Pushing Simulation Platform

1. Python Backend for Simulation Logic

   - Install Dependencies:

     pip install flask flask-sqlalchemy

   - Python Code (app.py):

   ```
   from flask import Flask, jsonify, request
   from flask_sqlalchemy import SQLAlchemy
   import math

   app = Flask(__name__)
   app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///simulation.db'
   db = SQLAlchemy(app)

   class Object(db.Model):
       id = db.Column(db.Integer, primary_key=True)
       name = db.Column(db.String(100), nullable=False)
       mass = db.Column(db.Float, nullable=False)
       position_x = db.Column(db.Float, nullable=False)
       position_y = db.Column(db.Float, nullable=False)
       velocity_x = db.Column(db.Float, nullable=False)
       velocity_y = db.Column(db.Float, nullable=False)

   db.create_all()
   ```

```
def apply_force(obj, force_x, force_y):
    acceleration_x = force_x / obj.mass
    acceleration_y = force_y / obj.mass

    obj.velocity_x += acceleration_x
    obj.velocity_y += acceleration_y

    obj.position_x += obj.velocity_x
    obj.position_y += obj.velocity_y

@app.route('/api/objects', methods=['GET'])
def get_objects():
    objects = Object.query.all()
    return jsonify([{
        'id': obj.id,
        'name': obj.name,
        'mass': obj.mass,
        'position_x': obj.position_x,
        'position_y': obj.position_y,
        'velocity_x': obj.velocity_x,
        'velocity_y': obj.velocity_y
    } for obj in objects])

@app.route('/api/objects/<int:obj_id>/apply_force', methods=['POST'])
def apply_force_to_object(obj_id):
    obj = Object.query.get_or_404(obj_id)
```

```
    data = request.get_json()
    force_x = data.get('force_x', 0)
    force_y = data.get('force_y', 0)

    apply_force(obj, force_x, force_y)
    db.session.commit()
    return jsonify({
        'id': obj.id,
        'name': obj.name,
        'position_x': obj.position_x,
        'position_y': obj.position_y,
        'velocity_x': obj.velocity_x,
        'velocity_y': obj.velocity_y
    })

if __name__ == '__main__':
    app.run(debug=True)
```

2. SQL Database Setup

- SQL Schema Example:

```
CREATE TABLE honeypot_logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    ip_address TEXT NOT NULL,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    action TEXT
```

);

```

3. Frontend Interface (HTML5, JavaScript)

- HTML5 & JavaScript Code:

```

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Pushing Simulation</title>

<style>

body { font-family: Arial, sans-serif; background-color: #f4f4f4; text-align: center; }

table { width: 50%; margin: 20px auto; border-collapse: collapse; }

th, td { padding: 8px; border: 1px solid #ddd; text-align: left; }

th { background-color: #4CAF50; color: white; }

button { padding: 10px 20px; background-color: #4CAF50; color: white; border: none; cursor: pointer; }

button:hover { background-color: #45a049; }

</style>

</head>

<body>

<h1>Pushing Simulation Platform</h1>

<h2>Objects in Simulation</h2>

<table id="objectsTable">

```
    <thead>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Position (X, Y)</th>
            <th>Velocity (X, Y)</th>
            <th>Mass</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody></tbody>
</table>

<h2>Apply Force to Object</h2>
<label for="force_x">Force X:</label>
<input type="number" id="force_x" value="0"><br>
<label for="force_y">Force Y:</label>
<input type="number" id="force_y" value="0"><br>
<label for="object_id">Object ID:</label>
<input type="number" id="object_id" value="1"><br>
<button onclick="applyForce()">Apply Force</button>

<script>
    async function fetchObjects() {
        const response = await fetch('http://127.0.0.1:5000/api/objects');
        const objects = await response.json();
        const tableBody = document.querySelector('#objectsTable tbody');
```

```
    tableBody.innerHTML = '';
    objects.forEach(obj => {
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>${obj.id}</td>
            <td>${obj.name}</td>
            <td>(${obj.position_x}, ${obj.position_y})</td>
            <td>(${obj.velocity_x}, ${obj.velocity_y})</td>
            <td>${obj.mass}</td>
            <td><button onclick="applyForceToObject(${obj.id})">Apply Force</button></td>
        `;
        tableBody.appendChild(row);
    });
}

async function applyForce() {
    const forceX = document.getElementById('force_x').value;
    const forceY = document.getElementById('force_y').value;
    const objectId = document.getElementById('object_id').value;

    const response = await fetch(`http://127.0.0.1:5000/api/objects/${objectId}/apply_force`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ force_x: parseFloat(forceX), force_y: parseFloat(forceY) })
    });
```

```
            const updatedObject = await response.json();
            alert(`Object ID ${updatedObject.id} updated. New position: (${updatedObject.position_x}, ${updatedObject.position_y})`);
            fetchObjects();
        }

        fetchObjects();
    </script>
</body>
</html>
```

4. Bash Automation for Monitoring and Data Collection

- Bash Script for Data Monitoring (monitor.sh):

```
#!/bin/bash
while true; do
    python3 fetch_simulation_data.py
    sleep 10
done
```

- Python Data Fetch Script (fetch_simulation_data.py):

```
import sqlite3
```

```
    conn = sqlite3.connect('simulation.db')

    cursor = conn.cursor()

    cursor.execute("SELECT * FROM object")

    rows = cursor.fetchall()

    for row in rows:

        print(f"Object ID: {row[0]}, Name: {row[1]}, Position: ({row[2]}, {row[3]}), Velocity: ({row[4]}, {row[5]})")

    conn.close()
```