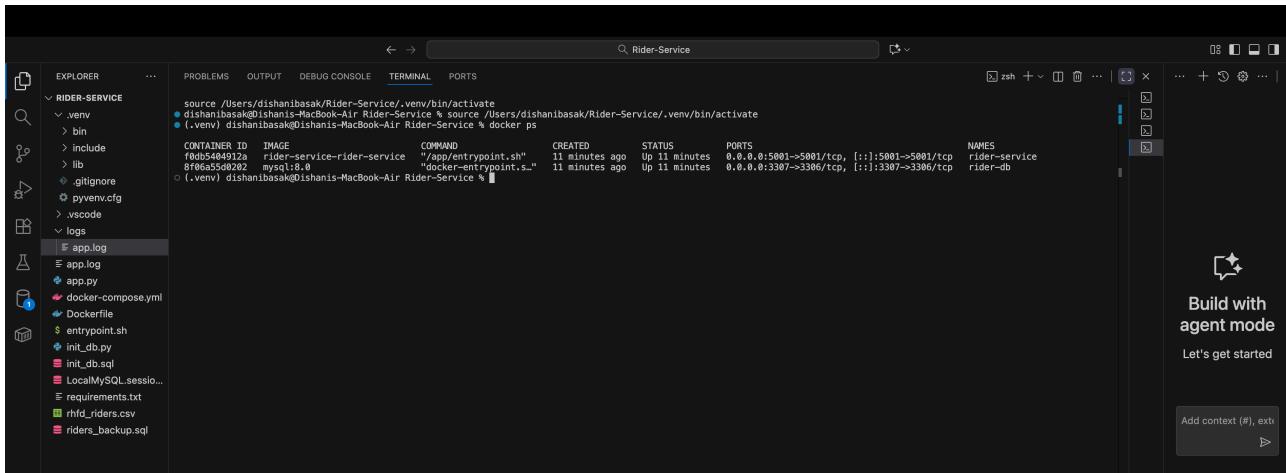
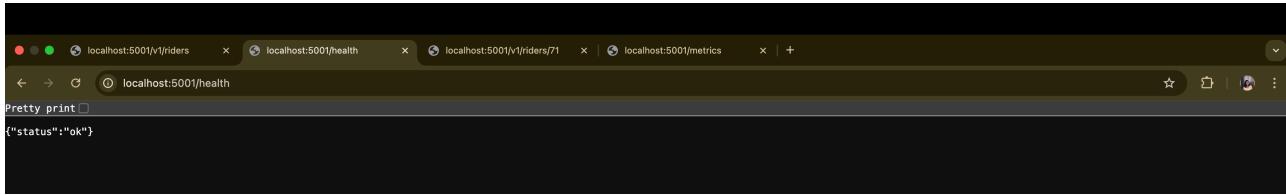


## docker ps

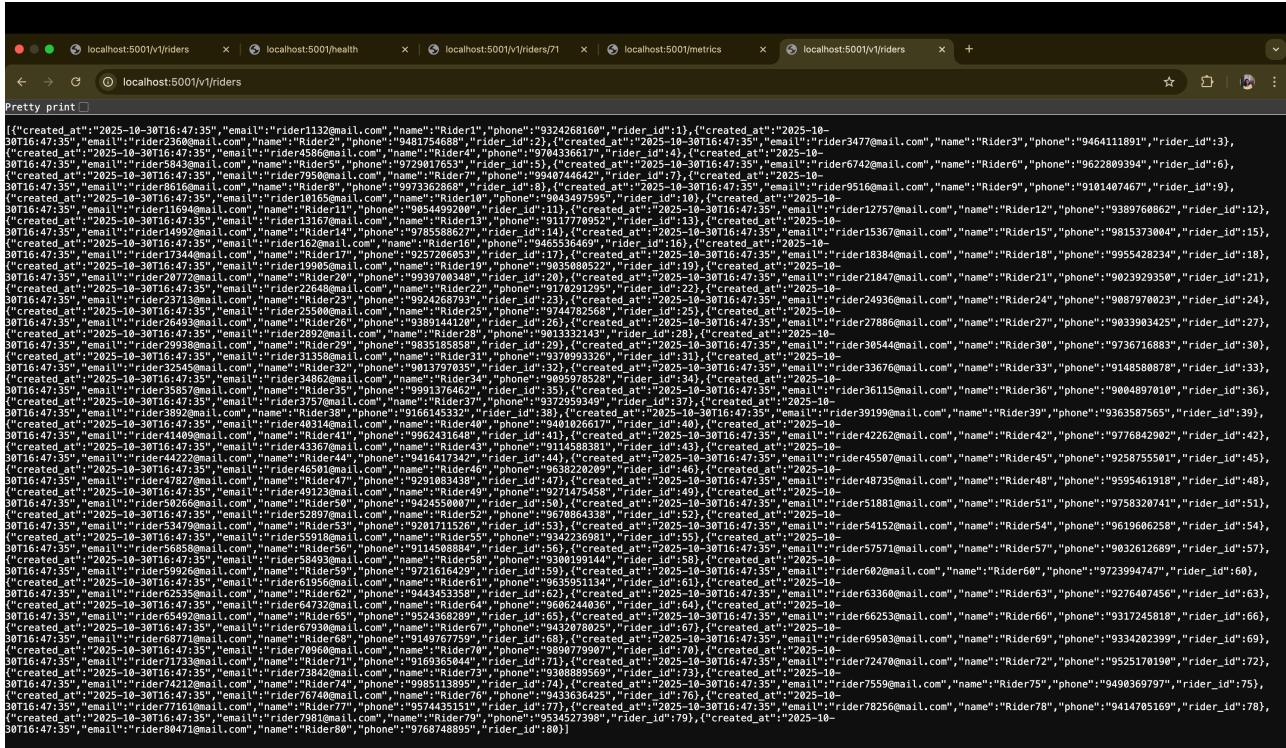


## service health endpoints

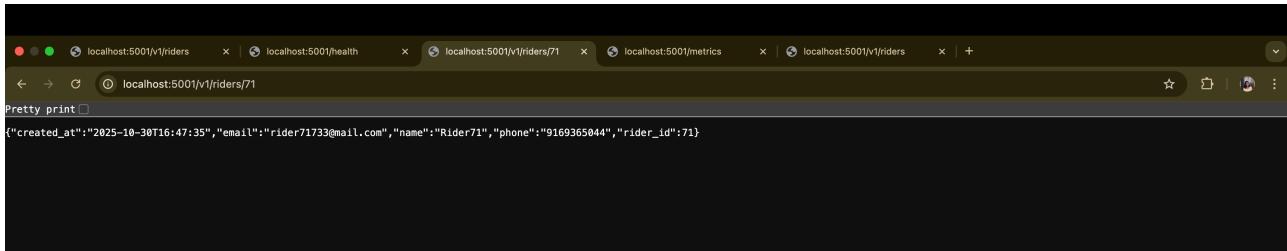


## sample API calls.

<http://localhost:5001/v1/riders>

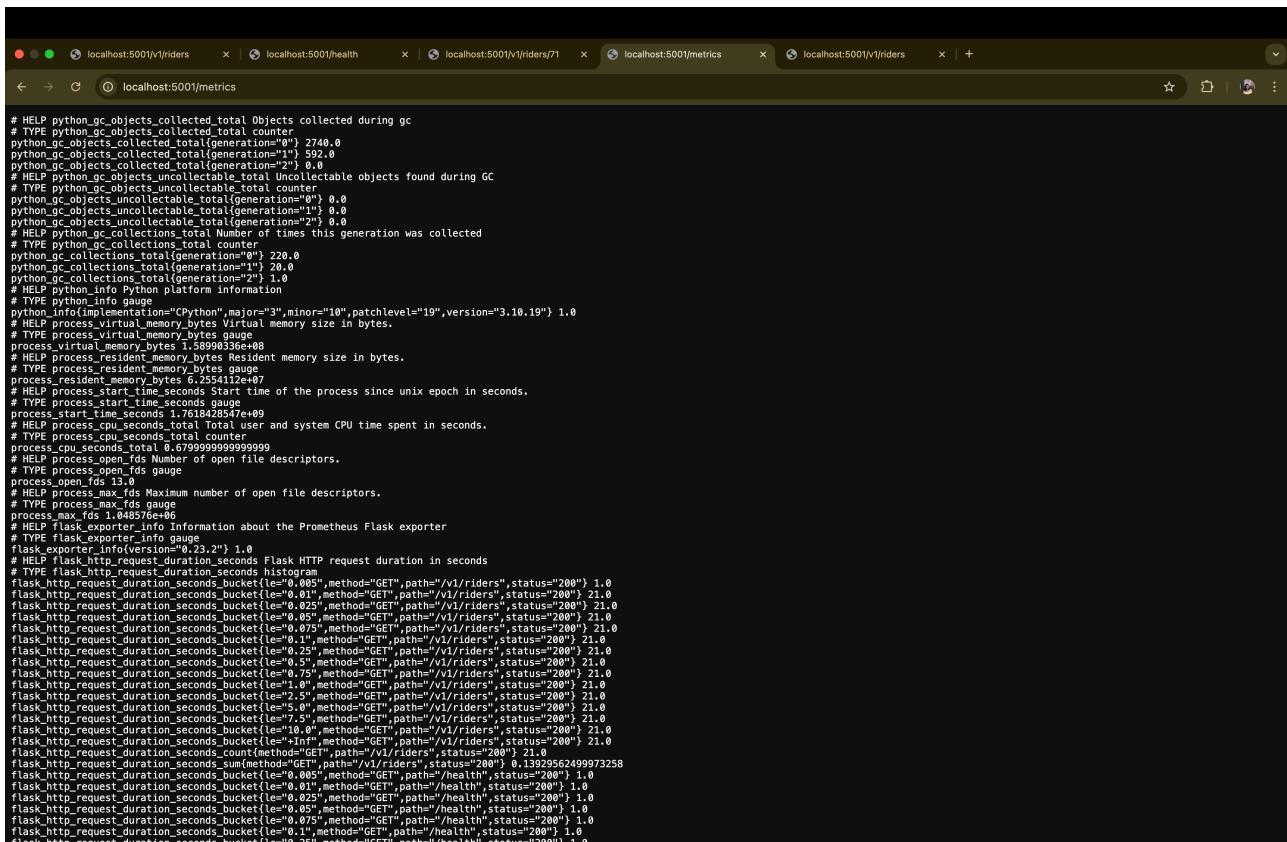


<http://localhost:5001/v1/riders/71>

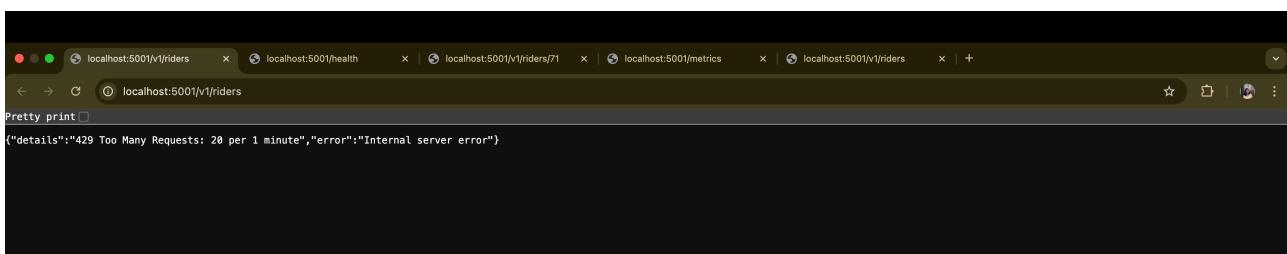


## Monitoring and logging section

## Prometheus metrics



## Rate Limiting



## Logs in MySQL / app.log

### DB logs

The screenshot shows the MySQL Workbench interface. On the left, the sidebar includes sections for Administration, INSTANCE, PERFORMANCE, and Object Info. The main area displays the results of a query:

```

USE riders_db;
SELECT * FROM logs_riders ORDER BY created_at DESC LIMIT 10;

```

The Result Grid shows the following data:

ID	Level	Message	Created At
49	INFO	Incoming GET request: /favicon.ico	2025-10-30 17:02:40
48	INFO	Incoming GET request: /v1/riders	2025-10-30 17:02:40
47	INFO	Incoming GET request: /favicon.ico	2025-10-30 16:57:23
46	INFO	Incoming GET request: /metrics	2025-10-30 16:57:23
45	INFO	Incoming GET request: /v1/riders	2025-10-30 16:56:34
44	INFO	Incoming GET request: /v1/riders	2025-10-30 16:56:34
43	INFO	Incoming GET request: /favicon.ico	2025-10-30 16:56:33
42	INFO	Incoming GET request: /v1/riders	2025-10-30 16:56:33
41	INFO	Incoming GET request: /favicon.ico	2025-10-30 16:56:32
40	INFO	Incoming GET request: /v1/riders	2025-10-30 16:56:32

Below the grid, the Action Output pane shows the history of queries:

Time	Action	Response	Duration / Fetch Time
4 22:26:00	use rider_db;	0 row(s) returned	0.0012 sec / 0.00000...
5 22:26:41	SELECT * FROM logs_riders ORDER BY created_at DESC LIMIT 10	6 row(s) returned	0.0014 sec
6 22:26:41	USE riders_db	0 row(s) affected	0.0013 sec / 0.00000...
7 22:39:23	SELECT * FROM logs_riders ORDER BY created_at DESC LIMIT 10	10 row(s) returned	0.0032 sec
8 22:39:23	USE rider_db;	0 row(s) affected	0.0022 sec / 0.00000...

At the bottom, the message "Query Completed" is displayed.

### app.log

The screenshot shows the Rider IDE interface. The Explorer view on the left shows the project structure with files like app.py, init\_db.py, init\_db.sql, requirements.txt, Dockerfile, entrypoint.sh, docker-compose.yml, and logs/app.log. The logs/app.log file is selected.

The terminal at the bottom shows the output of the command "docker-compose up --build".

```

dishanibasak@Dishanis-MacBook-Air:~/Rider-Service % docker-compose up --build
=> exporting manifest sha256:28dc140e6079c789fc095ac53b1cc592abb65590782bc955dc4951071bc
=> exporting config sha256:e939323f0bab3e5bc851340ee704f92f284afcb3c65c54ebfb72ec4dc4b2e35
=> exporting attestations manifest sha256:65966106e10ef0471b30325b21ebfd721c04d41b7756340b13a70494c60da1f
=> exporting metrics manifest sha256:7598e0f095754a646fe0f08bc2938a463e36a71236
=> naming to docker.io/library/rider-service:rider-service:latest
=> unpacking to docker.io/library/rider-service:rider-service:latest
=> resolving provenance for metadata file
  ✓ rider-service-rider-service      Built
  ✓ Network rider-service_rider-network Created
  ✓ Volume rider-service_rider_data     Created
  ✓ Container rider-service           Created
  ✓ Container rider-service           Created
Attaching to rider-db, rider-service

```

The status bar at the bottom indicates the number of rows (1, Col 1), spaces (Spaces: 4), and encoding (UTF-8). There is also a Log button.