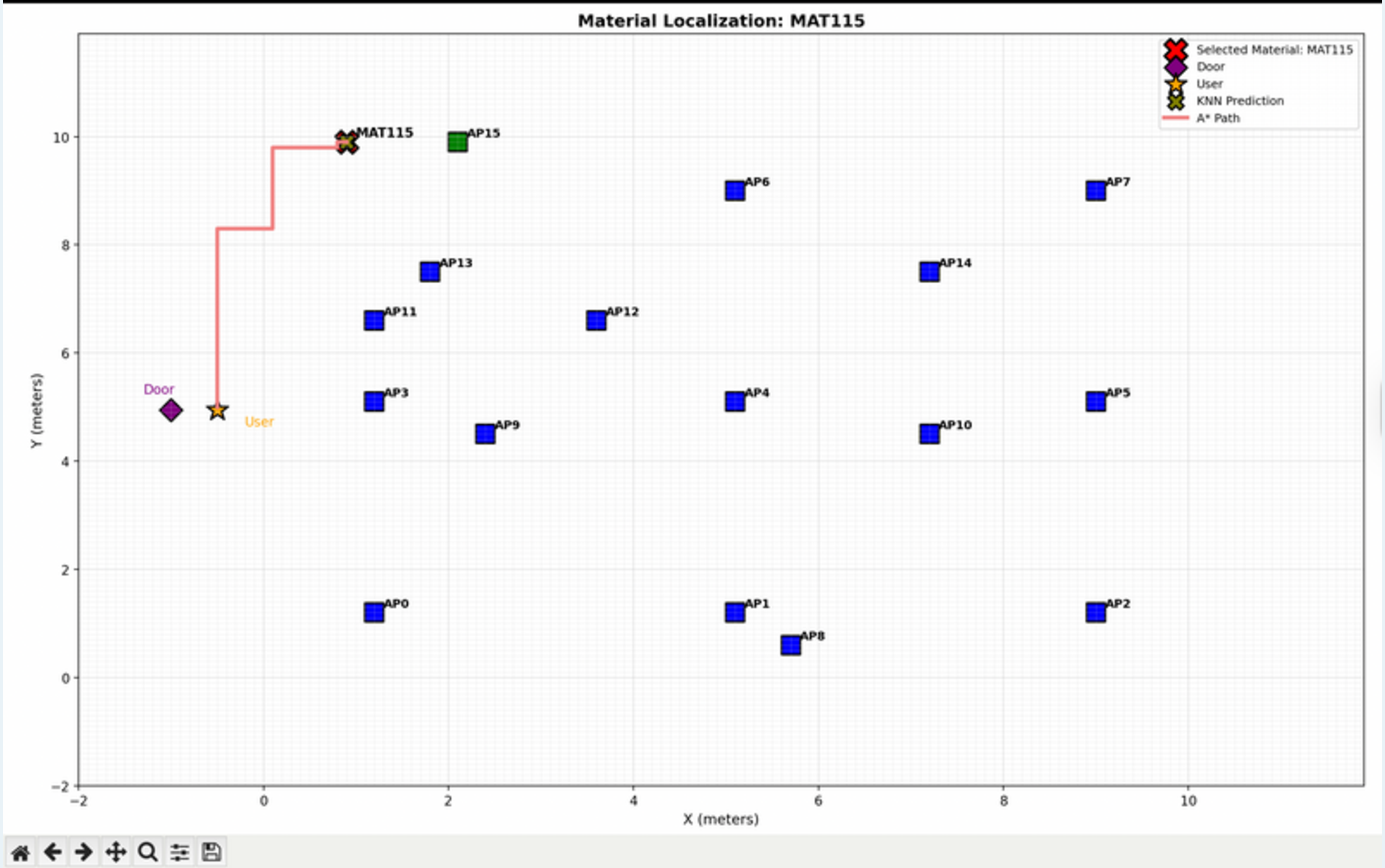


Figure 1
Model Comparison for MAT115 Material
Localization Error (m)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
KNN
Random Forest
MLP (NN)



Material Localization: MAT115
Selected Material: MAT115
Door
User
KNN Prediction
A* Path
MAT115
AP15
AP6
AP7
AP13
AP14
AP11
AP12
AP3
AP4
AP5
AP9
AP10
AP0
AP1
AP2
AP8
Door
User
Y (meters)
X (meters)

Resource Allocation & Communication

Node Definition

Edge & cloud with specs: CPU, memory, throughput, uplink/downlink delays,gw_id

Task Modeling

Tasks generated via material_task_source(), not just a simple generator

Scheduling Policies (Scheduler.decide)

FCFS → Always to cloud

Load Balancing (LB) →Based on Shortest estimated finish time

Latency-Aware → based on end-to-end latency

Tracks arrivals, route counts, completed tasks, latencies

Additional Features

MockBus → Simulates publish/subscribe communication

Integration → Linked with Person 2’s processing flow outputs

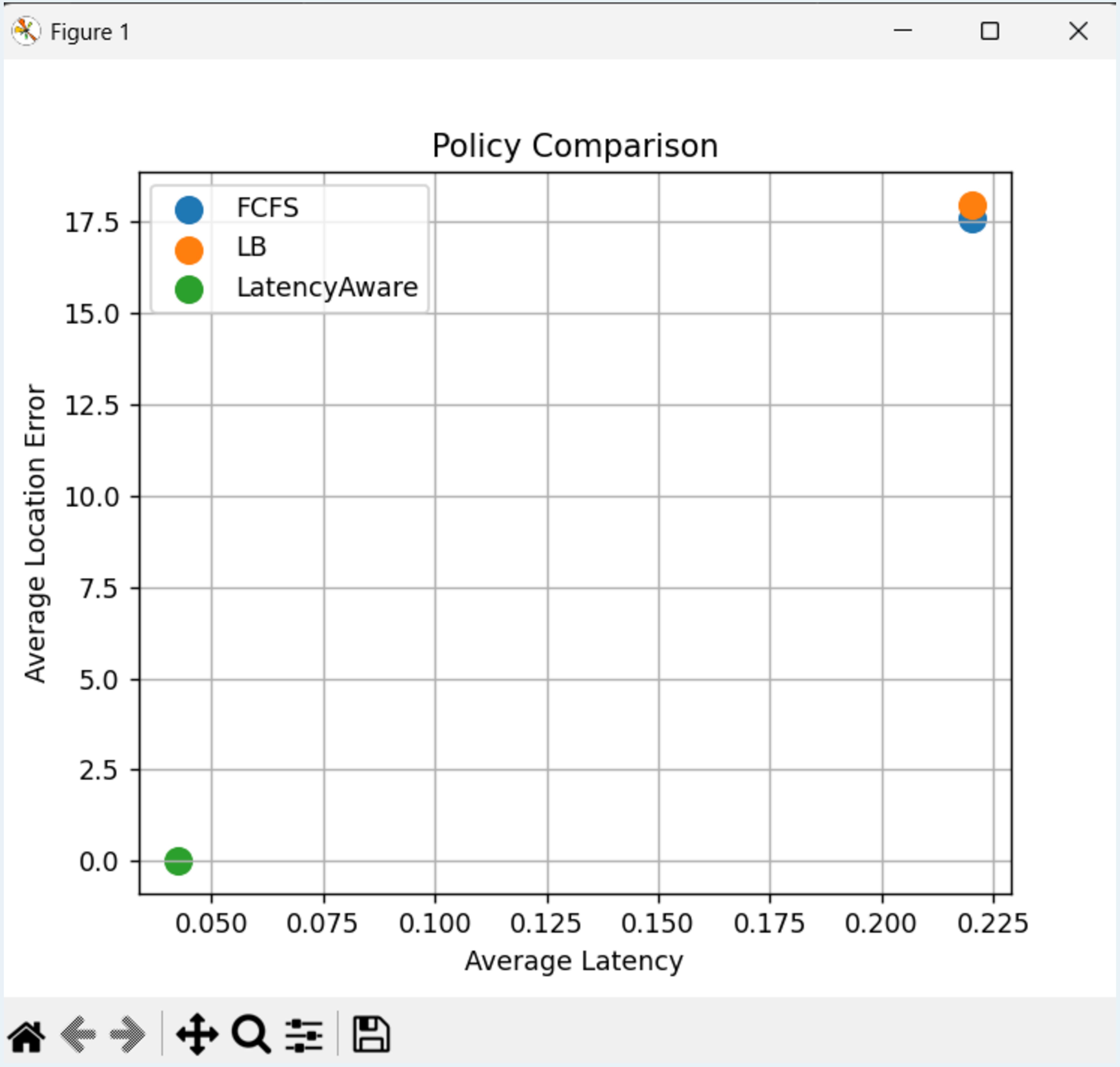


Figure 1
Policy Comparison
FCFS
LB
LatencyAware
Average Location Error
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0.225
Average Latency

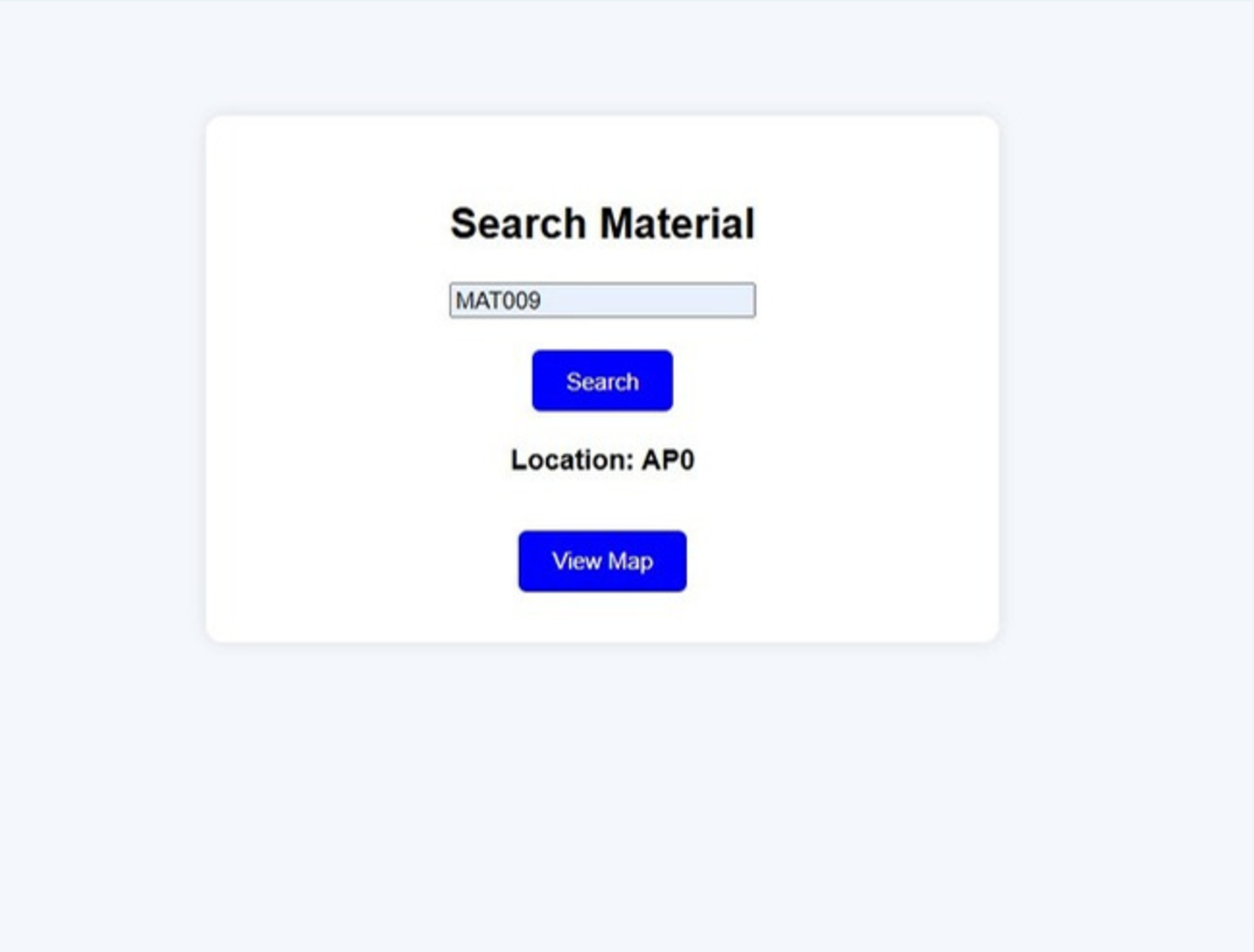
User Interaction

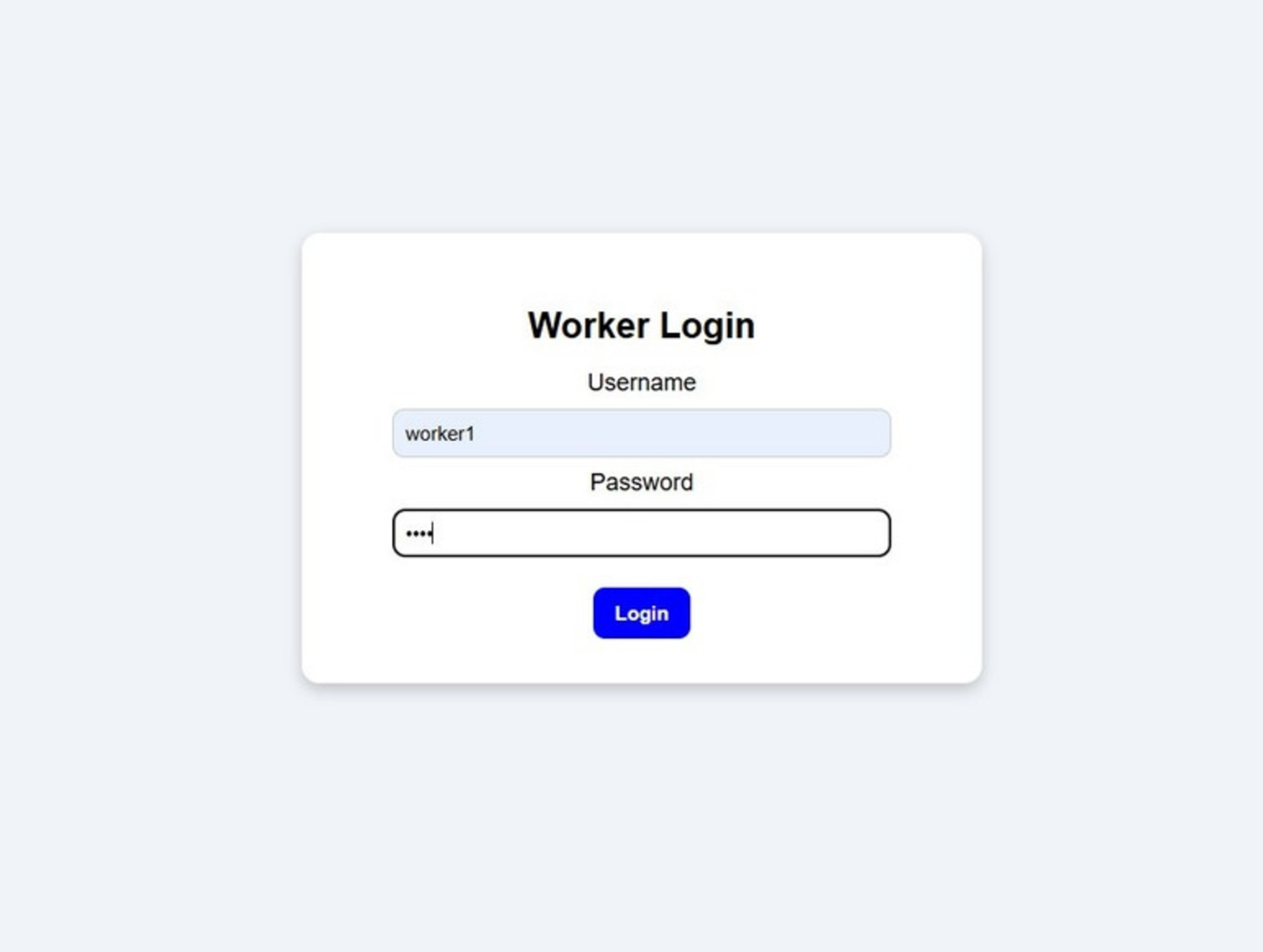
Implemented :

- A Login page for the workers
- Resource search page which gives instant location
- Manual “Item found button”
- Resource location Map
- Integrated with backend

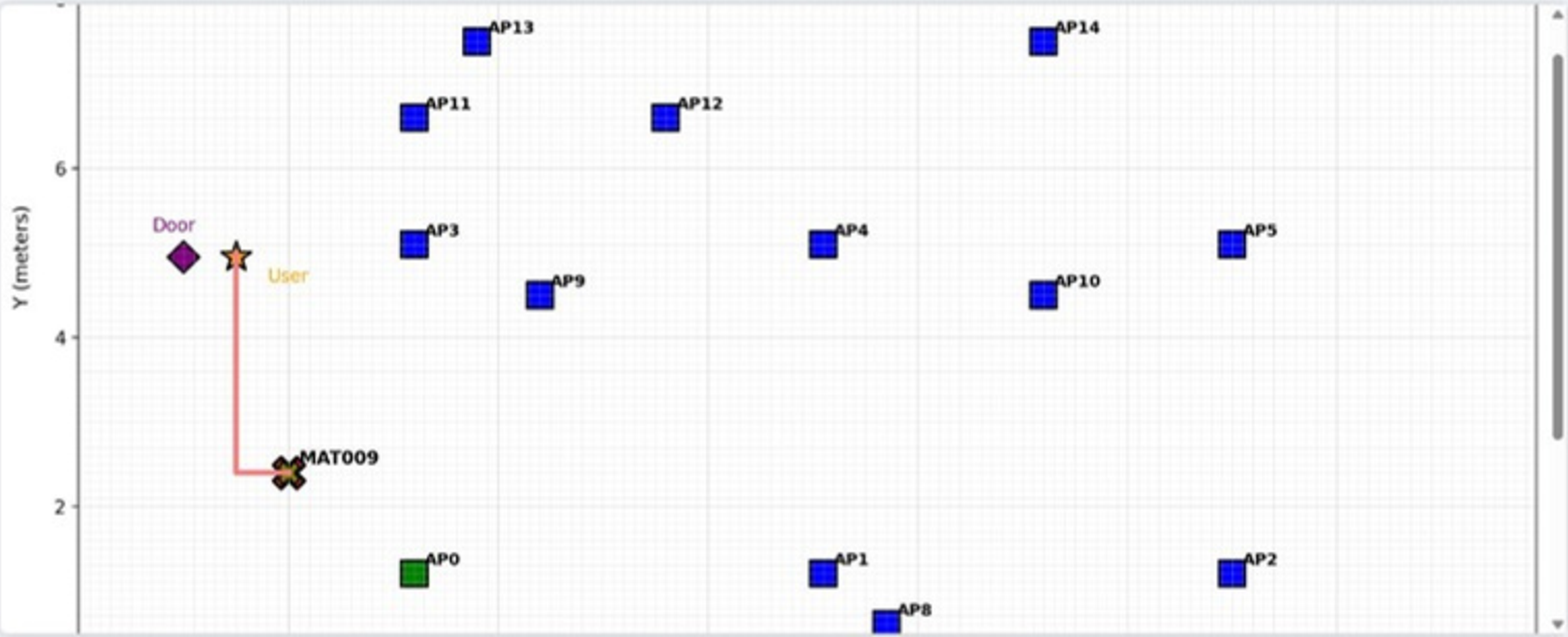
Tools Used :

FLASK,HTML,CSS,MATPLOTLIB,PYTHON


Search Material
MAT009
Search
Location: AP0
View Map


Worker Login
Username
worker1
Password
Login

Allocated Resource:



Back to Search

Mark as Picked