Allsky Camera Network for Detecting Bolides

Tyler Turner
Vincent Quintero
Jean-Pierre Derbes
Charles Derbes
Dr. Csaba Palotai

Task Matrix (Milestone 2)

Task	Completion	Tyler	Vincent	Jean-Pierre	Charles
Continuously fix endless stream of issues	90%	20%	20%	40%	20%
Add logs for easier diagnosis of issues	100%	0%	90%	0%	10%
Replace current C++ camera code	70%	30%	50%	20%	0%
Implement Server API	90%	70%	5%	25%	0%
Implement Client API	80%	10%	0%	50%	40%
Begin writing CLI	0%	0%	0%	0%	0%
IoT Style Setup	90%	20%	0%	30%	50%

Task Discussion

Continuously fix endless stream of issues -> Notification system bug and time zone bug fixed

Add logs for easier diagnosis of issues - > Logs are chunked and queryable

Replace current C++ camera code -> Switched to augmentation and background subtraction

Implement server API -> Wrote and tested routes for server API

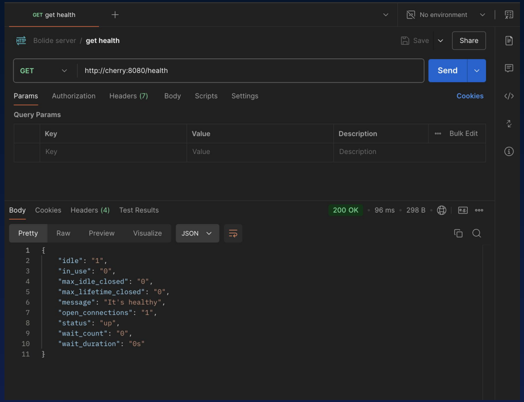
Implement client API -> Wrote and tested routes for client API

Begin writing CLI -> On hold until client is implemented and we have more events

IoT style setup -> Components are built but they still need to be glued together

Client + Server Demo

Server API

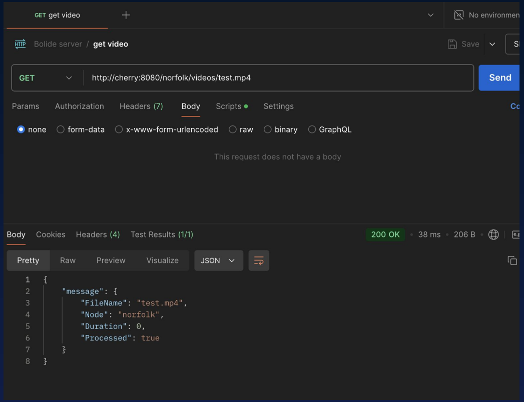


GET http://cherry:8080/health

Key	Value	Description
Key	Value	Description

200 OK · 96 ms · 298 B

```
{
    "idle": "1",
    "in_use": "0",
    "max_idle_closed": "0",
    "max_lifetime_closed": "0",
    "message": "It's healthy",
    "open_connections": "1",
    "status": "up",
    "wait_count": "0",
    "wait_duration": "0s"
}
```



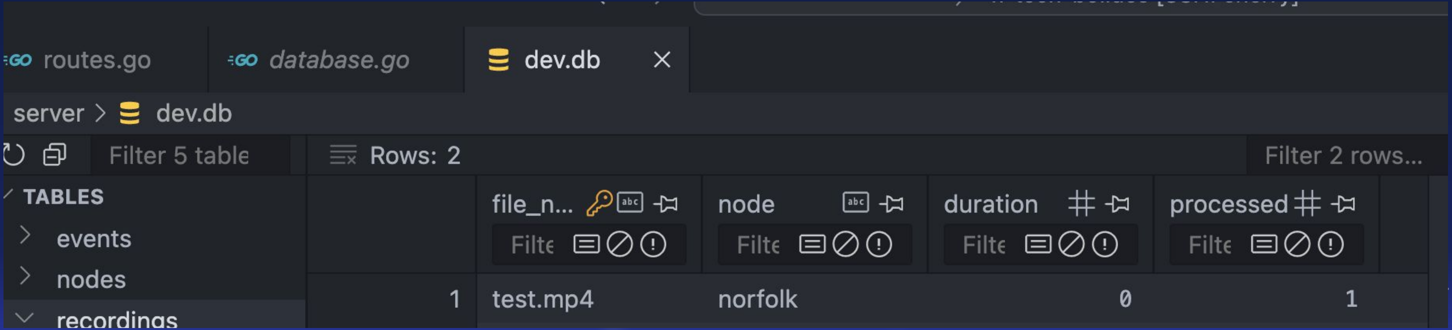
GET http://cherry:8080/norfolk/videos/test.mp4

[✓ none] form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

200 OK · 38 ms · 206 B

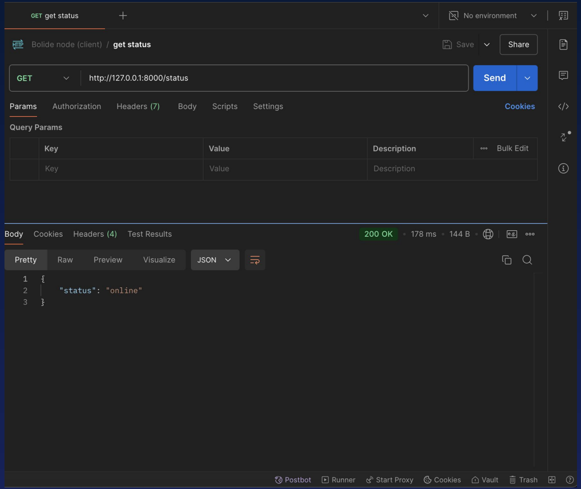
```
{
    "message": {
        "FileName": "test.mp4",
        "Node": "norfolk",
        "Duration": 0,
        "Processed": true
    }
}
```



Rows: 2

	file_n...	node	duration	processed
1	test.mp4	norfolk	0	1

Client API

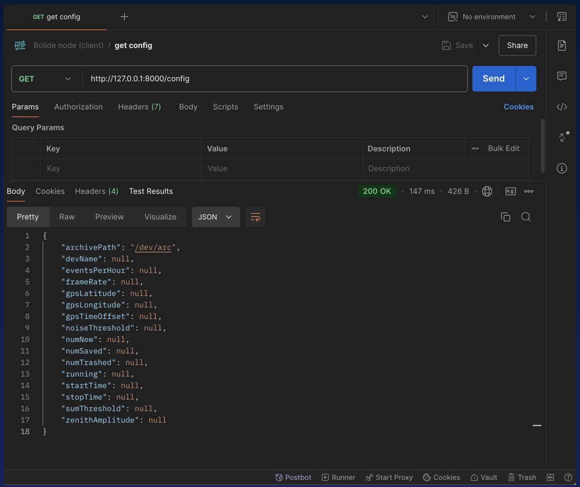


GET http://127.0.0.1:8000/status

Key	Value	Description
Key	Value	Description

200 OK • 178 ms • 144 B

```
{
    "status": "online"
}
```

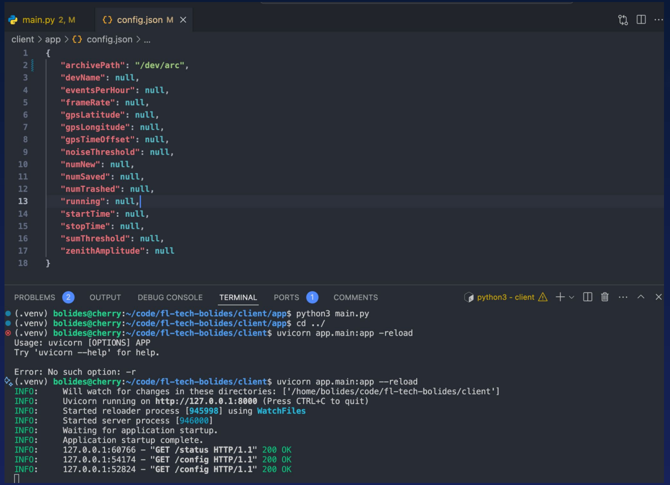


GET http://127.0.0.1:8000/config

Key	Value	Description
Key	Value	Description

200 OK • 147 ms • 426 B

```
{
    "archivePath": "/dev/arc",
    "devName": null,
    "eventsPerHour": null,
    "frameRate": null,
    "gpsLatitude": null,
    "gpsLongitude": null,
    "gpsTimeOffset": null,
    "noiseThreshold": null,
    "numNew": null,
    "numSaved": null,
    "numTrashed": null,
    "running": null,
    "startTime": null,
    "stopTime": null,
    "sumThreshold": null,
    "zenithAmplitude": null
}
```

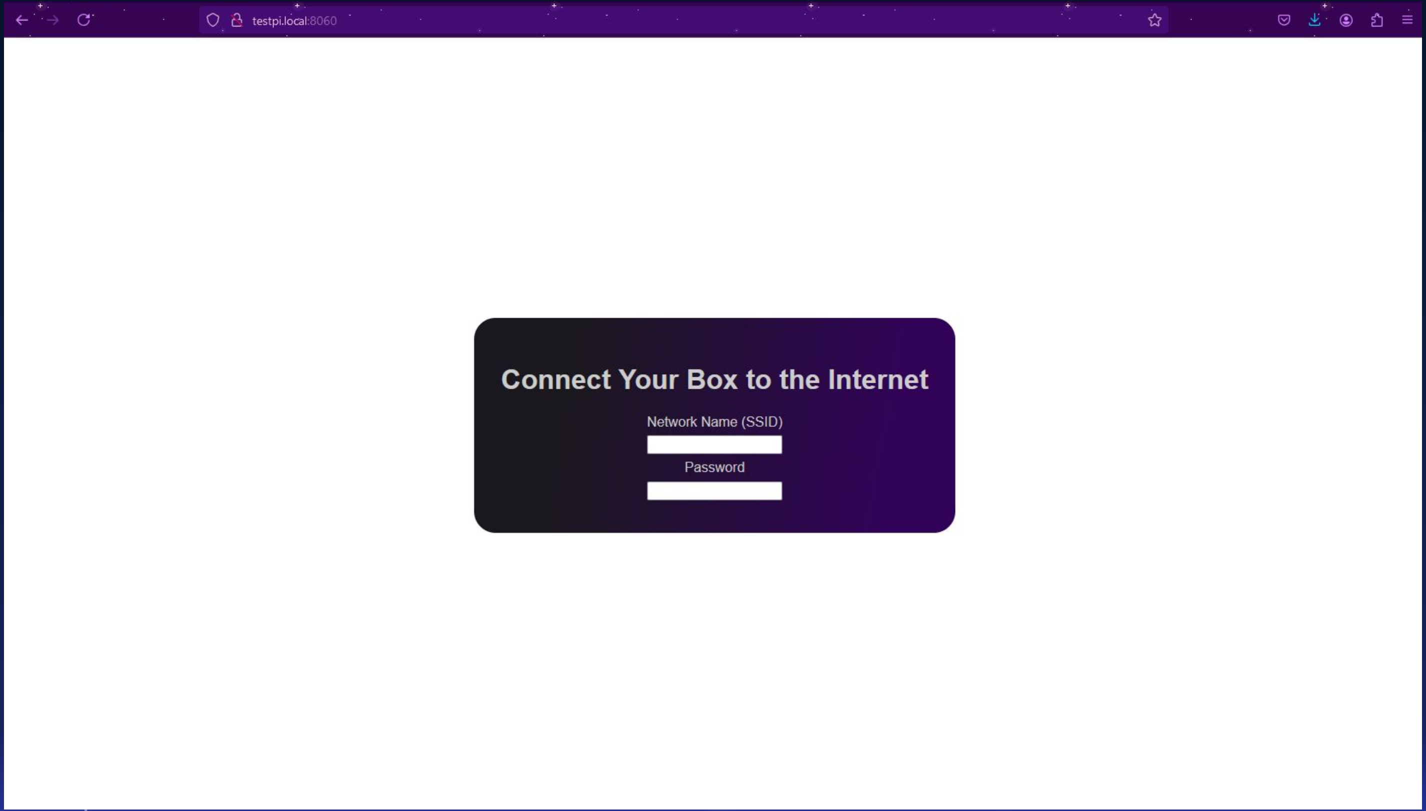


```
{
    "archivePath": "/dev/arc",
    "devName": null,
    "eventsPerHour": null,
    "frameRate": null,
    "gpsLatitude": null,
    "gpsLongitude": null,
    "gpsTimeOffset": null,
    "noiseThreshold": null,
    "numNew": null,
    "numSaved": null,
    "numTrashed": null,
    "running": null,
    "startTime": null,
    "stopTime": null,
    "sumThreshold": null,
    "zenithAmplitude": null
}
```

```
(.venv) bolides@cherry:~/code/fl-tech-bolides/client/app$ python3 main.py
(.venv) bolides@cherry:~/code/fl-tech-bolides/client/app$ cd ../
(.venv) bolides@cherry:~/code/fl-tech-bolides/client$ uvicorn app.main:app -reload
Usage: uvicorn [OPTIONS] APP
Try 'uvicorn --help' for help.

Error: No such option: -r
(.venv) bolides@cherry:~/code/fl-tech-bolides/client$ uvicorn app.main:app --reload
INFO:     Will watch for changes in these directories: ['/home/bolides/code/fl-tech-bolides/client']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [945998] using WatchFiles
INFO:     Started server process [946000]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:60766 - "GET /status HTTP/1.1" 200 OK
INFO:     127.0.0.1:54174 - "GET /config HTTP/1.1" 200 OK
INFO:     127.0.0.1:52824 - "GET /config HTTP/1.1" 200 OK
```

IoT



testpi.local:8060

Connect Your Box to the Internet

Network Name (SSID)

Password

Contribution of Each Member

Tyler Turner

- Server API endpoint implementation
- Video processing queue implementation
- Notification service implementation

Vincent Quintero

- Rewrote event detection software
- Improved image processing
- Improved event detection algorithm
- Event logging for recordings and processing

Contribution of Each Member

Jean-Pierre Derbes

- Fixed None/null errors in node state fields
- Implementation of server and client API endpoint handlers
- Database design and implementation

Charles Derbes

- IoT style setup
- Client API endpoint implementation
- Logging for node/client processes

Task Matrix (Milestone 3)

Task	Tyler	Vincent	Jean-Pierre	Charles
Replace current C++ camera code	10%	35%	45%	10%
Implement Server API	50%	0%	50%	0%
Implement Client API	20%	0%	20%	60%
Begin writing CLI	30%	10%	50%	10%
IoT style setup	20%	0%	10%	70%
Classification	0%	33%	33%	34%
Start writing UI	20%	70%	0%	10%
Create setup process for node	75%	0%	25%	0%

Task Discussion

Task 1: Replace current C++ camera code

- Event triggering + recording: C++ -> Python
- Detection algorithm and image processing improved
- Client side recording chunking

Task 2: Implement Server API

- Missing functionality on a few endpoints
- Need to stress test server with expected video sizes

Task 3: Implement Client API

- One endpoint handler needs implementation
- Need to test client api integration with server

Task Discussion

Task 4: Begin writing CLI

- Enough data to start CLI
- CLI exposes internal system functionality
  - Generate composites
  - Classify composites + classification report

Task 5: IoT Style Setup

- FastAPI + HTML + CSS + JS
- Edge cases need to be solved
- Communication with server API

Task 6: Classification

- Binary Classifier
- Need to augment data through transforms and simulations
- Tune CNN

Task Discussion

Task 7: UI

- On hold until until client and server infrastructure is implemented and well tested
- UI mockups will be modified based on feedback from past presentations

Task 8: Node Setup Process

- Workflow to set up and test newly built node
    - Install OS + required packages
- Ansible to manage node updates
- Easier to diagnose node issues

Thanks!