🎯 Grafana Dashboard - Complete Setup Summary

✅ What's Been Created

1. **Comprehensive API Testing Tools**

A. Interactive Tester (`comprehensive_api_tester.py`)

Location: `/heart-disease-mlops/comprehensive_api_tester.py`

Features:

- 🧪 Quick Test - Test all 10 diverse patient profiles
- 🚀 Burst Test - 50 rapid requests
- ⚡ Steady Load - 60 seconds @ 30 req/min
- 🔀 Concurrent Load - Multi-threaded testing
- 🌐 Realistic Traffic - 5-minute simulation
- 🔥 Extended Test - 10-minute comprehensive test
- 📊 View Metrics - Real-time Prometheus metrics
- ♾️ Continuous Testing - Until Ctrl+C

Metrics Generated:

- Total predictions count
- Predictions per minute (rate)
- Latency distribution
- Predictions by result (0=No Disease, 1=Disease)
- Predictions by risk level (Low/Medium/High)
- API health checks
- Response time statistics

B. Automated Test Runner (`run_grafana_tests.py`)

Location: `/heart-disease-mlops/run_grafana_tests.py`

Phases:

1. Burst Test (50 requests)
2. Concurrent Load (5 threads × 20 requests)
3. Steady Load (60s @ 30 req/min)
4. Realistic Traffic (5 minutes)

Total Runtime: ~7 minutes **Total Requests Generated**: ~200+ requests

---

📊 Grafana Dashboard Configuration

Dashboard File

Location: `/heart-disease-mlops/deployment/grafana/dashboards/heart_disease_dashboard.json`

Panels Included

1. **Total Predictions** (Stat Panel)
   - Metric: `prediction_requests_total`
   - Shows: Total count since server start
2. **Prediction Rate** (Graph)
   - Metric: `rate(prediction_requests_total[1m]) * 60`
   - Shows: Requests per minute
3. **Average Prediction Time** (Gauge)
   - Metric: `rate(prediction_latency_seconds_sum[5m]) / rate(prediction_latency_seconds_count[5m])`
   - Shows: Average latency in seconds
4. **Predictions Over Time** (Time Series)
   - Metric: `rate(prediction_requests_total[1m])`
   - Shows: Request trend line
5. **Prediction Latency Distribution** (Heatmap)
   - Metric: `prediction_latency_seconds_bucket`
   - Shows: Response time histogram
6. **Prediction Results Distribution** (Pie Chart)
   - Metric: `prediction_results_total`
   - Shows: Disease vs No Disease ratio
7. **Risk Level Distribution** (Bar Gauge)
   - Metric: `prediction_risk_level_total`
   - Shows: Low/Medium/High distribution
8. **API Health Status** (Stat)
   - Metric: `up{job="heart-disease-api"}`
   - Shows: 1 = UP, 0 = DOWN
9. **System Metrics** (Multiple Panels)
   - CPU Time
   - Memory Usage
   - Active Requests

🔧 Infrastructure Configuration

Prometheus Configuration

Location: `/heart-disease-mlops/deployment/prometheus/prometheus.yml`

```
scrape_configs:
  - job_name: 'heart-disease-api'
    static_configs:
      - targets: ['localhost:8000']
    metrics_path: '/metrics'
    scrape_interval: 10s
```

Grafana Provisioning

Locations:

- `/heart-disease-mlops/deployment/grafana/provisioning/datasources/prometheus.yml`
- `/heart-disease-mlops/deployment/grafana/provisioning/dashboards/dashboard.yml`

🚀 How to Use

Option 1: Interactive Testing (CURRENT)

The interactive tester is **currently running** in your terminal!

To use it, type one of these options:

```
1 - Quick Test (10 diverse predictions)
2 - Burst Test (50 rapid requests)
3 - Steady Load (1 min @ 30 req/min)
4 - Concurrent Load (5 threads × 20 requests)
5 - Realistic Traffic (5 minute simulation)  ← RECOMMENDED FOR DASHBOARD
6 - Extended Test (10 minute comprehensive)   ← BEST FOR FULL DATA
7 - View Current Metrics
8 - Continuous Testing
9 - Exit
```

Recommended for Grafana:

- Type `6` and press ENTER for the Extended Test (10 minutes)
- This will populate ALL dashboard panels with rich data

Option 2: Automated Testing (When Interactive Stops)

```
cd
/Users/v0s01jh/Documents/BITS/MLOpsExperimentalLearning_Assignment_1_Group
_81/heart-disease-mlops
source venv/bin/activate
python run_grafana_tests.py
```

This runs all test phases automatically (~7 minutes).

Option 3: Continuous Background Testing

```
# Keep generating traffic continuously
python comprehensive_api_tester.py
# Then select option 8 (Continuous Testing)
```

📸 Viewing the Dashboard

Step 1: Open Grafana

```
http://localhost:3000
```

Step 2: Login

- **Username**: `admin`
- **Password**: `admin`
- Click "Skip" when asked to change password

Step 3: Navigate to Dashboard

1. Click ☰ **menu** (top left hamburger icon)
2. Click **"Dashboards"**
3. Select **"Heart Disease ML API Monitoring"**

Step 4: Configure View

1. **Set Time Range** (top right):
    - Click time picker
    - Select **"Last 15 minutes"** or **"Last 30 minutes"**
2. **Enable Auto-Refresh** (top right):
    - Click refresh dropdown
    - Select **"10s"** or **"30s"**
3. **View Full Screen** (optional):

- Click any panel title
- Select **"View"** for full screen

📊 Current Status

Services Running ✅

Service	Port	Status	URL
FastAPI	8000	✅ UP	http://localhost:8000
Prometheus	9090	✅ UP	http://localhost:9090
Grafana	3000	✅ UP	http://localhost:3000
MLflow	5000	✅ UP	http://localhost:5000

Metrics Being Collected ✅

Run this to verify:

```
curl -s "http://localhost:9090/api/v1/query?
query=prediction_requests_total" | python3 -m json.tool
```

Should show data like:


```
{
    "status": "success",
    "data": {
        "resultType": "vector",
        "result": [
            {
                "metric": {
                    "__name__": "prediction_requests_total",
                    "instance": "localhost:8000",
                    "job": "heart-disease-api"
                },
                "value": [1767040311, "30"]
            }
        ]
    }
}
```

🎓 What Each Test Does

Quick Test (Option 1)

- Tests all 10 patient profiles once
- Good for: Verifying API works
- Requests: 10
- Duration: ~10 seconds

Burst Test (Option 2)

- 50 rapid requests in succession
- Good for: Peak load testing, latency spikes
- Requests: 50
- Duration: ~15 seconds

Steady Load (Option 3)

- Consistent 30 requests/minute for 60 seconds
- Good for: Average load patterns, trends
- Requests: 30
- Duration: 60 seconds

Concurrent Load (Option 4)

- 5 threads making 20 requests each simultaneously
- Good for: Thread safety, concurrent users
- Requests: 100
- Duration: ~30 seconds

Realistic Traffic (Option 5)

- Simulates real usage: quiet periods, normal traffic, busy hours, bursts
- Good for: Real-world patterns, all metrics
- Requests: ~100-150
- Duration: 5 minutes

Extended Test (Option 6) ⭐ BEST FOR DASHBOARD

- Runs all test types in sequence
- Good for: Complete dashboard population
- Requests: ~200+
- Duration: 10 minutes

🎯 Recommended Workflow

For Screenshots & Assignment:

1. **Run Extended Test** (10 minutes):

```
# In the running terminal, type: 6
# Then press ENTER
```

2. **Wait 2-3 minutes** for data to accumulate
3. **Open Grafana** (http://localhost:3000)
    - Login: admin/admin
    - Navigate to dashboard
4. **Configure Dashboard**:
    - Time range: "Last 15 minutes"
    - Auto-refresh: "10s"
5. **Take Screenshots**:
    - ✅ Full dashboard view
    - ✅ Individual panels (zoom in)
    - ✅ Time series graphs showing trends
    - ✅ Pie chart with distribution
    - ✅ Health status indicators
    - ✅ System metrics
6. **Alternative Views**:
    - Prometheus UI: http://localhost:9090
    - MLflow UI: http://localhost:5000
    - API Docs: http://localhost:8000/docs

📁 Files Created

```
heart-disease-mlops/
├── comprehensive_api_tester.py          ← Interactive tester (RUNNING
NOW)
├── run_grafana_tests.py                 ← Automated test runner
├── GRAFANA_GUIDE.md                     ← Complete usage guide
├── deployment/
│   ├── prometheus/
│   │   └── prometheus.yml               ← Metrics collection config
│   └── grafana/
│       ├── dashboards/
│       │   └── heart_disease_dashboard.json  ← Dashboard definition
│       └── provisioning/
│           ├── datasources/
│           │   └── prometheus.yml       ← Data source config
│           └── dashboards/
│               └── dashboard.yml        ← Auto-load config
```

🐛 Troubleshooting

Dashboard shows "No data"

1. Wait 10-20 seconds for Prometheus to scrape
2. Check time range (top right) - set to "Last 15 minutes"
3. Run more tests to generate data
4. Verify Prometheus is collecting: http://localhost:9090/targets

Metrics not updating

1. Check API is running: `curl http://localhost:8000/health`
2. Check Prometheus targets: http://localhost:9090/targets
3. Restart Prometheus if needed:

```
pkill prometheus
cd heart-disease-mlops
prometheus --config.file=deployment/prometheus/prometheus.yml \
  --storage.tsdb.path=./prometheus-data --web.enable-lifecycle &
```

Can't access Grafana

1. Check if running: `curl http://localhost:3000/api/health`
2. Restart if needed: `brew services restart grafana`
3. Check logs: `brew services list`

✅ Success Checklist

- [x] FastAPI running on port 8000
- [x] Prometheus running on port 9090 and scraping metrics
- [x] Grafana running on port 3000
- [x] Dashboard created and configured
- [x] Testing tools created
- [x] Interactive tester running
- [ ] **TODO: Run Extended Test (option 6) for 10 minutes**
- [ ] **TODO: Open Grafana and view dashboard**
- [ ] **TODO: Take screenshots for assignment**

🎉 You're All Set!

RIGHT NOW:

1. The comprehensive tester is waiting for your input
2. **Type `6` and press ENTER** to run the Extended Test
3. Wait 10 minutes while it generates rich data
4. Open Grafana (http://localhost:3000) and see your dashboard come alive!

Your dashboard will show:

- Real-time prediction metrics
- Performance trends
- Risk distribution
- System health
- And much more!

Perfect for your MLOps assignment demonstration! 🚀