Driver

Admin

Farmer

Driver Client
(Mobile)

Admin Client
(Web App)

Farmer Client

Modular Monolithic

User Module

Auth Module

Production Module

Fleet Module

Routing Module

Farmer Module

Notification Module

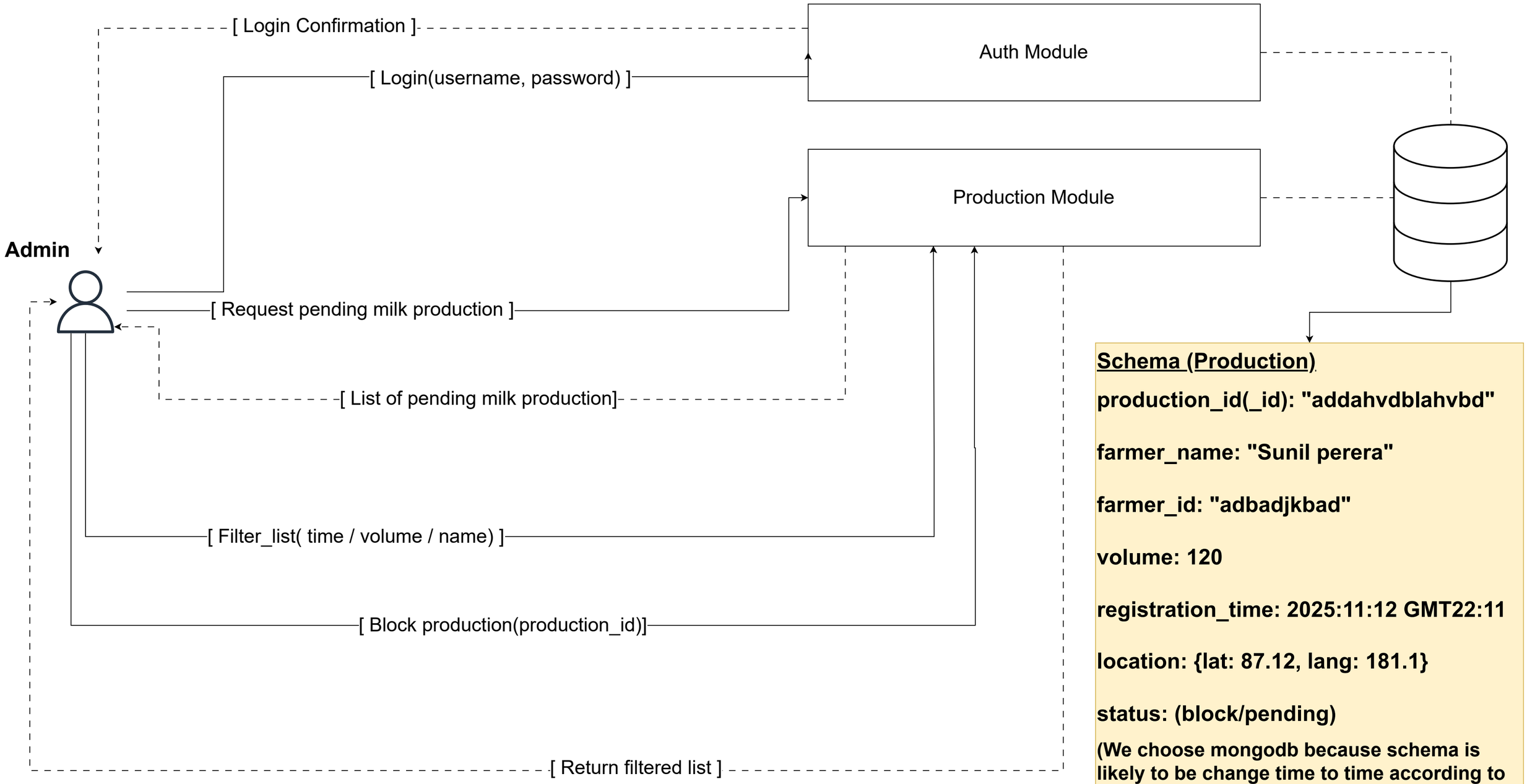
Feedback Module

Analytics Module

Mongo Database

Route Optimization Service (FAST API)

System Architecture



Schema (Production)

production_id(_id): "addahvdblahvbd"

farmer_name: "Sunil perera"

farmer_id: "adbadjkbad"

volume: 120

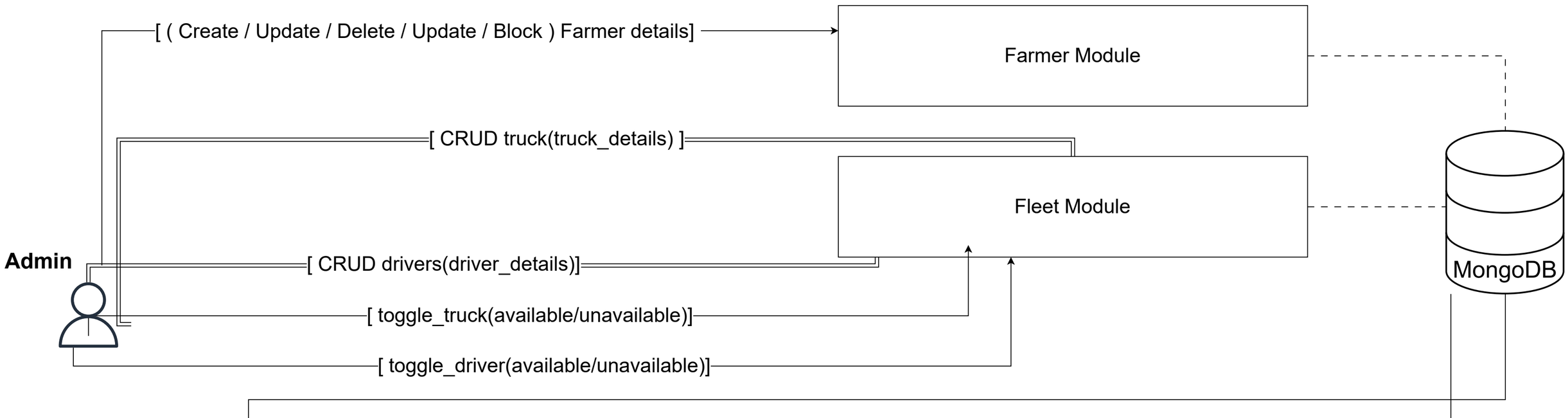
registration_time: 2025:11:12 GMT22:11

location: {lat: 87.12, lang: 181.1}

status: (block/pending)

(We choose mongodb because schema is likely to be change time to time according to changing requirements of the company)

System Architecture



Schema (Truck)

truck_id(_id): "addahvdblahvbd"

plate_no: "WP CAB-1234"

max_capacity: 2500L

depot_location: "uswewa"

status: (available/unavailable/inService)

registration_time: 2025:11:12 GMT22:11

model: "IZUSU truck"

Schema (Driver)

driver_id: "addahvdblahvbd"

name: "Sumanasiri"

status: (available/unavailable/inactive)

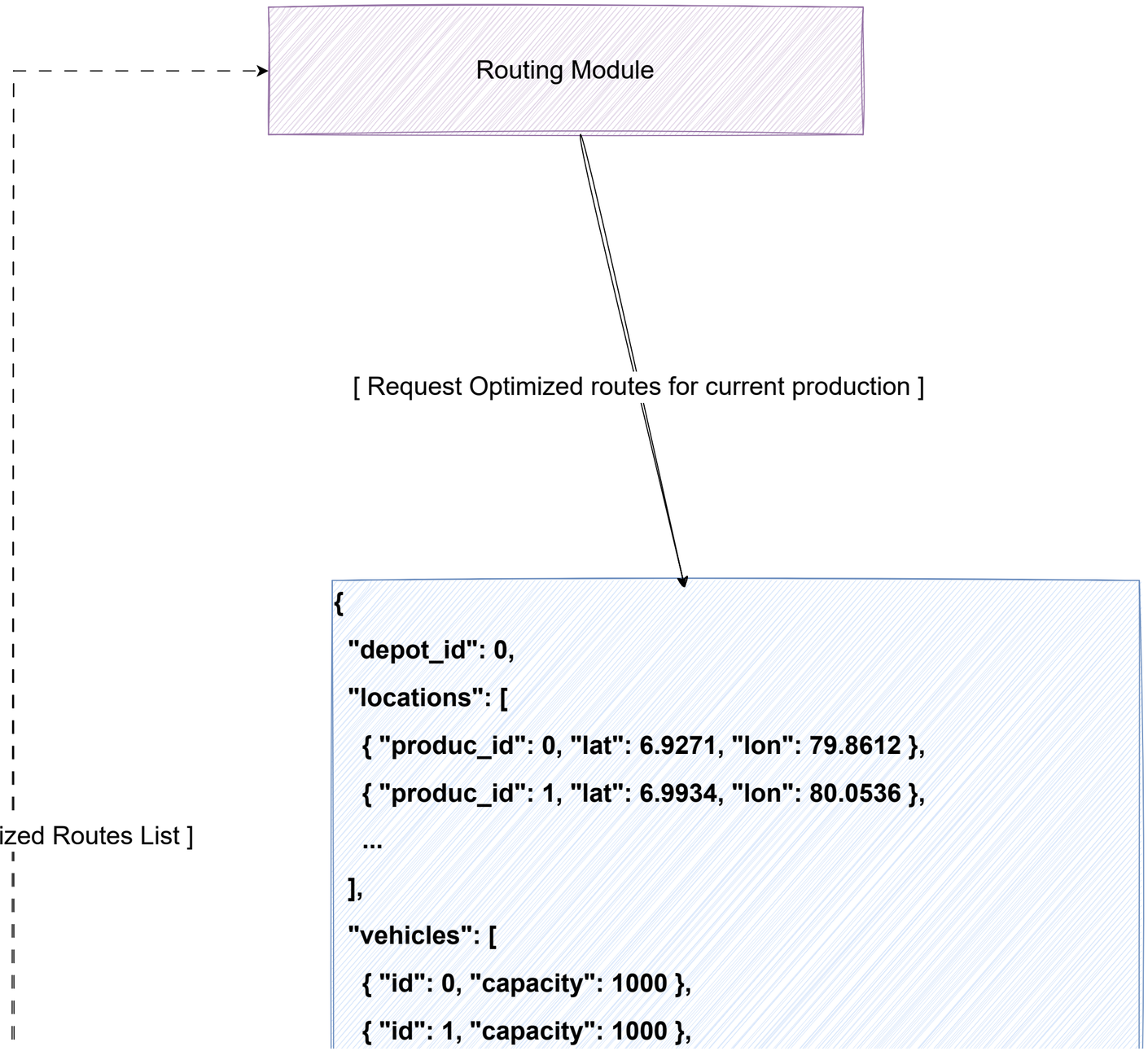
Summary

Node.js: `routing.service.ts`

- **Fetch pending productions**
- **Fetch available trucks**
- **Build VRP input JSON**
- **Send POST request → Python VRP service**
- **Receive optimized routes**
- **Save routes to DB**
- **Assign routes to trucks/drivers**

Python: `vrp_engine.py`

- **Receive JSON input**
- **Run OR-Tools VRP**
- **Compute stop sequences**
- **Return route results**

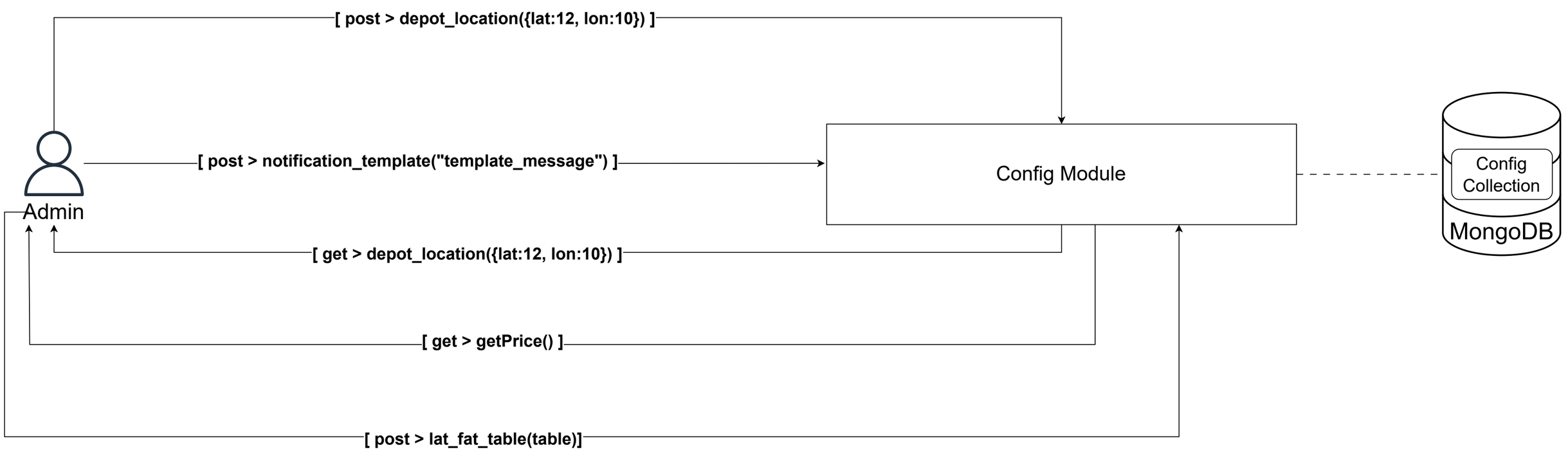


```
  { "id": 2, "capacity": 1000 },
  { "id": 3, "capacity": 1000 }
 ]
}
```


Map Service (Python)


```
{
 "routes": [
  {
   "vehicle_id": 0,
   "stops": [
    { "order": 0, "pro_id": 0, "lat": 6.9271, "lon": 79.8612 },
    { "order": 1, "pro_id": 1, "lat": 6.9934, "lon": 80.0536 },
    { "order": 2, "pro_id": 3, "lat": 7.2906, "lon": 80.6337 }
   ]
  },
  {
   "vehicle_id": 1,
   "stops": [...]
  }
 ]
}
```

System Architecture



Config Schema

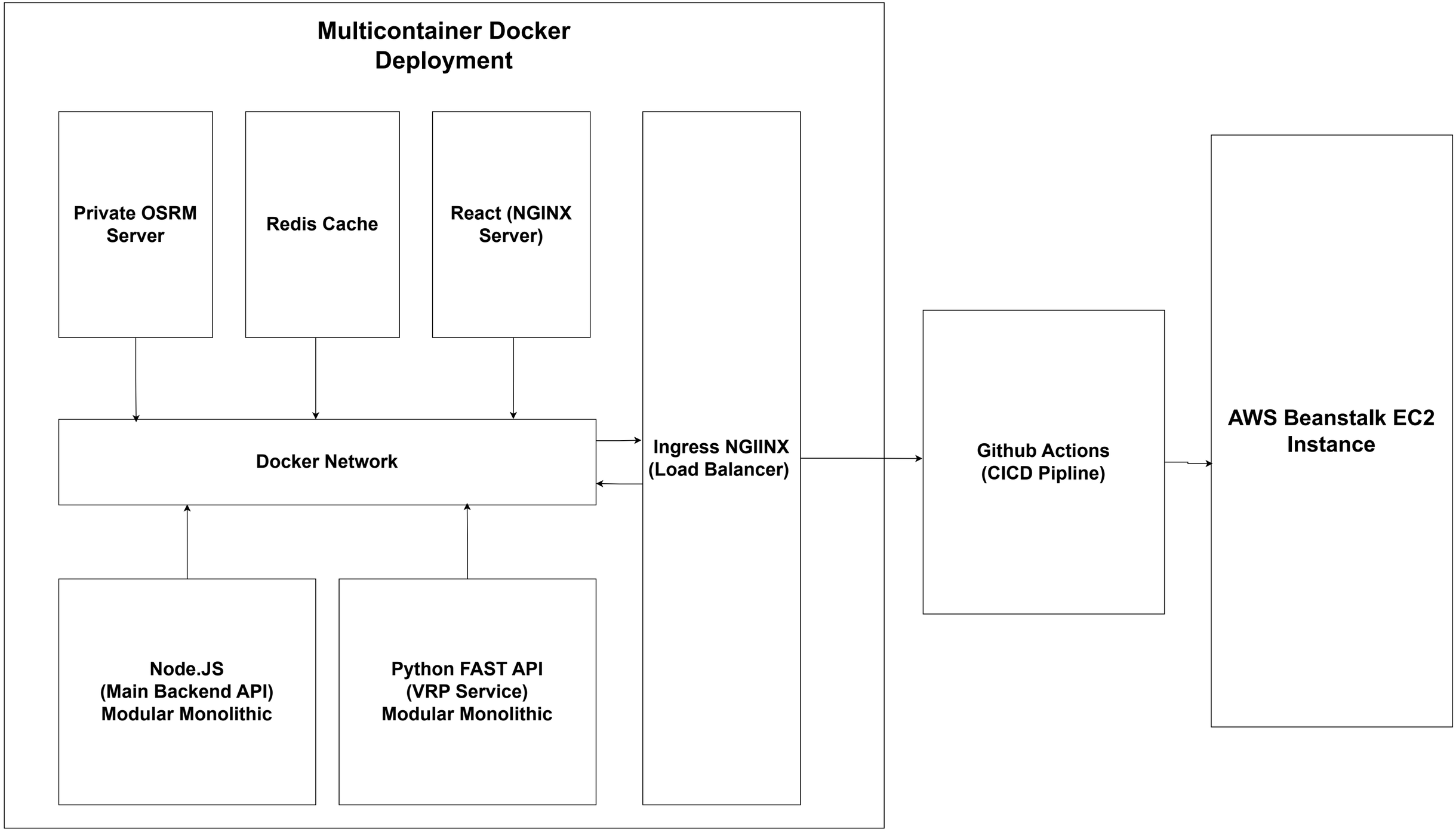
_id: "addahvdblahvbd"

depot_location: { lat: 11, lon: 12}

notification_template: "Your milk production of [VOLUME] liters is scheduled for pickup."

lat_fat_table: {[], [], [], []}

Synchronous microservices architecture using Docker and AWS Elastic Beanstalk



Modular Monolithic

Node.js Main Backend API
(RESTFUL)

- User Module
- Production Module
- Auth Module
- Fleet Module
- Routing Module
- Farmer Module
- Report Module
- Analytics Module
- Config Module

Modular Monolithic

Route Optimization Service (FAST API)
(RESTFULL)

- VRP Module

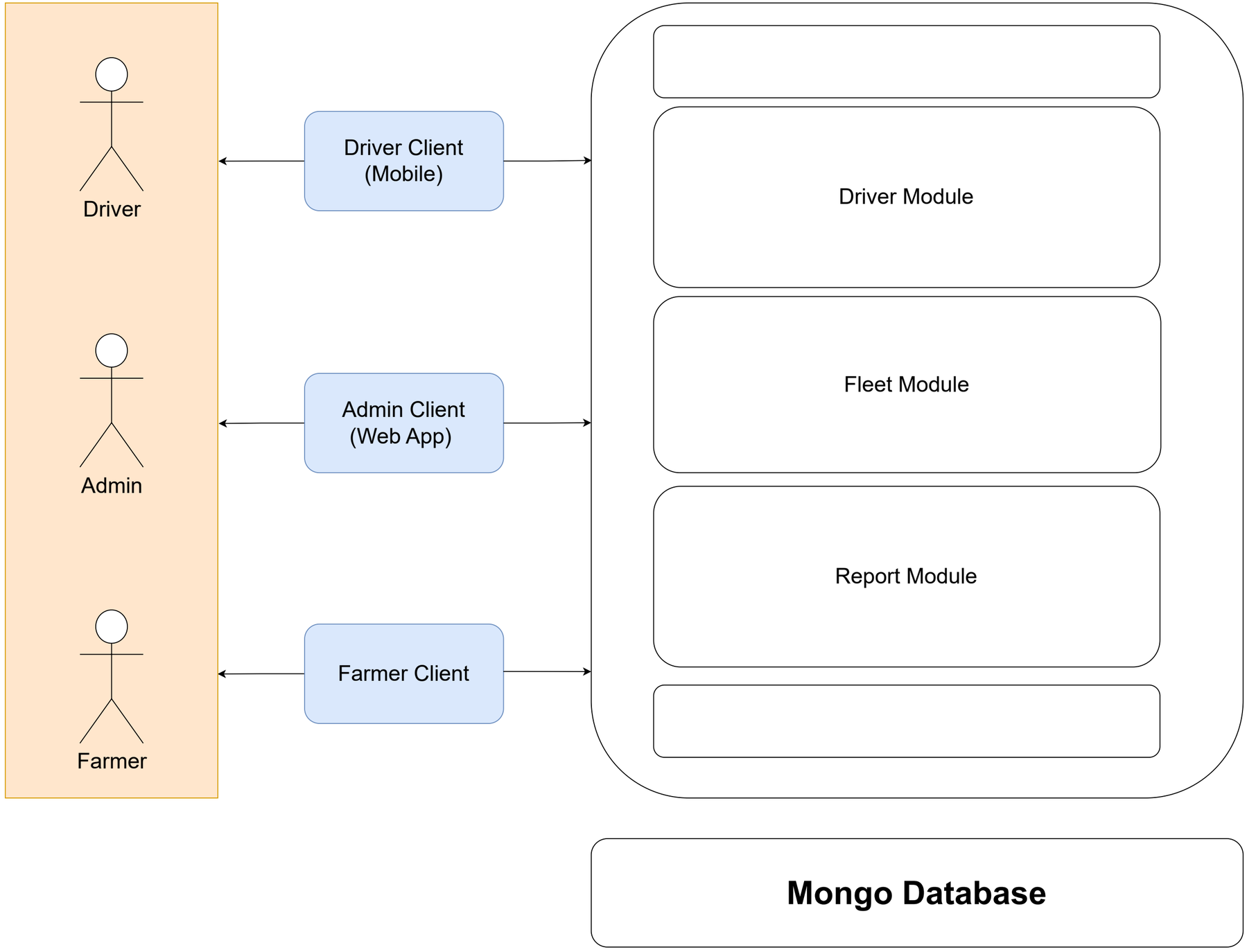
Mongo Database

Mongoose ORM

Driver Client
(Mobile)

Admin Client
(Web App)

Farmer Client
(Mobile)


Driver
Admin
Farmer
Driver Client
(Mobile)
Admin Client
(Web App)
Farmer Client
Driver Module
Fleet Module
Report Module
Mongo Database

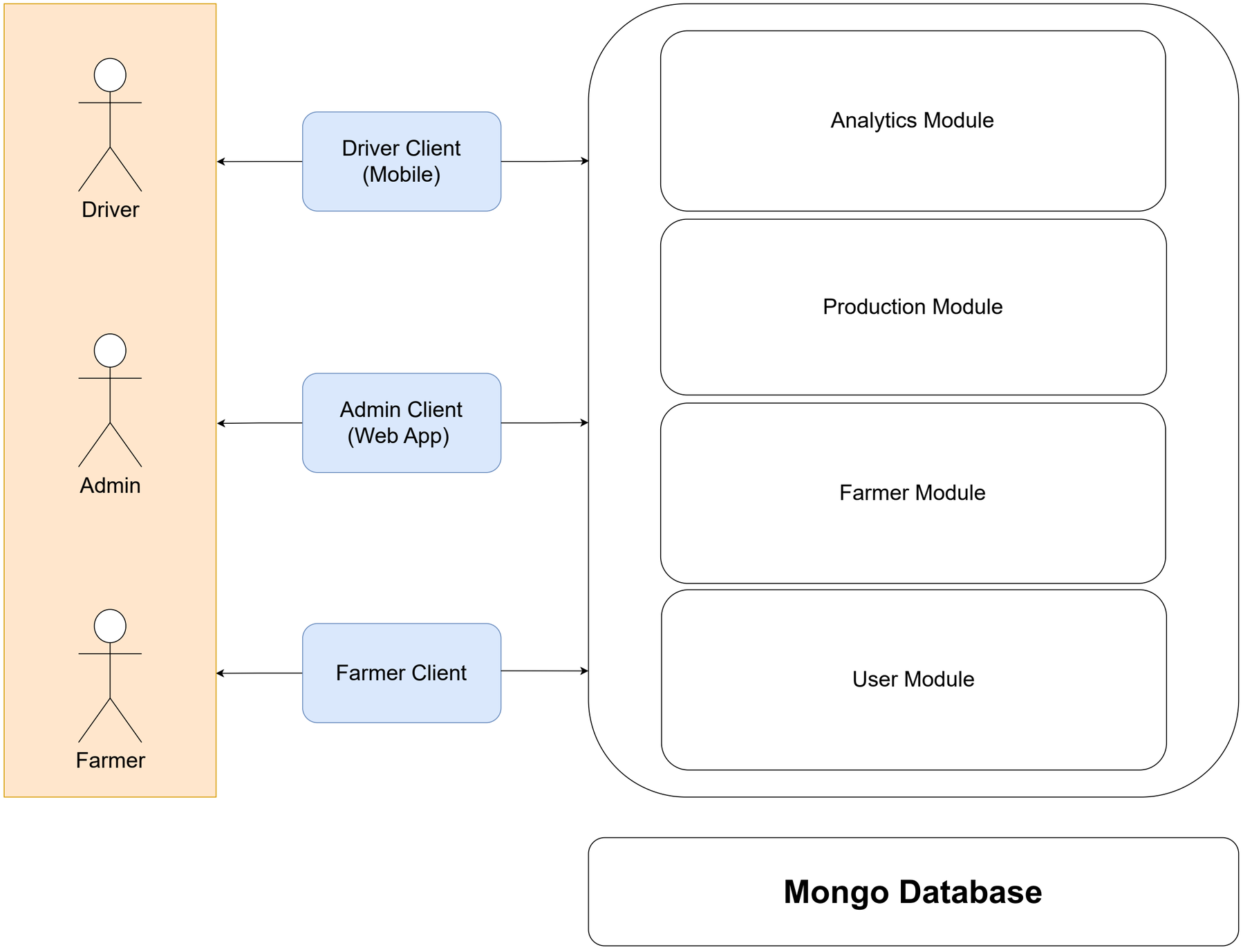
Dashboard - (Chart.js)

Total liters collected today	Total liters collected this month	Avg pickups per vehicle today	Total production pending today

Total liters per day [chart of last week]	Total distance (km) per day [chart of last week]	[Completed | Failed] production ratio [doughnut chart]
Total productions per day [chart of last week]	Some useful chart data	Some useful chart data

Backend for dashboard

/api/analytics/dashboard	**One single endpoint to fetch all dashboard details in a structured manner (JSON object) - use Chart.js**


Driver
Admin
Farmer
Driver Client
(Mobile)
Admin Client
(Web App)
Farmer Client
Analytics Module
Production Module
Farmer Module
User Module
Mongo Database