1.

fastapi

uvicorn

sqlalchemy

pydantic

2.

from sqlalchemy import create\_engine, Column, Integer, String, Float

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker

DATABASE\_URL = "sqlite:///./buses.db"

engine = create\_engine(DATABASE\_URL, connect\_args={"check\_same\_thread": False})

SessionLocal = sessionmaker(bind=engine, autoflush=False, autocommit=False)

Base = declarative\_base()  
  
3.

from sqlalchemy import Column, Integer, String, Float

from database import Base

class Bus(Base):

    \_\_tablename\_\_ = "buses"

    id = Column(Integer, primary\_key=True, index=True)

    bus\_id = Column(String, unique=True, index=True)

    latitude = Column(Float)

    longitude = Column(Float)

    occupancy = Column(Integer)  # number of passengers

    route = Column(String, nullable=True)

4.

import requests

import time

BASE\_URL = "http://127.0.0.1:8000"

def test\_update\_bus(bus\_id, latitude, longitude, occupancy, route=None):

    data = {

        "bus\_id": bus\_id,

        "latitude": latitude,

        "longitude": longitude,

        "occupancy": occupancy,

        "route": route

    }

    response = requests.post(f"{BASE\_URL}/update", json=data)

    print(f"Update bus {bus\_id}: {response.status\_code} - {response.json()}")

def test\_get\_bus(bus\_id):

    response = requests.get(f"{BASE\_URL}/bus/{bus\_id}")

    print(f"Get bus {bus\_id}: {response.status\_code} - {response.json()}")

def test\_get\_all\_buses():

    response = requests.get(f"{BASE\_URL}/buses")

    print(f"Get all buses: {response.status\_code} - {response.json()}")

if \_\_name\_\_ == "\_\_main\_\_":

    # Simulate continuous bus updates

    test\_update\_bus("bus123", 12.34, 56.78, 20, "RouteA")

    time.sleep(1)

    test\_update\_bus("bus456", 23.45, 67.89, 15, "RouteB")

    time.sleep(1)

    test\_update\_bus("bus123", 12.35, 56.79, 18, "RouteA")  # Update existing bus

    # Test get endpoints

    test\_get\_bus("bus123")

    test\_get\_bus("bus999")  # Non-existent bus

    test\_get\_all\_buses()

5.

import requests

BASE\_URL = "http://127.0.0.1:8000"

def test\_invalid\_bus\_id():

    response = requests.get(f"{BASE\_URL}/bus/invalid\_bus\_id")

    print(f"Get invalid bus\_id: {response.status\_code} - {response.json()}")

def test\_missing\_fields\_update():

    # Missing 'bus\_id'

    data = {

        "latitude": 12.34,

        "longitude": 56.78,

        "occupancy": 20,

        "route": "RouteA"

    }

    response = requests.post(f"{BASE\_URL}/update", json=data)

    print(f"Update missing bus\_id: {response.status\_code} - {response.json()}")

    # Missing 'latitude'

    data = {

        "bus\_id": "bus789",

        "longitude": 56.78,

        "occupancy": 20,

        "route": "RouteA"

    }

    response = requests.post(f"{BASE\_URL}/update", json=data)

    print(f"Update missing latitude: {response.status\_code} - {response.json()}")

if \_\_name\_\_ == "\_\_main\_\_":

    test\_invalid\_bus\_id()

    test\_missing\_fields\_update()

6.

import requests

response = requests.post("http://127.0.0.1:8000/update", json={

    "bus\_id": "bus1",

    "latitude": 13.0827,

    "longitude": 80.2707,

    "occupancy": 10,

    "route": "Route A"

})

print(response.json())

7.

import requests

response = requests.post("http://127.0.0.1:8000/update", json={

    "bus\_id": "bus1",

    "latitude": 13.0827,

    "longitude": 80.2707,

    "occupancy": 10,

    "route": "Route A"

})

print(response.json())

8.

import { MapContainer, TileLayer, Marker, Popup, useMap } from "react-leaflet";

import "leaflet/dist/leaflet.css";

import L from "leaflet";

import axios from "axios";

import { useEffect, useState } from "react";

const getBusIcon = (occupancy) => {

  let color;

  if (occupancy <= 20) color = "green";

  else if (occupancy <= 40) color = "yellow";

  else color = "red";

  return new L.DivIcon({

    html: '<div style="background-color: ' + color + '; width: 32px; height: 32px; border-radius: 50%; display: flex; justify-content: center; align-items: center; color: black; font-weight: bold; border: 2px solid white;">🚌</div>',

    className: "",

    iconSize: [32, 32],

    iconAnchor: [16, 16],

  });

};

function AutoFitBounds({ buses }) {

  const map = useMap();

  useEffect(() => {

    if (buses.length === 0) return;

    const bounds = L.latLngBounds(buses.map((b) => [b.latitude, b.longitude]));

    map.fitBounds(bounds, { padding: [50, 50] });

  }, [buses, map]);

  return null;

}

export default function BusMap() {

  const [buses, setBuses] = useState([]);

  useEffect(() => {

    const fetchBuses = async () => {

      try {

        const res = await axios.get("http://127.0.0.1:8000/buses");

        setBuses(res.data);

      } catch (err) {

        console.error("Error fetching buses:", err);

      }

    };

    fetchBuses();

    const interval = setInterval(fetchBuses, 5000); // refresh every 5s

    return () => clearInterval(interval);

  }, []);

  return (

    <MapContainer

      center={[13.0827, 80.2707]} // Default center (Chennai)

      zoom={12}

      style={{ height: "70vh", width: "100%", marginTop: "20px" }}

    >

      <TileLayer

        url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

        attribution="&copy; OpenStreetMap contributors"

      />

      <AutoFitBounds buses={buses} />

      {buses.map((bus) => (

        <Marker

          key={bus.bus\_id}

          position={[bus.latitude, bus.longitude]}

          icon={getBusIcon(bus.occupancy)}

        >

          <Popup>

            <b>Bus ID:</b> {bus.bus\_id} <br />

            <b>Occupancy:</b> {bus.occupancy} {bus.occupancy <= 20 ? "🟢" : bus.occupancy <= 40 ? "🟡" : "🔴"} <br />

            <b>Route:</b> {bus.route || "N/A"}

          </Popup>

        </Marker>

      ))}

    </MapContainer>

  );

}

9.

import axios from "axios";

import { useEffect, useState } from "react";

export default function BusTable() {

  const [buses, setBuses] = useState([]);

  useEffect(() => {

    const fetchBuses = async () => {

      try {

        const res = await axios.get("http://127.0.0.1:8000/buses");

        setBuses(res.data);

      } catch (err) {

        console.error("Error fetching buses:", err);

      }

    };

    fetchBuses();

    const interval = setInterval(fetchBuses, 5000);

    return () => clearInterval(interval);

  }, []);

  return (

    <table

      border="1"

      cellPadding="10"

      style={{ marginTop: "20px", width: "100%", borderCollapse: "collapse" }}

    >

      <thead>

        <tr style={{ backgroundColor: "#f2f2f2" }}>

          <th>Bus ID</th>

          <th>Latitude</th>

          <th>Longitude</th>

          <th>Occupancy</th>

          <th>Route</th>

        </tr>

      </thead>

      <tbody>

        {buses.map((bus) => (

          <tr key={bus.bus\_id}>

            <td>{bus.bus\_id}</td>

            <td>{bus.latitude.toFixed(4)}</td>

            <td>{bus.longitude.toFixed(4)}</td>

            <td>{bus.occupancy}</td>

            <td>{bus.route || "N/A"}</td>

          </tr>

        ))}

      </tbody>

    </table>

  );

}

10.

import BusMap from "./components/BusMap";

import BusTable from "./components/BusTable";

function App() {

  return (

    <div style={{ padding: "20px" }}>

      <h1 style={{ textAlign: "center" }}>🚍 Real-Time Bus Tracker</h1>

      <BusMap />

      <BusTable />

    </div>

  );

}}}