



TRƯỜNG ĐẠI HỌC FPT

MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Vigision - Real-time Intelligent Video Surveillance System

MamoruTech	
Group Members	Phan Hồng Phúc - CE171166 Võ Minh Đạt - CE160116 Nguyễn Lê Quang Thịnh - CE161130 Đinh Thế Anh - CE160305
Supervisor	Trần Ngọc Hoàng
Ext Supervisor	N/A
Capstone Project code	SEP490_15

- Can Tho, August 2024 -

Table of Contents

Acknowledgement	19
Definition and Acronyms	20
I. Project Introduction.....	21
1. Overview	21
1.1 Project Information.....	21
1.2 Project Team	21
1.2.1 Supervisor	21
1.2.2 Team Members.....	21
2. Product Background.....	21
2.1 Flask.....	21
2.2 React	22
2.3 TypeScript	22
2.4 Axios	23
2.5 Tailwind CSS	23
2.6 NGINX.....	24
2.7 ngrok	24
2.8 Vite	25
2.9 SQLite	25
2.10 peewee.....	26
2.11 FFmpeg.....	26
2.12 PyTorch	26
2.13 ZeroMQ	27
2.14 OpenCV	27
2.15 YOLO.....	28
2.16 AlphaPose	28
2.17 NumPy.....	28
2.18 Pandas.....	29
2.19 Docker	29
2.20 Python	30
3. Existing Systems	30
3.1 Related works.....	30
3.1.1 Efficient Convolutional Network Architecture.....	30
3.1.2 Fall Detection	31
3.2 Existing Systems	32
3.2.1 Blue Iris.....	32

3.2.2 Zoneminder.....	33
3.2.3 Frigate	34
3.2.4 Viseron	35
3.2.5 MotionEye.....	36
3.2.6 iSpy	36
3.2.7 Shinobi	38
4. Business Opportunity.....	39
5. Software Product Vision	40
6. Project Scope & Limitations	40
6.1 Major Features.....	40
6.2 Limitations & Exclusions	43
II. Project Management Plan	43
1. Overview	43
1.1 Scope & Estimation	43
1.2 Project Objectives	46
1.3 Project Risks	47
2. Management Approach	48
2.1 Project Process.....	49
2.2 Quality Management	50
2.3 Training Plan.....	50
3. Project Deliverables	51
4. Project Organization	52
4.1 Team & Structures	52
4.2 Roles & Responsibilities	53
5. Project Communications.....	54
6. Configuration Management.....	54
6.1 Document Management.....	54
6.2 Source Code Management.....	55
6.3 Tools & Infrastructures	56
III. Software Requirement Specification	57
1. Overall Description.....	57
1.1 Product Overview.....	57
1.2 Business Rules	57
2. User Requirements	59
2.1 Overview	59
2.1.1 System Actors.....	59

2.1.2 Use Case Diagrams.....	60
2.1.3 Use Case List.....	61
2.2. Authentication	63
2.2.1 Login.....	63
2.2.2 Logout	64
2.2.3 Forgot password	66
2.3. User management.....	68
2.3.1 Manage users.....	68
2.4. Cameras & Surveillance	73
2.4.1 View live cameras	73
2.4.2 Manage cameras.....	75
2.4.3 Manage camera groups	80
2.4.4 Edit camera group layout.....	84
2.4.5 View in full screen	86
2.4.6 View detail of live camera.....	88
2.4.7 Enable/disable detection	90
2.4.8 Enable/disable recording	92
2.4.9 Enable/disable snapshots	94
2.5. AI Feature Setting and Debugging	96
2.5.1 View defined masks and zones	96
2.5.2 Manage motion masks.....	98
2.5.3 Manage zones	103
2.5.4 Manage object masks	110
2.5.5 Fine-tune motion detection	115
2.5.6 Debug	117
2.6 Events	122
2.6.1 View alerts/detections	122
2.6.2 View motion events	125
2.6.3 View event recording	126
2.6.4 Mark events as reviewed	128
2.6.5 Delete events	130
2.7 Exports	131
2.7.1 View exports	131
2.7.2 View export detail	133
2.7.3 Export event recording	134
2.7.4 Export camera recording.....	136

2.7.5 Download export.....	137
2.7.6 Rename export.....	139
2.7.7 Delete export	140
2.8 Configuration	142
2.8.1 Config	142
2.9 Statistics Dashboard.....	146
2.9.1 View system metrics	146
2.10 General.....	148
2.10.1 Restart application	148
2.10.2 Change application theme	149
2.10.3 Get shareable link	150
3. Functional Requirements.....	152
3.1 System Functional Overview.....	152
3.1.1 Screens Flow	152
3.1.2 Screen Descriptions.....	153
3.1.3 Screen Authorization	158
3.1.4 Non-Screen Functions	159
3.1.5 Entity Relationship Diagram.....	160
4. Non-Functional Requirements.....	163
4.1 External Interfaces	163
4.2.1 User interfaces	163
4.2.2 Software Interfaces.....	163
4.2.2 Communication Interfaces.....	164
4.2 Quality Attributes.....	164
4.2.1 Usability.....	164
4.2.2 Reliability.....	164
4.2.3 Performance.....	164
4.2.4 Security	164
4.2.5 Supportability.....	164
4.2.6 Design Constraints	164
IV. Software Design Description	165
1 System Design	165
1.1 System Architecture.....	165
1.2 Package Diagram.....	166
1.3 Machine Learning Model Architecture	168
1.3.1 Methodology.....	168

1.3.2 Experiments and evaluation	172
2 Background Processing Design	182
2.1 Recording Management Processing	182
2.1.1 Class Diagram	182
2.1.2 Sequence Diagram	183
2.2 Review Segment Management Processing.....	187
2.2.1 Class Diagram	187
2.2.2 Sequence Diagram	188
2.3 Object Detection Processing.....	195
2.3.1 Class Diagram	195
2.3.2 Sequence Diagram	196
2.4 Detected Frame Processing	198
2.4.1 Class Diagram	198
2.4.2 Sequence Diagram	199
2.5 Video Output Processing.....	201
2.5.1 Class Diagram	201
2.5.2 Sequence Diagram	202
2.6 Camera frames processing.....	204
2.6.1 Class Diagram	204
2.6.2 Sequence Diagram	205
2.7 Camera capture processing	208
2.7.1 Class Diagram	208
2.7.2 Sequence Diagram	209
2.8 Timeline Processing	212
2.8.1 Class Diagram	212
2.8.2 Sequence Diagram	213
2.9 Event Processing	216
2.9.1 Class Diagram	216
2.9.2 Sequence Diagram	217
2.10 Event Cleanup Processing	219
2.10.1 Class Diagram	219
2.10.2 Sequence Diagram	220
2.11 Recording Cleanup Processing.....	223
2.11.1 Class Diagram	223
2.11.2 Sequence Diagram	224
2.12 Storage Maintenance Processing.....	227

2.12.1 Class Diagram	227
2.12.2 Sequence Diagram	228
2.13 Stats Emission Processing	232
2.13.1 Class Diagram	232
2.13.2 Sequence Diagram	233
2.14 Start Vigision Watchdog	235
2.14.1 Class Diagram	235
2.14.2 Sequence Diagram	236
3 Detailed Design	237
3.1 Login	237
3.1.1 Class Diagram	237
3.1.2 Sequence Diagram	238
3.2 Logout	239
3.2.1 Class Diagram	239
3.2.2 Sequence Diagram	239
3.3 Forgot password	240
3.3.1 Class Diagram	240
3.3.2 Sequence Diagram	241
3.4 Manage users	243
3.4.1 Class Diagram	243
3.4.2 Sequence Diagram	244
3.5 View live cameras	252
3.5.1 Class Diagram	252
3.5.2 Sequence Diagram	253
3.6 Manage cameras	255
3.6.1 Class Diagram	255
3.6.2 Sequence Diagram	256
3.7 Manage camera groups	263
3.7.1 Class Diagram	263
3.7.2 Sequence Diagram	264
3.8 Edit camera group layout	268
3.8.1 Class Diagram	268
3.8.2 Sequence Diagram	269
3.9 View in full screen	270
3.9.1 Class Diagram	270
3.9.2 Sequence Diagram	270

3.10 View detail of live camera.....	271
3.10.1 Class Diagram.....	271
3.10.2 Sequence Diagram	272
3.11 Get shareable link	274
3.11.1 Class Diagram.....	274
3.11.2 Sequence Diagram	274
3.12 Enable/disable detection	275
3.12.1 Class Diagram.....	275
3.12.2 Sequence Diagram	276
3.13 Enable/disable recording	277
3.13.1 Class Diagram.....	277
3.13.2 Sequence Diagram	278
3.14 Enable/disable snapshots	279
3.14.1 Class Diagram.....	279
3.14.2 Sequence Diagram	280
3.15 View defined masks and zones	281
3.15.1 Class Diagram.....	281
3.15.2 Sequence Diagram	282
3.16 Manage motion masks.....	285
3.16.1 Class Diagram.....	285
3.16.2 Sequence Diagram	286
3.17 Manage zones	293
3.17.1 Class Diagram.....	293
3.17.2 Sequence Diagram	294
3.18 Manage object masks	302
3.18.1 Class Diagram.....	302
3.18.2 Sequence Diagram	303
3.19 Fine-tune motion detection	309
3.19.1 Class Diagram.....	309
3.19.2 Sequence Diagram	310
3.20 Debug	312
3.21 Class Diagram.....	312
3.22 Sequence Diagram	313
3.21 View alerts/detections	315
3.21.1 Class Diagram.....	315
3.21.2 Sequence Diagram	316

3.22 Manage exports	328
3.22.1 Class Diagram.....	328
3.22.2 Sequence Diagram	329
3.23 Config	338
3.23.1 Class Diagram.....	338
3.23.2 Sequence Diagram	339
3.24 View system metrics	340
3.24.1 Class Diagram.....	340
3.24.2 Sequence Diagram	341
3.25 Restart application	343
3.25.1 Class Diagram.....	343
3.25.2 Sequence Diagram	344
3.26 Change application theme	345
3.26.1 Class Diagram.....	345
3.26.2 Sequence Diagram	345
4. Class Specifications	346
4.1 VigisionConfig.....	346
4.1.1 Properties of class	346
4.1.2 Class methods	347
4.2 VigisionApp	347
4.2.1 Properties of class	347
4.2.2 Class methods	348
4.3 CameraConfig.....	352
4.3.1 Properties of class	352
4.3.2 Class methods	352
4.4 RecordConfig.....	353
4.4.1 Properties of class	353
4.4.2 Class methods	354
4.5 RecordingMaintainer	354
4.5.1 Properties of class	354
4.5.2 Class methods	355
4.6 Event	356
4.6.1 Properties of class	356
4.6.2 Class methods	357
4.7 Recordings.....	357
4.7.1 Properties of class	357

4.7.2 Class methods	358
4.8 RecordRetainConfig	358
4.8.1 Properties of class	358
4.8.2 Class methods	358
4.9 EventsConfig	358
4.9.1 Properties of class	358
4.9.2 Class methods	359
4.10 RetainConfig.....	359
4.10.1 Properties of class	359
4.10.2 Class methods	359
4.11 RetainModeEnum	359
4.11.1 Properties of class	359
4.11.2 Class methods	360
4.12 ConfigSubscriber	360
4.12.1 Properties of class	360
4.12.2 Class methods	360
4.13 DetectionSubscriber.....	361
4.13.1 Properties of class	361
4.13.2 Class methods	361
4.14 InterProcessRequestor.....	361
4.14.1 Properties of class	361
4.14.2 Class methods	362
4.15 SegmentInfo.....	362
4.15.1 Properties of class	362
4.15.2 Class methods	362
4.16 DatabaseConfig	363
4.16.1 Properties of class	363
4.16.2 Class methods	363
4.17 VigisionBaseModel.....	363
4.17.1 Properties of class	363
4.17.2 Class methods	363
4.18 ReviewSegmentMaintainer.....	363
4.18.1 Properties of class	363
4.18.2 Class methods	364
4.19 PendingReviewSegment	365
4.19.1 Properties of class	365

4.19.2 Class methods	365
4.20 SharedMemoryFrameManager	366
4.20.1 Properties of class	366
4.20.2 Class methods	366
4.21 DetectionTypeEnum	367
4.21.1 Properties of class	367
4.21.2 Class methods	367
4.22 TrackedObject	367
4.22.1 Properties of class	367
4.22.2 Class methods	368
4.23 SeverityEnum	369
4.23.1 Properties of class	369
4.23.2 Class methods	369
4.24 ReviewConfig	369
4.24.1 Properties of class	369
4.24.2 Class methods	369
4.25 AlertsConfig	369
4.25.1 Properties of class	369
4.25.2 Class methods	370
4.26 DetectionsConfig	370
4.26.1 Properties of class	370
4.26.2 Class methods	370
4.27 ManualEventState	370
4.27.1 Properties of class	370
4.27.2 Class methods	371
4.28 ReviewSegment	371
4.28.1 Properties of class	371
4.28.2 Class methods	372
4.29 ModelConfig	372
4.29.1 Properties of class	372
4.29.2 Class methods	372
4.30 BaseDetectorConfig	373
4.30.1 Properties of class	373
4.30.2 Class methods	373
4.31 ObjectDetectProcess	373
4.31.1 Properties of class	373

4.31.2 Class methods	374
4.32 LocalObjectDetector	375
4.32.1 Properties of class	375
4.32.2 Class methods	375
4.33 TrackedObjectProcessor	375
4.33.1 Properties of class	375
4.33.2 Class methods	377
4.34 Dispatcher	378
4.34.1 Properties of class	378
4.34.2 Class methods	378
4.35 CameraState.....	379
4.35.1 Properties of class	379
4.35.2 Class methods	381
4.36 DetectionPublisher.....	381
4.36.1 Properties of class	381
4.36.2 Class methods	382
4.37 EventUpdatePublisher	382
4.37.1 Properties of class	382
4.37.2 Class methods	382
4.38 EventEndSubscriber	383
4.38.1 Properties of class	383
4.38.2 Class methods	383
4.39 EventTypeEnum	383
4.39.1 Properties of class	383
4.39.2 Class methods	383
4.40 EventStateEnum.....	384
4.40.1 Properties of class	384
4.40.2 Class methods	384
4.41 SnapshotsConfig.....	384
4.41.1 Properties of class	384
4.41.2 Class methods	385
4.42 MotionConfig	385
4.42.1 Properties of class	385
4.42.2 Class methods	386
4.43 ObjectConfig	386
4.43.1 Properties of class	386

4.43.2 Class methods	386
4.44 PreviewRecorder.....	387
4.44.1 Properties of class	387
4.44.2 Class methods	387
4.45 RecordQualityEnum	388
4.45.1 Properties of class	388
4.45.2 Class methods	388
4.46 PreviewFFMpegConverter	388
4.46.1 Properties of class	388
4.46.2 Class methods	389
4.47 RecordPreviewConfig.....	389
4.47.1 Properties of class	389
4.47.2 Class methods	389
4.48 JsmpegCamera	390
4.48.1 Properties of class	390
4.48.2 Class methods	390
4.49 FallDetectConfig.....	391
4.49.1 Properties of class	391
4.49.2 Class methods	391
4.50 CameraMetricsTypes	391
4.50.1 Properties of class	391
4.50.2 Class methods	392
4.51 FilterConfig.....	392
4.51.1 Properties of class	392
4.51.2 Class methods	393
4.52 ImprovedMotionDetector.....	393
4.52.1 Properties of class	393
4.52.2 Class methods	394
4.53 DetectConfig	394
4.53.1 Properties of class	394
4.53.2 Class methods	395
4.54 RemoteObjectDetector.....	395
4.54.1 Properties of class	395
4.54.2 Class methods	396
4.55 NorfairTracker	396
4.55.1 Properties of class	396

4.55.2 Class methods	397
4.56 SPPEFastPose	398
4.56.1 Properties of class	398
4.56.2 Class methods	398
4.57 TSSTG	399
4.57.1 Properties of class	399
4.57.2 Class methods	399
4.58 Graph.....	399
4.58.1 Properties of class	399
4.58.2 Class methods	400
4.59 DenseSTGCN	400
4.59.1 Properties of class	400
4.59.2 Class methods	400
4.60 StreamSpatialTemporalGraph	401
4.60.1 Properties of class	401
4.60.2 Class methods	401
4.61 DenseSTGCNBlock.....	402
4.61.1 Properties of class	402
4.61.2 Class methods	402
4.62 STGCNLayer.....	402
4.62.1 Properties of class	402
4.62.2 Class methods	403
4.63 GraphConvolution.....	403
4.63.1 Properties of class	403
4.63.2 Class methods	403
4.64 SEResnet.....	403
4.64.1 Properties of class	403
4.64.2 Class methods	404
4.65 Bottleneck	405
4.65.1 Properties of class	405
4.65.2 Class methods	405
4.66 FastPose	406
4.66.1 Properties of class	406
4.66.2 Class methods	406
4.67 InferenceNetFastRes50.....	406
4.67.1 Properties of class	406

4.67.2 Class methods	407
4.68 EventsPerSecond.....	407
4.68.1 Properties of class	407
4.68.2 Class methods	407
4.69 SELayer.....	408
4.69.1 Properties of class	408
4.69.2 Class methods	408
4.70 CameraWatchdog	408
4.70.1 Properties of class	408
4.70.2 Class methods	409
4.71 CameraCapture	410
4.71.1 Properties of class	410
4.71.2 Class methods	411
4.72 CameraFfmpegConfig	411
4.72.1 Properties of class	411
4.72.2 Class methods	411
4.73 TimelineProcessor.....	411
4.73.1 Properties of class	411
4.73.2 Class methods	412
4.74 Timeline.....	412
4.74.1 Properties of class	412
4.74.2 Class methods	413
4.75 EventProcessor.....	413
4.75.1 Properties of class	413
4.75.2 Class methods	413
4.76 EventEndPublisher	414
4.76.1 Properties of class	414
4.76.2 Class methods	414
4.77 EventUpdateSubscriber	415
4.77.1 Properties of class	415
4.77.2 Class methods	415
4.78 EventCleanupType	416
4.78.1 Properties of class	416
4.78.2 Class methods	416
4.79 EventCleanup	416
4.79.1 Properties of class	416

4.79.2 Class methods	417
4.80 RecordingCleanup	418
4.80.1 Properties of class	418
4.80.2 Class methods	418
4.81 Previews	418
4.81.1 Properties of class	418
4.81.2 Class methods	419
4.82 StorageMaintainer	419
4.82.1 Properties of class	419
4.82.2 Class methods	420
4.83 StatsEmitter	421
4.83.1 Properties of class	421
4.83.2 Class methods	422
4.84 StatsTrackingTypes	423
4.84.1 Properties of class	423
4.84.2 Class methods	424
4.85 User	424
4.85.1 Properties of class	424
4.85.2 Class methods	424
4.86 CameraGroupConfig	424
4.86.1 Properties of class	424
4.86.2 Class methods	425
4.87 Communicator	425
4.87.1 Properties of class	425
4.87.2 Class methods	425
4.88 StationaryMaxFramesConfig	426
4.88.1 Properties of class	426
4.88.2 Class methods	426
4.89 StationaryConfig	426
4.89.1 Properties of class	426
4.89.2 Class methods	427
4.90 RuntimeMotionConfig	427
4.90.1 Properties of class	427
4.90.2 Class methods	427
4.91 WebSocketClient	428
4.91.1 Properties of class	428

4.91.2 Class methods	429
4.92 WebSocket	430
4.92.1 Properties of class	430
4.92.2 Class methods	430
4.93 ConfigPublisher	430
4.93.1 Properties of class	430
4.93.2 Class methods	431
4.94 RecordExportConfig	432
4.94.1 Properties of class	432
4.94.2 Class methods	432
4.95 RuntimeFilterConfig	432
4.95.1 Properties of class	432
4.95.2 Class methods	433
4.96 FfmpegConfig	433
4.96.1 Properties of class	433
4.96.2 Class methods	433
4.97 Export	434
4.97.1 Properties of class	434
4.97.2 Class methods	435
4.98 RecordingExporter	435
4.98.1 Properties of class	435
4.98.2 Class methods	436
4.99 PlaybackFactorEnum	436
4.99.1 Properties of class	436
4.99.2 Class methods	437
5. Database Design	437
5.1 Local database	437
V. Software Testing Documentation	439
1. Scope of Testing	439
2. Test Strategy	440
2.1 Testing Levels	440
2.2 Test Types	441
2.3 Supporting Tools	442
3. Test Plan	442
3.1 Human Resources	442
3.2 Test Environment	442

3.3 Test Milestones	442
4. Test Cases.....	442
5. Test Reports	443
VI. Release Package & User Guides.....	443
1. Deliverable Package	443
1.1 Source codes & documents	443
1.2 Known Issues, Limitations & Restrictions	444
2. Installation Guides	444
2.1 System Requirements	444
2.2 Installation Instruction	445
2.2.1 Install and setup Docker	445
2.2.2 Install and setup WSL 2 (Windows Subsystem for Linux 2)	448
2.2.3 Install CUDA Toolkit on Windows and WSL	452
3. User Manual.....	456
3.1 Terms and Definitions	456
3.2 System Requirements	457
3.3 Application Usage	457
3.3.1 Overview	457
3.3.2 User guide for admin	459
3.3.3 User guide for member.....	639
VII. Appendix.....	640
1. Research Paper	640
1.1 Information	640
1.2 Abstract.....	641
2. References	642

Acknowledgement

We extend our heartfelt thanks to everyone who contributed to the successful completion of our capstone project, "Vigision - Real-time Intelligent Video Surveillance System." Our deepest gratitude goes to our supervisor, Trần Ngọc Hoàng, whose expert guidance and unwavering support were invaluable throughout the development process.

We are also immensely grateful to our project team members at MamoruTech, who worked diligently to bring this vision to life. Our project was greatly enhanced by the robust technologies we employed, including Flask, React, and PyTorch, among others. We appreciate the developers and communities behind these technologies for making our implementation smoother and more efficient.

Additionally, we would like to thank our families and friends for their encouragement and support, which have been a constant source of motivation throughout this journey.

Lastly, we are open to feedback and suggestions from all our peers and instructors to further improve our system and make it a benchmark in AI-integrated surveillance solutions.

Definition and Acronyms

Table 1. Definition and Acronyms

Acronym	Definition
BA	Business Analysis
BR	Business Rule
ERD	Entity Relationship Diagram
GUI	Graphical User Interface
PM	Project Manager
SDD	Software Design Description
SPMP	Software Project Management Plan
SRS	Software Requirement Specification
UAT	User Acceptance Test
UC	Use Case
API	Application Program Interface
AI	Artificial Intelligence
UI	User Interface

I. Project Introduction

1. Overview

1.1 Project Information

- Project name: Vigision - Real-time Intelligent Video Surveillance System
- Project code: SEP490_15
- Group name: MamoruTech
- Software type: AI Vision System

1.2 Project Team

1.2.1 Supervisor

Table 2. Supervisor

Full Name	Role	Email	Mobile
Trần Ngọc Hoàng	Lecturer	hoangtn20@fe.edu.vn	

1.2.2 Team Members

Table 3. Team members

Full Name	Role	Email	Mobile
Phan Hồng Phúc	Leader	phucphce171166@fpt.edu.vn	0354914539
Võ Minh Đạt	Member	datvmce160116@fpt.edu.vn	0899007584
Nguyễn Lê Quang Thịnh	Member	thinhnlqce161130@fpt.edu.vn	0916257956
Đinh Thế Anh	Member	anhdtce160305@fpt.edu.vn	0986846696

2. Product Background

Vigision is an innovative Network Video Recorder (NVR) and AI Computer Vision software that transforms traditional home camera system into proactive guardian. Developed in response to a growing need for enhanced home surveillance, it utilizes Deep Learning for features like object detection, pose estimation and object detection. Vigision stands out from conventional systems by only alerting users to significant motions, reducing unnecessary notifications. It can be self-hosted on local hardware devices via Docker, allowing users to set up their camera system, configure functions, and receive important alerts from anywhere in the world. This solution offers homeowners reliable, intelligent protection for their homes and loved ones, with the added safety of fall detection for vulnerable family members.

The system is developed using the following technologies: Flask, React, TypeScript, Axios, Tailwind CSS, NGINX, ngrok, Vite, SQLite, peewee, FFmpeg, PyTorch, OpenCV, YOLO, AlphaPose, NumPy, pandas, Docker, Python, ZeroMQ.

2.1 Flask

Flask [1], a micro web framework crafted in Python, earns its classification due to its lean structure, devoid of mandatory tools or libraries. Absent are standard components like database abstraction layers and form validation, typically furnished by third-party libraries. This minimalist design ensures

swift initiation, yet remains scalable for intricate applications. Initially, Flask emerged as a modest enclosure around Werkzeug and Jinja, gradually evolving into one of Python's most sought-after web frameworks. As a Python module, Flask facilitates effortless web application development with its compact and highly adaptable core. Unlike frameworks equipped with Object Relational Managers (ORM), Flask abstains from such features, prioritizing simplicity. Nevertheless, it boasts a myriad of functionalities including URL routing and template engines. Operating as a WSGI web application framework, Flask upholds its reputation as a microframework by preserving a straightforward and expandable core. Rather than furnishing an inbuilt database abstraction layer, Flask extends support for such capabilities through user-friendly extensions. In essence, Flask empowers developers to swiftly construct lightweight web applications with its rich array of Flask Libraries.



Figure 1. Product background - Flask

2.2 React

React [2] is a JavaScript library for building user interfaces, particularly for single-page applications. Its core features include a component-based architecture, virtual DOM for efficient UI updates, and a declarative programming paradigm, enabling developers to describe the desired UI state while React handles DOM manipulation. JSX, a syntax extension, allows for HTML-like code within JavaScript, enhancing readability and expressiveness. With a unidirectional data flow model, data propagates from parent to child components, ensuring a clear and predictable flow within the application. React's ecosystem offers a variety of tools and libraries, such as Redux for state management and Jest for testing, making it a comprehensive solution for creating modern, interactive web applications.



Figure 2. Product background - React

2.3 TypeScript

TypeScript [3], an extension of JavaScript, is a programming language renowned for its strong typing, which significantly enhances tooling capabilities across diverse project scales. It introduces additional syntax to JavaScript, streamlining integration with development environments and enabling early error detection directly within the editor. TypeScript code is transpiled to JavaScript, ensuring seamless compatibility across various platforms where JavaScript is supported, spanning browsers, Node.js, Deno, and other applications. Leveraging its deep understanding of JavaScript, TypeScript utilizes type inference to deliver advanced tooling experiences without the need for excessive manual type annotations.

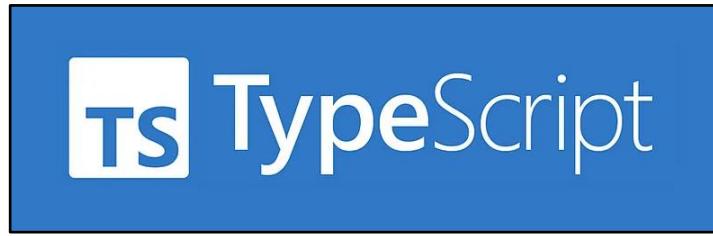


Figure 3. Product background - TypeScript

2.4 Axios

Axios [4] presents a straightforward and user-friendly API tailored for transmitting and retrieving data from web servers, streamlining the development of applications interacting with REST APIs and various web services. Notably, Axios offers robust capabilities in managing request and response interceptors, empowering developers to customize request and response behaviors effortlessly. This flexibility proves invaluable for tasks such as incorporating authentication headers, error handling, or data transformation. Furthermore, Axios boasts features like cancellation, timeouts, and progress tracking, enabling the creation of resilient and responsive applications adaptable to diverse network conditions and user interactions. Beyond its core functionalities, Axios encompasses additional strengths, including seamless compatibility with browser and Node.js environments, support for various data formats like JSON and XML, and seamless integration with prominent front-end frameworks such as React, Angular, and Vue. Leveraged by major enterprises like Amazon, Microsoft, and Google, as well as individual developers and small businesses, Axios continues to witness burgeoning popularity, as its benefits become increasingly evident to developers across diverse projects and industries.



Figure 4. Product background - Axios

2.5 Tailwind CSS

Tailwind CSS [5] is a cutting-edge utility-first CSS framework that revolutionizes the way developers approach styling web applications. Unlike traditional frameworks, Tailwind provides a vast array of utility classes that can be directly applied in HTML markup, allowing for rapid customization without the need to write CSS from scratch. Its modular design fosters a composable approach, enabling developers to effortlessly combine utility classes to craft intricate layouts and designs. With built-in support for responsive design and dark mode, Tailwind empowers developers to create visually stunning and highly adaptable interfaces. Furthermore, its customizable nature allows for fine-tuning of colors, spacing, and other design elements to perfectly match project requirements. By optimizing CSS output for speed and efficiency, Tailwind ensures lightning-fast load times and superior performance. As a result, Tailwind CSS has gained popularity among developers seeking a modern and efficient solution for building stylish and responsive web applications.



Figure 5. Product background - Tailwind CSS

2.6 NGINX

NGINX [6] is a high-performance, open-source web server and reverse proxy server software. Originally developed to address the challenges of handling high traffic loads on websites, NGINX has evolved into a versatile tool used for various purposes, including serving static content, acting as a reverse proxy, load balancer, and more. Key features of NGINX include its efficient handling of concurrent connections, low memory footprint, and ability to scale and handle large volumes of traffic efficiently. It is commonly used as a front-end server for websites and web applications, sitting in front of application servers to handle tasks like SSL termination, caching, and serving static files. NGINX's configuration is typically done through simple text-based configuration files, which offer flexibility and ease of customization. Its robustness, performance, and flexibility have made NGINX a popular choice for powering some of the world's busiest websites and applications.



Figure 6. Product background - NGINX

2.7 ngrok

ngrok [8] is a secure tunneling service that allows developers to expose a local development server to the internet. By creating secure tunnels to localhost, ngrok enables developers to test webhooks, APIs, and other web services running on their local machines as if they were hosted on a public server. This is particularly useful for debugging and testing external integrations. With features such as custom subdomains, HTTP and TCP tunnels, and the ability to inspect traffic, ngrok provides a versatile and powerful tool for modern web development workflows.



Figure 7. Product background - ngrok

2.8 Vite

Vite [7] is a build tool for modern web development projects, designed to provide a fast and efficient development experience. Developed by the creators of Vue.js, Vite is optimized for building Vue.js applications but can also be used with other frameworks like React or Preact. The key feature of Vite is its "zero-config" setup, which means developers can start a new project without needing to configure build tools or bundlers manually. Vite leverages ES module imports to serve code directly to the browser during development, eliminating the need for bundling during the development process. This results in significantly faster build times, as only the modules that are actually needed for the current page are bundled. Additionally, Vite supports hot module replacement (HMR), allowing developers to see changes reflected instantly in the browser without needing to refresh the page. Overall, Vite provides a streamlined and efficient development experience for modern web projects, enabling developers to focus on writing code without being bogged down by complex build configurations.



Figure 8. Product background - Vite

2.9 SQLite

SQLite [9] is a lightweight, serverless, and open-source relational database management system (RDBMS) renowned for its simplicity and portability. As an embedded database, SQLite seamlessly integrates directly into applications, eliminating the need for a separate database server. Its file-based architecture allows the entire database to be stored in a single disk file, making it easy to manage and highly portable across different environments. Despite its compact size, SQLite offers comprehensive SQL support, including transactions, triggers, views, and stored procedures, ensuring data integrity and reliability. Widely utilized in mobile apps, desktop applications, web browsers, and embedded systems, SQLite serves as a versatile solution for scenarios requiring fast, local storage or offline data access.



Figure 9. Product background - SQLite

2.10 peewee

peewee [10] is a small, expressive ORM (Object-Relational Mapping) library for Python that simplifies database interactions by allowing developers to write Python code instead of SQL queries. Designed to be lightweight and easy to use, peewee supports SQLite, MySQL, and PostgreSQL databases. Its features include model definitions, query building, relationships, and transactions, making it a suitable choice for small to medium-sized applications. With its intuitive syntax and comprehensive documentation, peewee enables rapid development and efficient database management.



Figure 10. Product background - peewee

2.11 FFmpeg

FFmpeg [12] is a powerful open-source multimedia framework used for recording, converting, and streaming audio and video. It provides a suite of tools for handling various multimedia formats, including the ability to decode, encode, transcode, mux, demux, stream, filter, and play almost anything that humans and machines have created. FFmpeg is known for its high performance and flexibility, making it a go-to solution for developers and professionals working with digital video and audio. Its command-line interface and extensive codec library support make it ideal for automation and integration into various media processing workflows.



Figure 11. Product background - FFmpeg

2.12 PyTorch

PyTorch [11] stands as an open-source machine learning library renowned for its flexibility and efficiency. What sets PyTorch apart is its dynamic computation graph, offering a more intuitive model development experience compared to static graph frameworks. Its dynamic nature allows developers to define and modify computational graphs during runtime, easing model debugging and experimentation. With automatic differentiation provided by its `autograd` package, PyTorch simplifies gradient computation for optimization algorithms. Moreover, PyTorch boasts a Pythonic API, leveraging familiar syntax and data structures to streamline model development. Its versatility extends to flexible deployment across various platforms, including CPUs, GPUs, and distributed computing environments. Supported by a vibrant ecosystem of libraries like torchvision and torchtext, PyTorch enjoys extensive community support and continuous development, solidifying its position as a leading framework in machine learning research and applications.



Figure 12. Product background - PyTorch

2.13 ZeroMQ

FFmpeg [12] is a powerful open-source multimedia framework used for recording, converting, and streaming audio and video. It provides a suite of tools for handling various multimedia formats, including the ability to decode, encode, transcode, mux, demux, stream, filter, and play almost anything that humans and machines have created. FFmpeg is known for its high performance and flexibility, making it a go-to solution for developers and professionals working with digital video and audio. Its command-line interface and extensive codec library support make it ideal for automation and integration into various media processing workflows.



Figure 13. Product background - ZeroMQ

2.14 OpenCV

OpenCV [14], or Open Source Computer Vision Library, is an indispensable open-source software library renowned for its prowess in computer vision and machine learning applications. Originally introduced by Intel in 1999, OpenCV has emerged as a cornerstone in the field, offering a vast array of tools and algorithms for image processing, object detection and recognition, feature detection, and machine learning integration. Its comprehensive capabilities encompass tasks such as camera calibration, 3D reconstruction, and cross-platform support across multiple programming languages and operating systems. With its versatility and robustness, OpenCV finds widespread utility in diverse domains including robotics, healthcare, automotive, security, and entertainment. Continuously evolving and supported by an active community, OpenCV remains at the forefront of innovation, empowering developers and researchers to push the boundaries of computer vision and image processing technologies.



Figure 14. Product background - OpenCV

2.15 YOLO

YOLO [15], which stands for "You Only Look Once," represents a groundbreaking real-time object detection system that has redefined the landscape of computer vision. Introduced in 2016 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, YOLO boasts unparalleled efficiency by achieving high detection speeds while maintaining remarkable accuracy. Its single-shot detection approach, in contrast to traditional methods, enables rapid object detection within a single pass through the neural network, making it exceptionally fast and resource-efficient. YOLO's unified architecture directly predicts bounding boxes and class probabilities from full images, eliminating the need for multi-step processing. With its versatility and capability to detect a diverse range of objects across various categories, YOLO has found widespread application in fields like autonomous driving, surveillance, and robotics.

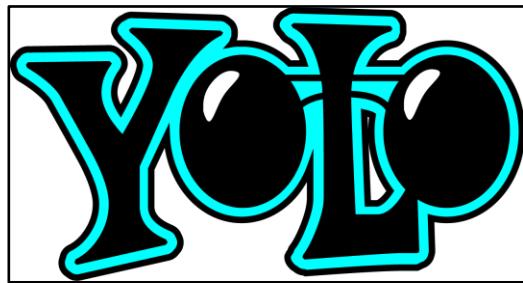


Figure 15. Product background - YOLO

2.16 AlphaPose

AlphaPose [16] is an advanced open-source pose estimation system developed by researchers from the University of Science and Technology of China and SenseTime. It is designed to accurately detect and track human poses in images or video frames, enabling applications such as action recognition, activity analysis, and human-computer interaction. AlphaPose utilizes deep learning techniques, specifically convolutional neural networks (CNNs), to estimate the keypoints of human body joints, including the positions of the head, shoulders, elbows, wrists, hips, knees, and ankles. Key features of AlphaPose include its high accuracy, real-time performance, and support for multiple human subjects in complex scenes. It has become a popular tool in the field of computer vision and has been applied in various domains, including sports analytics, healthcare, and security.



Figure 16. Product background - AlphaPose

2.17 NumPy

NumPy [17], shorthand for Numerical Python, emerges as a cornerstone in Python's scientific computing landscape, offering essential tools for numerical operations and data manipulation. At its

core lies the ndarray, a versatile array object that efficiently handles large multi-dimensional datasets. This array-centric approach enables seamless execution of element-wise operations, facilitating mathematical computations and transformations with unparalleled efficiency. NumPy's broadcasting feature further enhances its capabilities, enabling operations across arrays of varying dimensions without the need for explicit looping. Moreover, NumPy's extensive library of mathematical functions encompasses linear algebra operations, random number generation, and integration with other Python libraries such as SciPy, pandas, and Matplotlib. As a result, NumPy serves as the foundation for a wide array of applications in scientific computing, data analysis, machine learning, and beyond, empowering users to tackle complex numerical challenges with ease and efficiency.



Figure 17. Product background - NumPy

2.18 Pandas

Pandas [18] is a powerful and flexible open-source data analysis and manipulation library for Python. Built on top of NumPy, pandas provides data structures like DataFrame and Series, which are optimized for data manipulation and analysis tasks. It offers a wide range of functionalities, including data cleaning, preparation, aggregation, visualization, and statistical analysis. pandas is widely used in data science, machine learning, and scientific research for its ability to handle large datasets with ease and its integration with other Python libraries, such as Matplotlib, SciPy, and Scikit-learn.



Figure 18. Product background - pandas

2.19 Docker

Docker [19] stands as an open-source containerization platform, facilitating the streamlined creation, deployment, and operation of applications within lightweight and portable containers. These containers encapsulate an application alongside its dependencies, forming a self-contained package capable of seamless execution across diverse machines, irrespective of underlying operating systems or hardware configurations. By furnishing a consistent and reproducible runtime environment, Docker simplifies the development, testing, and deployment of software. Moreover, it empowers developers to architect applications as microservices, allowing for effortless scalability and agility in response to changing demands. Docker's management and deployment are facilitated through a command-line interface or integration with complementary tools like Kubernetes or Docker Compose. Witnessing a surge in popularity, Docker has emerged as a cornerstone for developers, IT professionals, and organizations alike, revolutionizing application development and deployment workflows, curtailing costs, and bolstering the scalability and reliability of software systems.

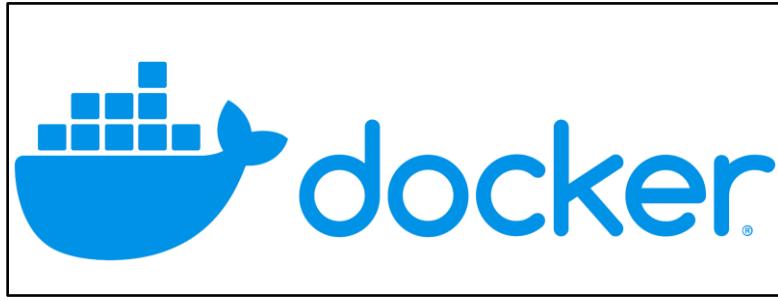


Figure 19. Product background - Docker

2.20 Python

Python [20] is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.



Figure 20. Product background - Python

3. Existing Systems

3.1 Related works

3.1.1 Efficient Convolutional Network Architecture

Efficient convolutional network architectures have been pivotal in advancing skeleton action recognition. One notable technique is the integration of residual connections [21], which enable the network to learn residual mappings. This approach has proven effective in mitigating the vanishing gradient problem and facilitating the training of deeper networks. However, while residual connections enhance the network's ability to capture intricate features, they also introduce additional computational complexity, which can hinder real-time performance in resource-constrained environments. Thus, striking a balance between model complexity and computational efficiency remains a challenge in the design of efficient convolutional network architectures for skeleton action recognition.

Dense aggregation [22] involves the aggregation of features from multiple layers within a network, facilitating the integration of both local and global contextual information. By densely connecting layers, dense aggregation allows for the propagation of information across different levels of abstraction, leading to improved feature representation and discriminative power. However, while dense aggregation enhances the network's ability to capture complex spatial-temporal dependencies,

it also introduces challenges related to computational complexity and memory requirements. The dense connectivity pattern increases the number of parameters and computational operations, potentially leading to scalability issues, especially in deeper networks.

3.1.2 Fall Detection

Optical Flow approaches

Menacho and Ordoñez (2020) present an innovative approach to fall detection using a mobile robot equipped with CNN models processing temporal features derived from optical flow between consecutive images [23]. They emphasize the advantages of vision-based systems in dynamic environments, contrasting with static methods. Traditional systems relying on static cameras or wearables face limitations such as user compliance and environmental obstructions. The integration of CNNs on mobile robots adds complexity, demanding real-time processing on limited hardware. The authors address this challenge by developing an efficient CNN architecture for Nvidia's Jetson TX2 platform, balancing accuracy with operational speed for real-time applications.

Motion History Image (MHI) approaches

Related work in fall detection using Motion History Image (MHI) approach has demonstrated its efficacy in capturing temporal dynamics for identifying falls. Previous studies have utilized MHIs to encode motion history within video sequences, enabling the extraction of motion patterns crucial for fall detection. Results from the paper of Cai et al. (2019) [24] have indicated notable success rates in detecting falls, often achieving accuracies above 90%. However, the MHI approach is limited by its dependence on hand-crafted features, which may not fully capture the complexity of human motion in diverse environments. Additionally, the reliance on predefined thresholds and parameters poses challenges in adapting to varying contexts, potentially leading to reduced robustness and generalization. Despite these limitations, the MHI-based approaches have laid foundational groundwork in the field of fall detection, contributing valuable insights into temporal analysis techniques for activity recognition and safety monitoring.

Long Short-Term Memory (LSTM)

Previous research into Long Short-Term Memory (LSTM) networks for identifying falls has attracted considerable interest owing to their capacity to comprehend time-based connections in sequential data. As an illustration, in a study conducted by Li et al. (2021) [25], LSTM networks were employed for detecting falls in video monitoring setups. The outcomes showcased promising efficacy in precisely pinpointing falls, achieving reported accuracies of up to 98.2%. Nevertheless, a noteworthy drawback of LSTM networks is their vulnerability to the challenges of either diminishing or escalating gradient issues during the training phase, which may impede the acquisition of extended-term dependencies proficiently. This predicament could curtail the model's aptitude to grasp subtle chronological patterns in intricate activities, potentially resulting in diminished efficacy in real-world fall detection scenarios.

Graph Convolutional Networks (GCNs) approaches

Graph Convolutional Networks (GCNs) have become increasingly influential in analyzing data structured as graphs, particularly in the realm of detecting human actions and falls. Among these methods, the Spatial-Temporal Graph Convolutional Network (ST-GCN) has emerged as a leading approach. It effectively utilizes the inherent structure of skeletal data to capture both spatial and temporal dimensions, making it particularly successful in tasks such as action recognition.

Research by H. Zheng and Y. Liu (2022) [26] showcased the effectiveness of ST-GCN in detecting falls using skeletal data, yet its computational demands hindered real-time application. In response, Duan

et al. (2022) [27] introduced a lighter version of ST-GCN, mitigating this challenge without sacrificing accuracy. However, processing efficiency remains an ongoing concern, particularly with extensive datasets or time-sensitive contexts.

Egawa et al. (2023) [28] recently delved into enhancing the performance of ST-GCN by incorporating attention mechanisms. Their study aimed to refine the model's focus on specific skeletal joints, thereby improving the accuracy of fall detection. However, this enhancement came at the cost of increased complexity and processing time, which is a common trade-off in GCN-based research. The research highlights the ongoing challenge of balancing model efficacy with computational efficiency, a central theme in the field of fall detection utilizing graph convolutional networks.

Despite these strides, ST-GCN and similar methods encounter constraints. The intricate computational requirements of ST-GCN pose deployment hurdles for devices with limited processing capabilities, like mobile or edge devices. Moreover, reliance on skeletal data may prove impractical in settings where skeleton extraction is challenging or prone to errors. Additionally, the temporal dimension of ST-GCN demands substantial memory and processing resources, affecting its real-time efficacy.

3.2 Existing Systems

3.2.1 Blue Iris

Blue Iris is a popular video security and surveillance software that allows users to view, record, and manage their security cameras. It supports a wide range of cameras and provides features like motion detection, alerts, and remote access.

Official Website: <https://blueirissoftware.com/>



Figure 21. Existing system - Blue Iris

Key Features:

- Multi-Camera Management: Supports USB or Network IP cameras and analog capture cards, making it versatile in terms of camera compatibility (Blue Iris Software).
- Live Monitoring and Alerts: Users can view live or recorded video via mobile devices and receive alerts through email, SMS, or phone calls. This feature enables prompt responses to potential security issues (GetApp).

- Access Control and Permissions: Allows for controlled access to video feeds, providing varying permission levels for different users.
- AI Integration: Integrates with AI systems like DeepStack for intelligent alert validation.
- Mobile App Support: Offers mobile applications compatible with both iOS and Android devices.
- Reporting and Analytics: Provides detailed statistics and reporting for comprehensive monitoring.
- API Access: Includes an API for custom integration and automation needs.

Advantages:

- Wide compatibility with various camera brands and models.
- User-friendly interface for managing multiple cameras.
- Advanced features like motion detection, alerting, and scheduling.

Disadvantages:

- Requires a Windows-based PC for installation, limiting deployment options.
- Some users report occasional software glitches and stability issues.

3.2.2 Zoneminder

ZoneMinder is an open-source video surveillance software that offers real-time monitoring, recording, and analysis of CCTV systems. It provides features like motion detection, remote access, and support for multiple camera types.

Official Website: <https://zoneminder.com/>

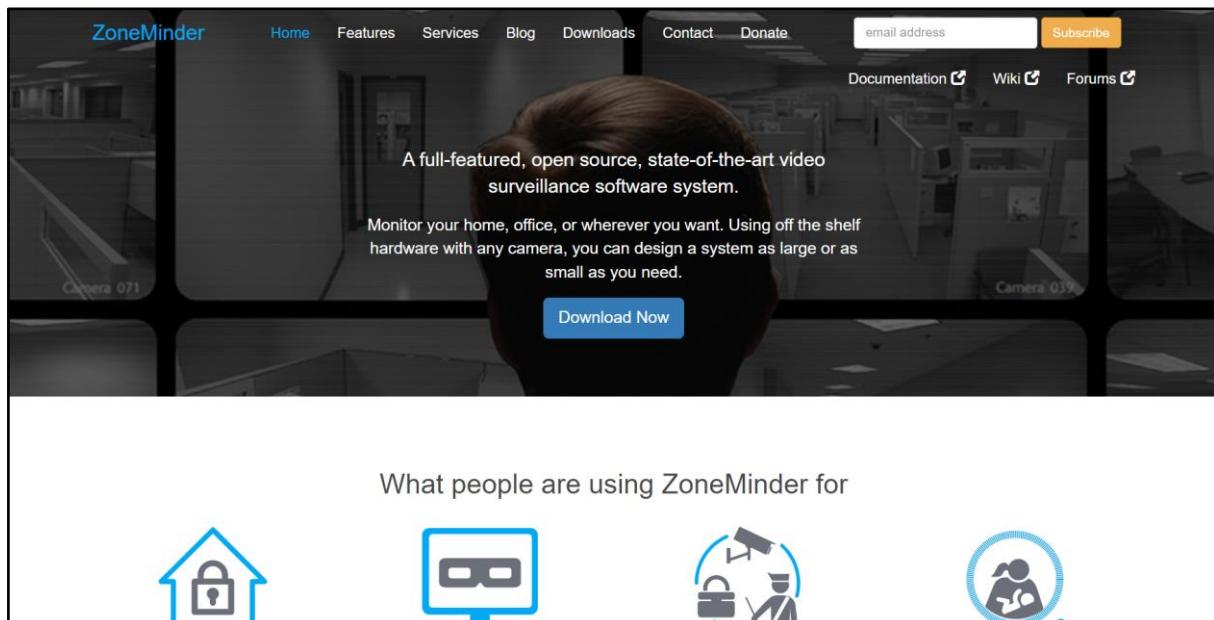


Figure 22. Existing system - Zoneminder

Key Features:

- Monitor from anywhere
- Use any camera
- You are in control of your data
- Run a really small, or super big, system
- Advanced AI-powered detection
- Actively maintained, built by those who care

Advantages:

- Open-source nature allows for community-driven development and customization.
- Supports a wide range of camera models and video formats.
- Flexible configuration options for motion detection and alerts.
- Web UI is intuitive and very responsive.

Disadvantages:

- Installation and setup can be complex for users without technical expertise.
- Requires dedicated hardware and software configuration.
- User interface can be less intuitive compared to commercial alternatives.

3.2.3 Frigate

Frigate is an open-source, self-hosted video surveillance software designed for use with Home Assistant, a popular home automation platform. It utilizes machine learning algorithms for object detection and offers features like motion detection, real-time alerts, and video recording.

Official Website: <https://frigate.video/>

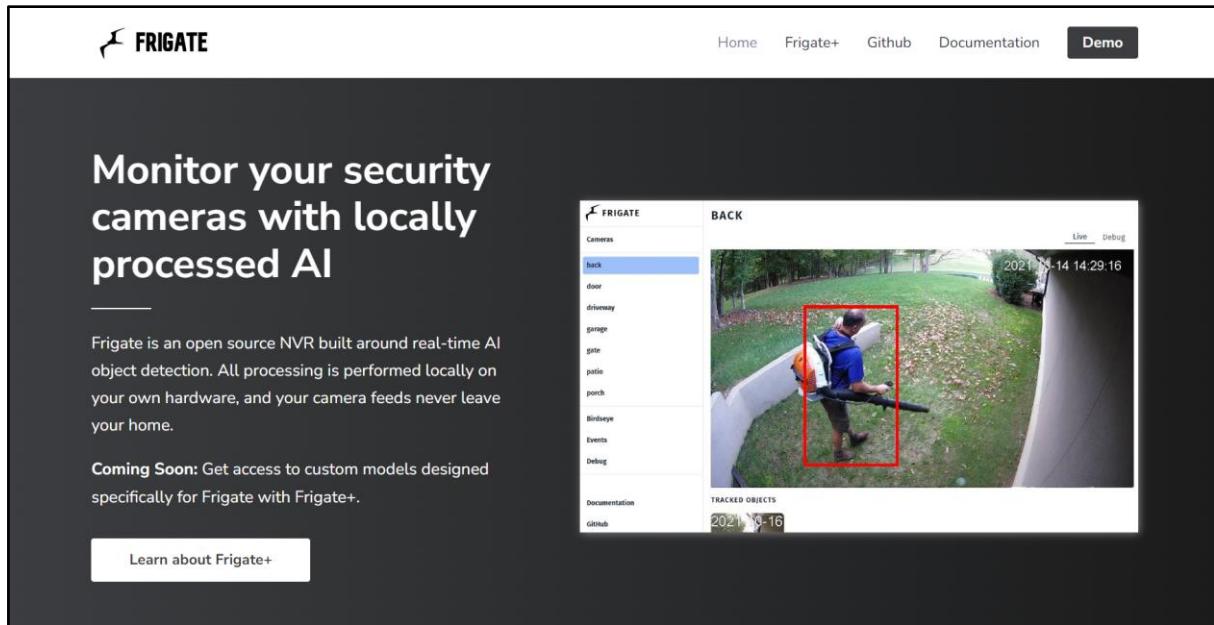


Figure 23. Existing system - Frigate

Key Features:

- Reduce false positives with local object detection
- Stop reviewing shadows and wind and start reviewing detections that matter
- Fine-tune your events and alerts with zones
- Integrate with Home Assistant and other automation platforms
- Watch a dynamic real-time video feed for your cameras

Advantages:

- Designed to integrate seamlessly with Home Assistant for a comprehensive smart home solution.
- Utilizes machine learning for advanced object detection, reducing false alerts.
- Offers flexible configuration options and supports a variety of IP cameras.

Disadvantages:

- Requires technical knowledge for installation, configuration, and maintenance.

- Relatively newer compared to established commercial solutions, may have fewer features or stability issues.

3.2.4 Viseron

Viseron is an open-source, self-hosted video management system that utilizes deep learning and computer vision algorithms to provide advanced video analytics features. It is designed to be deployed on local hardware devices using Docker.

Official Website: <https://viseron.netlify.app/>

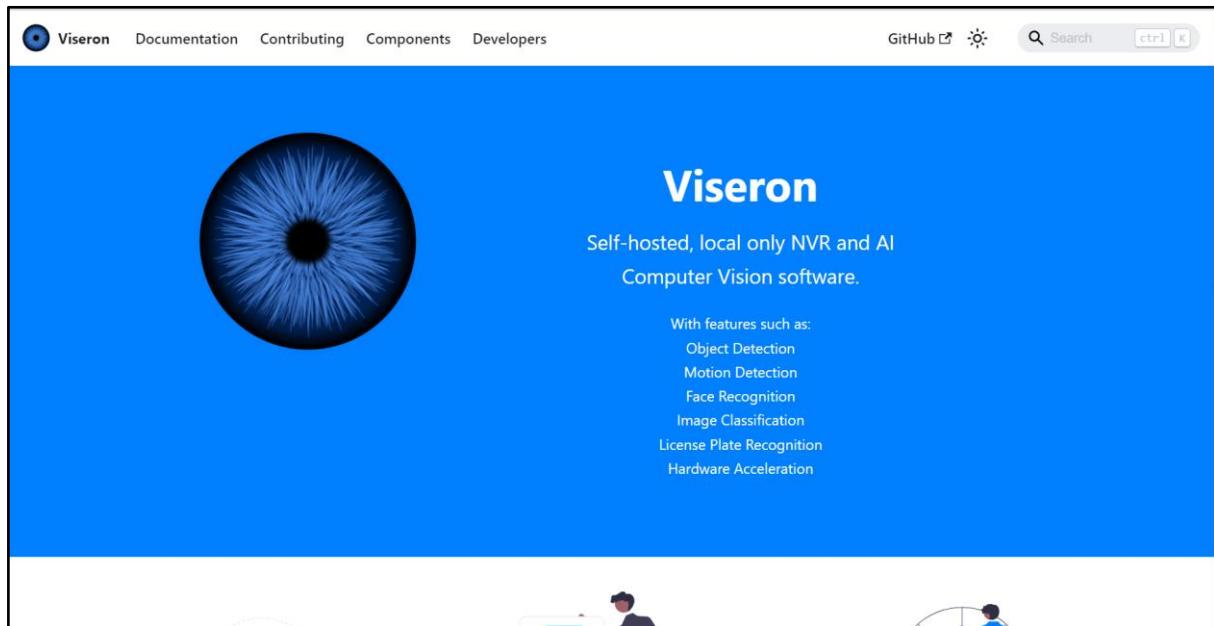


Figure 24. Existing system - Viseron

Key Features:

- Object Detection
- Motion Detection
- Fall Detection
- Image Classification
- License Plate Recognition
- Hardware Acceleration

Advantages:

- Open-source and free to use
- Supports a wide range of IP cameras and can be deployed on various platforms
- Offers advanced video analytics features like object detection, person tracking, facial recognition, and license plate recognition
- Provides mobile apps and web-based interface for remote access and monitoring
- Customizable and extensible with additional plugins and integrations

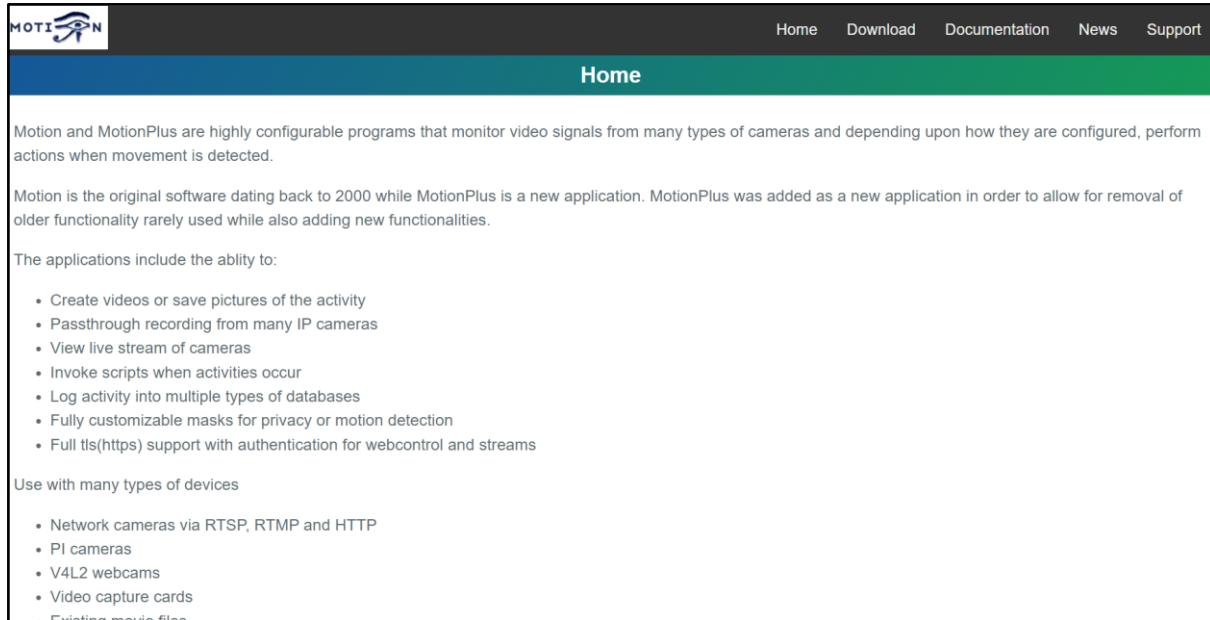
Disadvantages:

- Requires more technical expertise to set up and configure compared to some commercial solutions
- May not have the same level of enterprise-grade features and support as commercial offerings
- Ongoing development and support may not be as reliable as commercial software

3.2.5 MotionEye

MotionEye is an open-source video surveillance software that turns a single-board computer, such as Raspberry Pi, or a network-attached storage (NAS) device into a comprehensive surveillance system. It offers features like motion detection, real-time monitoring, recording, and remote access.

Official Website: <https://github.com/motioneye-project/motioneye>



The screenshot shows the official website for MotionEye. The header includes the MotionEye logo and navigation links for Home, Download, Documentation, News, and Support. The main content area has a green header bar with the word "Home". Below it, there is a section about Motion and MotionPlus, followed by a list of application features, a section on supported devices, and a "Existing movie file" link.

Motion and MotionPlus are highly configurable programs that monitor video signals from many types of cameras and depending upon how they are configured, perform actions when movement is detected.

Motion is the original software dating back to 2000 while MotionPlus is a new application. MotionPlus was added as a new application in order to allow for removal of older functionality rarely used while also adding new functionalities.

The applications include the ability to:

- Create videos or save pictures of the activity
- Passthrough recording from many IP cameras
- View live stream of cameras
- Invoke scripts when activities occur
- Log activity into multiple types of databases
- Fully customizable masks for privacy or motion detection
- Full tls(https) support with authentication for webcontrol and streams

Use with many types of devices

- Network cameras via RTSP, RTMP and HTTP
- IP cameras
- V4L2 webcams
- Video capture cards

[Existing movie file](#)

Figure 25. Existing system - MotionEye

Key Features:

- Create videos or save pictures of the activity
- Passthrough recording from many IP cameras
- View live stream of cameras
- Invoke scripts when activities occur
- Log activity into multiple types of databases
- Fully customizable masks for privacy or motion detection
- Full tls(https) support with authentication for web control and streams

Advantages:

- User-friendly web interface for easy setup and configuration.
- Supports various IP cameras and USB cameras, providing flexibility in hardware choices.

Disadvantages:

- Limited to users who are comfortable with Linux-based systems and command-line interfaces for installation and configuration.
- May lack some advanced features found in commercial surveillance systems.
- Most of the time being too basic, clunky and buggy.

3.2.6 iSpy

iSpy is a feature-rich open-source video surveillance software designed for Windows PCs. It supports a wide range of cameras, including webcams, IP cameras, and microphones. iSpy offers features like motion detection, remote access, alerting, and video recording.

Official Website: <https://www.ispyconnect.com/>

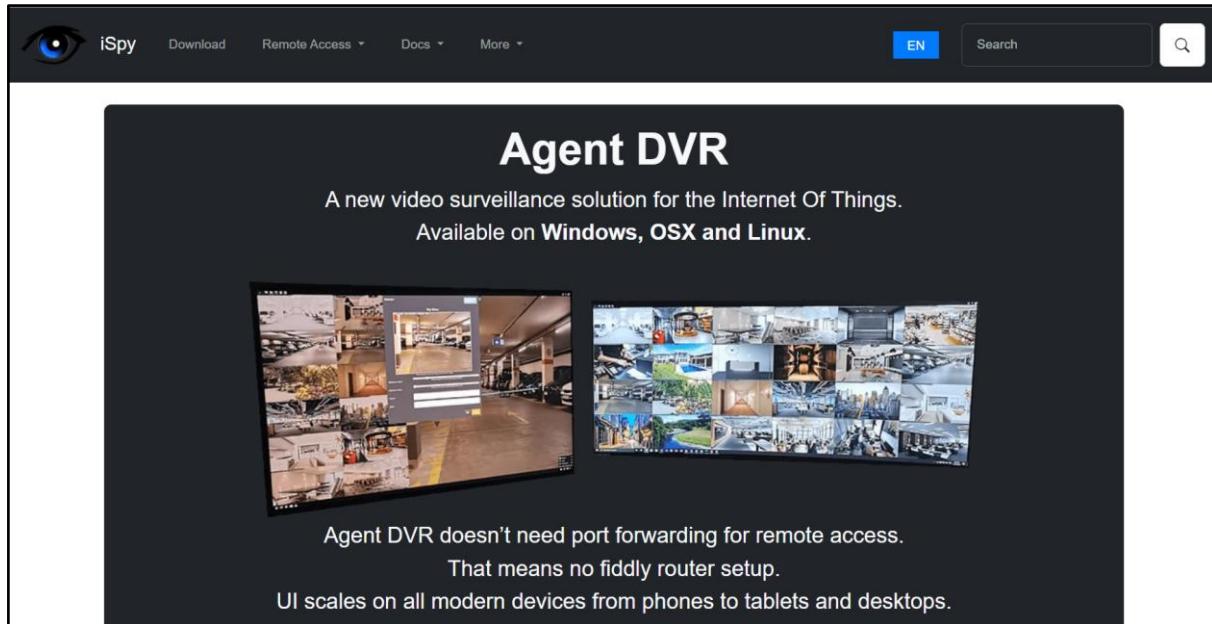


Figure 26. Existing system - iSpy

Key Features:

- Recording Modes:
 - Record on detect, alert, or demand.
 - Record by schedule or via API/external events.
 - Timelapse and adaptive modes.
 - Dual streaming for low-res live and high-res direct recording.
 - Pre and post-event buffers.
- Motion Detection:
 - Configurable areas with ONVIF motion event support.
 - Trip wires, speed, and object tracking.
 - Sound detection and recognition with Listen plugin.
 - Heat maps and counting detectors.
 - Integrates with AI tools like OpenAI, DeepStack AI, and CodeProject.AI for advanced alert filtering.
- Services:
 - FTP/SFTP and cloud uploads (Box, Google Drive, Dropbox, OneDrive).
 - MQTT, SMTP, Twitter DM, and RTMP live streaming.
 - Home Assistant, Alexa, IFTTT integrations, and a full API.
- Tools:
 - Picture-in-picture and customizable timestamps.
 - Events and actions for advanced automation.
 - Text-to-speech, PTZ control, and storage management.
- Storage Management:
 - Multi-level storage options with various media locations.
 - Max size and age limits, archival options, and disk space alerts.
 - Automatic archival and delete functions.

Advantages:

- Wide compatibility with various camera types, including both local and networked cameras.
- Offers advanced features like motion detection, alerting, and scheduling.
- User-friendly interface with customizable layouts for monitoring multiple cameras.
- Supports remote access via web interface or mobile app for monitoring on-the-go.

Disadvantages:

- Limited to Windows-based PCs, excluding users on other operating systems.
- Some advanced features may require a subscription to iSpyPRO or additional plugins, which can incur additional costs.
- Configuration and setup may be more complex compared to simpler surveillance solutions.
- Randomly stopped working

3.2.7 Shinobi

Shinobi is an open-source video surveillance software designed for home and business security monitoring. It provides features such as real-time streaming, motion detection, alerting, and video recording. Shinobi offers a web-based interface for easy access and management of surveillance cameras.

Official Website: <https://shinobi.video/>

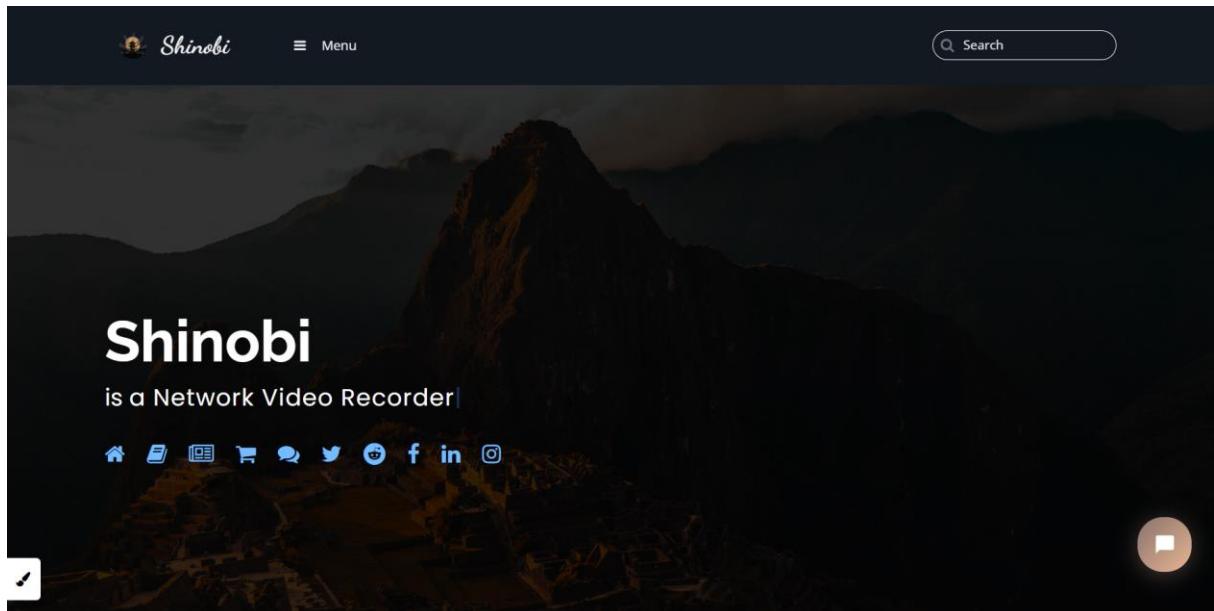


Figure 27. Existing system - iSpy

Key Features:

- Server Operations:
 - Node.js Backend: Real-time operations handled via Node.js for seamless performance.
 - Web Interface: Easy access through modern browsers for live stream viewing.
- Video and Time-Lapse Management:
 - Recording Management: Curate videos with automatic purging based on age or storage limits.
 - Time-Lapse: Create time-lapses and compile them into videos easily.
- User Management:
 - Tiered System: Allows different privilege levels for users.

- Multitenancy: Admin accounts manage cameras independently.
- Advanced Features:
 - Motion and Object Detection: Trigger recordings or other actions.
 - 2-Factor Authentication: Enhanced security for user accounts.
 - RESTful API: Full access to the platform's features and customization.
- Camera and Stream Management:
 - ONVIF Scanner: Quickly detect and add cameras.
 - Multicast Streams: Efficiently splits one stream into various outputs.
 - PTZ Control: Remotely adjust camera angles.
- Mobile and Cross-Platform:
 - Mobile App: Manage and view streams remotely.
 - Cross-Platform: Runs on Linux, Mac, and Windows.

Advantages:

- Supports a wide range of IP cameras and hardware devices.
- Offers features like motion detection, alerting, and video recording.
- Web-based interface allows for remote access and management from any device with a browser.

Disadvantages:

- Installation and setup may require some technical knowledge, particularly for users not familiar with Linux-based systems.
- User interface and configuration options may be overwhelming for beginners.
- Slow when trying to seek saved videos.

4. Business Opportunity

In the current market, traditional surveillance methods like static CCTV cameras dominate the home security solutions landscape. These systems, while providing a basic level of security, lack the advanced features necessary for today's technology-driven society. A major drawback of these outdated systems is the high number of false alarms and unnecessary notifications, which lead to wasted resources and time. Additionally, these systems suffer from limited remote access and control capabilities, restricting their overall effectiveness[29].

Amidst this backdrop, Vigision introduces a significant market opportunity with its ability to transform basic camera systems into proactive guardians. By utilizing advanced AI algorithms and deep learning, Vigision enhances motion detection, such as identifying falls, which significantly reduces the occurrence of false alarms. Moreover, its capability to self-host on local hardware devices not only facilitates remote access from anywhere in the world but also caters to the growing consumer demand for data privacy by storing data locally[29].

The market has recently witnessed a surge in the development of smart home security solutions that incorporate AI and machine learning. However, systems like Vigision that combine such advanced features are still relatively rare. Unlike traditional systems that require physical presence for operations, Vigision offers the convenience of setting up and receiving alerts remotely, adding a layer of mobility and convenience for the users. This addresses key issues in the home security domain, notably the frequent false alarms generated by conventional motion detection systems and the privacy concerns associated with cloud-based surveillance[30].

The strategic direction towards AI-integrated surveillance software such as Vigision aligns perfectly with current market trends that lean towards AI-based technologies, smart home functionalities, and stringent data privacy requirements. By continually developing features that resonate with these market dynamics, Vigision is poised to maintain its competitive edge and lead in the evolving home security market [29].

5. Software Product Vision

Our product, Vigision - Real-time Intelligent Video Surveillance System, heralds a paradigm shift in home security, ushering in a future where safety and peace of mind are seamlessly integrated into everyday life. Our vision encapsulates both the aspirations of our diverse customer base and the strategic direction of our organization, grounded in the realities of contemporary technological landscapes and user needs.

The envisioned world with the Vigision - Real-time Intelligent Video Surveillance System is homes become sanctuaries fortified by the power of AI-driven surveillance. Our product transforms traditional home camera systems into proactive guardians, leveraging Deep Learning for features like fall detection, object recognition. In this envisioned future, homeowners no longer fret over incessant notifications but are instead alerted only to significant events, fostering a sense of security without inundating users with unnecessary information.

This vision is not merely idealistic; it's a pragmatic response to the growing need for enhanced home security and safety. It aligns with the evolving landscape of smart home technologies. By prioritizing user-centric design and cutting-edge technology, Vigision empowers homeowners to safeguard their loved ones and property with confidence.

In the world with Vigision - Real-time Intelligent Video Surveillance System, homes are fortified, families are protected, and peace of mind is paramount. It's a future where technology harmonizes with our most fundamental need: safety and security.

6. Project Scope & Limitations

6.1 Major Features

FE-01. Cameras & Surveillance Features: This feature allows users to view connected cameras, live streams from all cameras, and toggle camera detection, recording, and snapshot features. Users can also view the detail of live cameras and access the system in full-screen mode.

FE-02. AI Computer Vision Features: The real-time AI computer vision features provide users with motion detection, object detection, object tracking, pose estimation and human fall detection capabilities. Users can view defined masks and zones, manage motion and object masks, and fine-tune motion detection settings.

FE-03. AI Feature Setting and Debugging: This feature enables users to view various debug options, such as bounding boxes, motion boxes, regions, zones, and masks. Admins can manage zones, debug system issues, and ensure the AI functionalities are working optimally.

FE-04. Event Management: This feature allows users to view events and event details. Admins can delete events, mark events as reviewed, and download event snapshots and recordings. Users can also view motion events and alerts/detections.

FE-05. Recording Features: This feature allows admins to set up recordings, enable or disable recording for cameras, and download recorded footage. Admins can also export event recordings and camera recordings, as well as view export details and manage exports.

FE-06. Configuration: The configuration feature enables users to view and apply pre-configured settings. Administrators can configure app settings, create and save custom configurations, and apply them across the system.

FE-07. Statistics Dashboard: The statistics dashboard allows users to monitor system performance, view storage information, and access system metrics. Admins and Members can View live cameras and system metrics to keep track of the system's health and performance.

FE-08. Authentication: The system only accepts access from authorized people. Admins can handle user management tasks, including viewing the list of users, creating new users, updating user details, and deleting users.

FE-09. General Features: This feature includes options for members and admins to toggle light/dark mode, enable automatic dark mode. Users can also change the application theme according to their preference, enhancing the user experience.

FE-10. User Management: The user management feature is accessible only to admins, who can manage users, review user feedback, and send notifications to users. Admins have complete access to all features and functionalities, ensuring the system's security and efficiency.

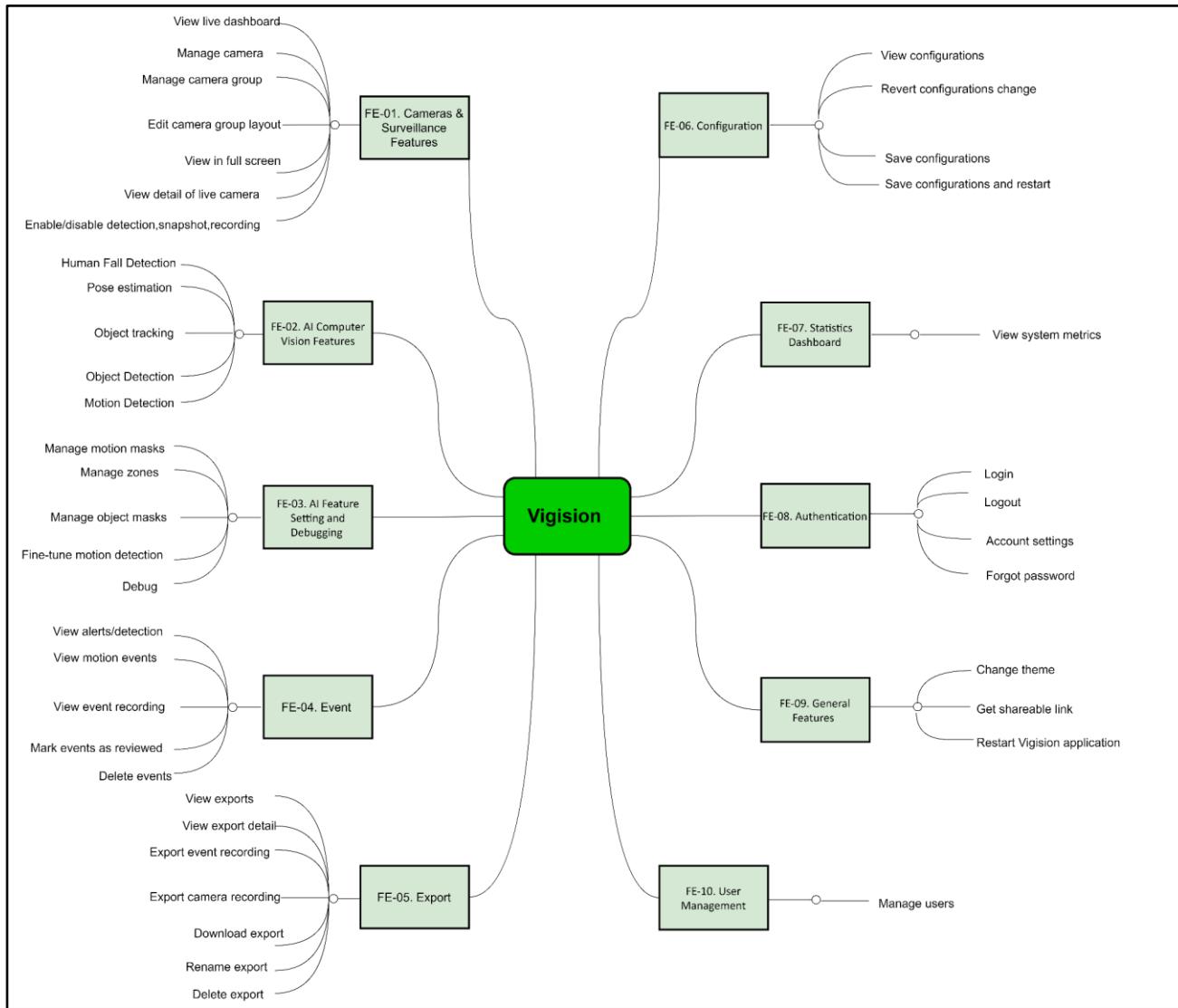


Figure 28. Major Feature Tree

6.2 Limitations & Exclusions

- LI-01: No Cloud Storage: Vigision stores data locally, lacking cloud-based storage or remote access.
- LI-02: No Mobile App: Initial release lacks a dedicated mobile app; users access surveillance via web interface.
- LI-03: No Voice/Audio Support: Focus is on video surveillance; no support for two-way audio or voice control.
- LI-04: Integration Limitations: Third-party integration may be limited initially. Deeper integrations with home automation, security systems, or IoT devices are not included in the core functionality.
- LI-05: Scalability Constraints: The system targets small to medium-sized residential and small business deployments, not large-scale commercial or enterprise-level systems with hundreds of cameras.
- LI-06: Offline Operation: Functionality may be limited during network/power outages. The system may not be able to provide real-time alerts or remote access during such disruptions.
- LI-07: Ongoing Maintenance: Users are responsible for local hardware and software upkeep; Vigision lacks a managed cloud service with automatic updates.

II. Project Management Plan

1. Overview

1.1 Scope & Estimation

Table 4. Scope & Estimation

#	WBS Item	Complexity	Est. Effort (man-days)
1	<i>Project management & Planning</i>		7
1.1	Define project scope and objectives	Medium	1
1.2	Develop project schedule and milestones	Complex	3
1.3	Identify and manage project risks	Medium	2
1.4	Establish communication plan & reporting procedures	Simple	1
2	<i>Requirement analysis</i>		7
2.1	Gather and analyze user requirements	Medium	2
2.2	Define system functional and technical specifications	Complex	5
3	<i>Documentation</i>		99
3.1	Report1_Project Introduction	Medium	2
3.2	Report2_Project Management Plan	Medium	4
3.3	Report2_Sample Project Schedule	Medium	4

3.4	Report3_Software Requirement Specification	Complex	21
3.5	Report3_Project Tracking	Medium	21
3.6	Report4_Software Design Document	Complex	21
3.7	Report5_Test Documentation	Medium	14
3.8	Report5_Test Report	Medium	7
3.9	Report6_Software User Guides	Simple	2
3.10	Report7_Final Project Report	Medium	3
4	<i>Design</i>		30
4.1	Design system architecture	Complex	7
4.2	Design database	Medium	5
4.3	Design prototype	Medium	7
4.4	Design AI workflow	Medium	4
4.5	Design AI network architectures	Complex	7
5	<i>Implement</i>		110
5.1	Recording Management Processing	Complex	4
5.2	Review Segment Management Processing	Complex	4
5.3	Object Detection Processing	Medium	2
5.4	Detected Frame Processing	Complex	7
5.5	Video Output Processing	Simple	1
5.6	Camera frames processing	Complex	3
5.7	Camera capture processing	Simple	1
5.8	Timeline Processing	Simple	1
5.9	Event Processing	Simple	1
5.10	Event Cleanup Processing	Medium	2
5.11	Recording Cleanup Processing	Complex	4
5.12	Storage Maintenance Processing	Complex	5
5.13	Stats Emission Processing	Complex	5
5.14	Start Vigision Watchdog	Medium	2
5.15	Login	Medium	2
5.16	Logout	Medium	2

5.17	Forgot password	Medium	2
5.18	Create user	Medium	2
5.19	Account setting	Medium	2
5.20	Update password	Medium	2
5.21	Delete user	Complex	4
5.22	Update email	Simple	2
5.23	Enable/disable receive alert by email	Medium	2
5.24	View live cameras	Complex	3
5.25	Manage cameras	Simple	1
5.26	Manage camera groups	Medium	1
5.27	Edit camera group layout	Medium	1
5.28	View in full screen	Simple	1
5.29	View detail of live camera	Medium	2
5.30	Enable/disable detection	Medium	2
5.31	Enable/disable recording	Medium	1
5.32	Enable/disable snapshots	Complex	7
5.33	View defined masks and zones	Simple	1
5.34	Manage motion masks	Medium	1
5.35	Manage zones	Medium	1
5.36	Manage object masks	Medium	2
5.37	Fine-tune motion detection	Medium	2
5.38	Debug	Medium	3
5.39	View alerts/detections	Medium	3
5.40	Config	Simple	1
5.41	View system metrics	Medium	2
5.42	Restart application	Medium	1
5.43	Change application theme	Medium	1
5.44	Get shareable link	Simple	1
6	<i>Testing and maintenance</i>		30

6.1	Integration testing	Medium	7
6.2	System testing	Medium	14
6.3	Acceptance test	Medium	4
6.4	Fix bugs	Medium	5
7	Deployment		14
7.1	Deploy system to production environment	Complex	7
7.2	Address and resolve deployment issues	Medium	7
8	Maintenance		14
8.1	Maintain system performance	Medium	7
8.2	Maintain and update software documentation	Medium	7

Total Estimated Effort (man-days) **311**

1.2 Project Objectives

The primary objective of this project is to develop an AI-powered video surveillance system called Vigision. This system aims to enhance traditional home security by leveraging deep learning algorithms for advanced features such as object detection and fall detection. Important phases of the project will include initiating, planning, executing (which includes analysis, design, implement, and testing), monitoring and controlling, and closing. Below is a summary of the precise target numbers for each step.

Table 5. Project Objectives

#	Stage	Milestone	Notes
1	Initiation	Conduct Feasibility Study	Analyze the technical, economic, and operational feasibility of the project.
2		Create Project Charter	Define project objectives, scope, and stakeholders.
3		Organize kick-off meeting	Introduce team, review charter, discuss plan.
4		Get project charter approval	Obtain signatures from key stakeholders.
5		Create stakeholder register	Identify stakeholders, roles, and communication plans.
6	Planning	Complete Scope Management Plan	Define project scope, deliverables, and change control.
7		Complete Time Management Plan	Develop a project schedule with milestones and deadlines.
8		Complete Risk management plan	Identify risks and mitigation strategies.
9		Meeting with team to discuss about plans	Review plans, discuss roles, and address concerns.

10		Deliver Project Management Plan	Finalize and distribute project plans to stakeholders.
11	Executing	Set up development environment and tools	Install software, configure hardware, and establish coding standards.
12		Analyze software requirement	Gather user requirements and define functional/non-functional requirements.
13		Define sprint backlog and task	Break down work into sprints, create backlog with tasks, and assign tasks.
14		Implement system architecture and design	Design system architecture and develop specifications.
15		Develop and test software modules	Implement modules, conduct unit testing, and fix bugs.
16		Integrate system components	Integrate modules and hardware, conduct integration testing.
17		Conduct integration, system and acceptance testing	Perform thorough testing and address issues.
18		Deploy system to production environment	Deploy system, monitor, and provide support.
19	Monitoring and Controlling	Monitor project progress and performance against plan	Track progress and address deviations.
20		Identify and address any deviations from the plan	Analyze reasons for deviations, develop corrective actions, and update plans.
21		Manage project risks and issues	Track risks, implement mitigation strategies, and resolve issues.
22	Closing	Create project final report	Document accomplishments, lessons learned, and metrics.
23		Conduct lesson learn	Discuss successes, challenges, and areas for improvement.
24		Deliver final project documentation and deliverables	Deliver all project documentation and deliverables.
25		Close out project	Formally close the project, release resources, and archive files.

1.3 Project Risks

Table 6. Project Risks

#	Risk Description	Impact	Possibility	Response Plans
1	Technical complexity: The project involves complex technologies such as AI, deep learning, and computer vision. These technologies may be	High	High	<ul style="list-style-type: none"> - Conduct thorough feasibility studies and technology assessments. - Break down the project into smaller, more manageable tasks.

	difficult to implement and integrate, leading to delays.			- Conduct thorough testing and debugging throughout the development process. - Seek expert advice and support when needed.
2	Data privacy concerns: The project involves collecting and processing sensitive data, such as video footage and personal information. This raises concerns about data privacy and security.	High	Medium	- Implement robust data security measures, including encryption, access controls, and data anonymization. - Obtain necessary consents and comply with relevant data privacy regulations. - Conduct regular security audits and vulnerability assessments. - Develop a clear data privacy policy and communicate it effectively to stakeholders.
3	Hardware and software compatibility: The system should be compatible with a wide range of operating systems, and hardware configurations.	Medium	Low	- Thoroughly test the system with different hardware and software configurations. - Provide clear documentation and support for users to ensure compatibility. - Consider offering pre-configured hardware bundles to simplify deployment.
4	Project schedule delays: The project may face unexpected delays, such as technical challenges or unforeseen circumstances.	Medium	Low	- Develop a realistic project schedule and identify potential risks. - Implement risk mitigation strategies to address potential delays. - Monitor project progress closely and adjust the schedule as needed.

2. Management Approach

The development of Vigision will be guided by the Agile Scrum methodology. This iterative and incremental approach prioritizes collaboration, flexibility, and continuous improvement, making it ideal for projects with evolving requirements and complex features. Scrum offers numerous benefits for software development projects, making it a compelling choice for teams seeking an agile and collaborative approach. Here are some key reasons to consider using Scrum:

- Increased Flexibility and Adaptability: Scrum's iterative nature allows teams to respond quickly to changing requirements. With short sprints and regular feedback loops, the team can adjust course and prioritize features based on the latest insights, ensuring the project remains relevant and delivers value.

- Enhanced Focus and Productivity: Scrum promotes a focused approach by breaking down work into manageable sprints. This helps teams stay on track, avoid distractions, and deliver working software increments at regular intervals. The clear goals and timeframes of each sprint boost productivity and motivation.
- Improved Transparency and Collaboration: Scrum emphasizes transparency through regular meetings, shared backlogs, and visible progress tracking. This fosters open communication, collaboration, and trust among team members, stakeholders, and clients. Everyone is kept informed and involved, leading to better decision-making and a shared sense of ownership.
- Reduced Risk and Increased Predictability: By delivering working software in short iterations, Scrum allows for early feedback and identification of potential issues. This reduces the risk of costly rework and ensures the project stays on track. The regular sprint reviews and retrospectives provide opportunities to learn from successes and failures, continuously improving the process and predictability of outcomes.
- Continuous Improvement: Scrum promotes a culture of continuous improvement. The regular retrospectives provide opportunities for the team to reflect on their processes, identify areas for improvement, and adapt their approach for future sprints. This constant learning and adaptation ensure the team becomes more efficient and effective over time.
- Improved Team Morale and Motivation: Scrum's collaborative and transparent nature fosters a positive and supportive team environment. Team members feel empowered to contribute, share ideas, and take ownership of their work. This leads to increased motivation, engagement, and a sense of accomplishment.

By embracing the Agile Scrum methodology, the Vigision team can deliver high-quality software that meets the needs of the stakeholders while maintaining flexibility and adaptability throughout the development process.

2.1 Project Process

Scrum employs a structured framework consisting of a prioritized product backlog, sprint backlogs for focused development, and short sprints (typically 2-4 weeks) where the team works on completing specific tasks. Each sprint begins with planning, followed by execution, review, and adaptation based on feedback. This ensures a responsive and client-centric development process.

Key roles in the project include the Product Owner, responsible for prioritizing the product backlog and ensuring the team focuses on high-value items; the Scrum Master, who facilitates the Scrum process and removes obstacles; and the Development Team, responsible for developing the software and delivering working features at the end of each sprint.

Effective communication and collaboration are crucial for success. The team will utilize various tools and techniques, including regular meetings, collaboration tools, and documentation like user stories and sprint burndown charts.

Progress will be monitored using metrics like sprint burndown charts, velocity (measuring the team's average completion rate), and quality metrics. Regular project reviews will identify areas for improvement.

Potential risks will be identified and assessed, with mitigation plans developed to address them. The team will continuously strive to improve the development process by identifying areas for improvement and implementing changes.

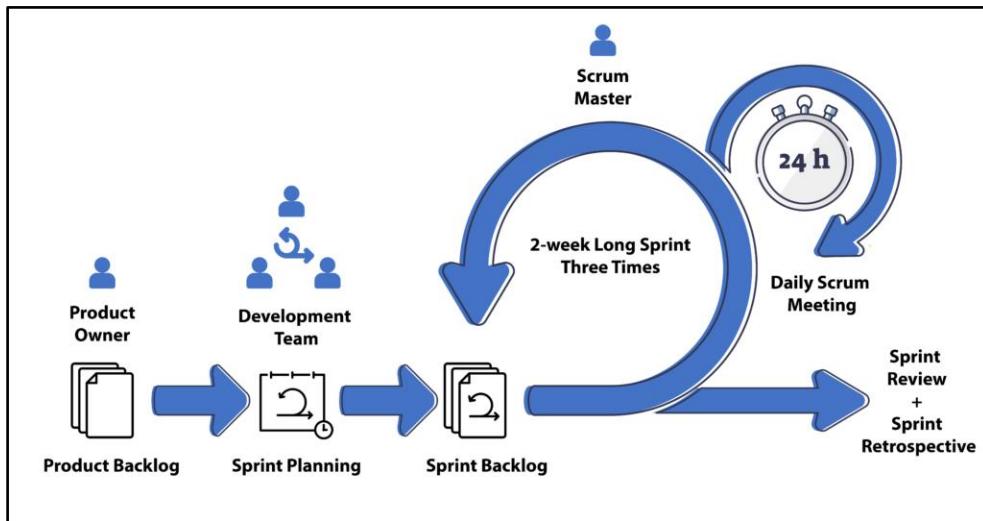


Figure 29. Project Process - Scrum

2.2 Quality Management

Vigision prioritizes delivering a high-quality product that meets user expectations and operates reliably. To achieve this, we will implement a comprehensive quality management approach encompassing the following aspects:

- Defect Prevention: The development team will employ proactive measures to identify and address potential issues early in the software development lifecycle. This includes thorough code reviews, automated unit tests, and rigorous static code analysis to catch bugs and code quality problems before they manifest in the final product.
- Reviewing: Regular code reviews will be conducted by the team to ensure adherence to coding standards, best practices, and the project's technical requirements. This collaborative approach allows for the identification and resolution of issues, knowledge sharing, and the maintenance of code quality throughout the development process.
- Integration Testing: Comprehensive integration testing will be performed to verify the seamless interoperability of the various components and technologies that make up the Vigision system. This testing will validate the correct functioning of the data flow, communication protocols, and overall system integration, identifying and resolving any integration-related defects.
- System Testing: Rigorous system testing will be carried out to validate the end-to-end functionality of the Vigision system. This includes testing the system's performance, security, reliability, and compliance with the defined requirements. The testing will be conducted in a staged approach, starting with component-level testing and progressing to integration and system-level testing to ensure the overall quality and stability of the final product.

By incorporating these quality management approaches, the Vigision project team will strive to deliver a high-quality, reliable, and secure intelligent video surveillance system that meets the high standards of quality.

2.3 Training Plan

Table 7. Training Plan

Training Area	Participants	When, Duration	Waiver Criteria
Docker	Phan Hong Phuc	06/05/2024, 3-day	Mandatory

	Vo Minh Dat Nguyen Le Quang Thinh Dinh The Anh		
Flask	Phan Hong Phuc Vo Minh Dat Nguyen Le Quang Thinh Dinh The Anh	09/05/2024, 5-day	Mandatory
React	Vo Minh Dat Dinh The Anh	14/05/2024, 4-day	Mandatory
TypeScript	Phan Hong Phuc Vo Minh Dat Nguyen Le Quang Thinh Dinh The Anh	18/05/2024, 2-day	Mandatory
NGINX	Phan Hong Phuc Vo Minh Dat Nguyen Le Quang Thinh	22/05/2024, 1-day	Mandatory
Tailwind CSS	Phan Hong Phuc Vo Minh Dat Dinh The Anh	23/05/2024, 1-day	Mandatory
PyTorch	Vo Minh Dat Dinh The Anh Nguyen Le Quang Thinh	24/05/2024, 1-day	Mandatory
Git, GitHub	Dinh The Anh	25/05/2024, 1-day	Mandatory

3. Project Deliverables

Table 8. Project Deliverables

#	Deliverable	Due Date	Notes
1	Report 1_Project Introduction	09/05/2024	
2	Report 2_Project management plan	16/05/2024	
3	Report 3_Software Requirement Specification	23/05/2024	
4	Report 4_Software Design Document	30/05/2024	
5	Code Iteration 1 Source code (FE, BE), SDD, Test Report, User Guides: – Authentication – User management	20/06/2024	

	- Cameras & Surveillance Features		
6	Code Iteration 2 Source code (FE, BE), SDD, Test Report, User Guides: - AI Computer Vision Features - Event - Export - AI Feature Setting and Debugging	04/07/2024	
7	Code Iteration 3 Source code (FE, BE), SDD, Test Report, User Guides: - AI Feature Setting and Debugging - Configuration - Statistics Dashboard - General Features	11/07/2024	
8	Report 5_Test Document	16/07/2024	
9	Report 5_Test Report	16/07/2024	
10	Report 6_Software User Guides	18/07/2024	
11	Report 7_Final Project Report	20/07/2024	
12	Final source code	20/07/2024	
13	Final presentation slide	25/07/2024	

4. Project Organization

4.1 Team & Structures

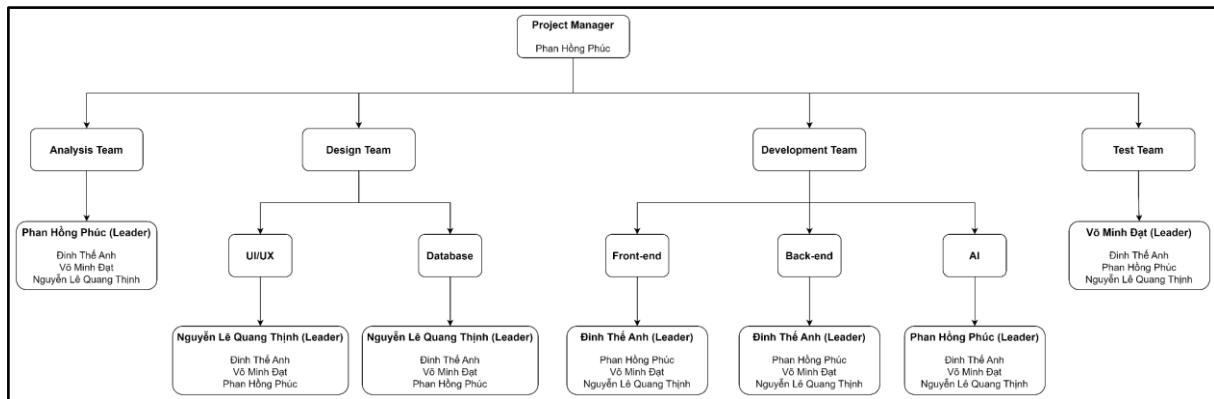


Figure 30. Project Organization - Team & Structures

4.2 Roles & Responsibilities

Table 9. Project Organization - Roles & Responsibilities

Role	Role Responsibility
Project Manager	The Project Manager oversees the planning, execution, and completion of the project, ensuring effective management of resources, risks, and stakeholders to successfully meet project objectives.
Analysis Leader	The Analysis Leader is tasked with collecting, analyzing, and documenting business requirements and converting these into technical specifications for the project.
Analysis Member	The Analysis Member aids in the collection and documentation of business requirements, performs research, and offers insights to enhance the project's analytical efforts.
Design Leader	The Design Leader steers the project's design initiatives, collaborates with stakeholders, and supervises the production of high-quality visual concepts and user interface designs.
Design Member	The Design Member supports the creation of visual concepts and user interface designs, works under the guidance of the Design Leader, and contributes to the project's design process.
Development Leader	The Development Leader manages the project's technical dimensions, leads the development team, and ensures the creation of high-quality software solutions that fulfill project specifications.
Development Member	The Development Member engages in coding, implementing, and testing software solutions according to project specifications, and collaborates with the Development Leader and other team members to ensure quality outcomes.
Test Leader	The Test Leader directs testing operations, devises test strategies, oversees the testing team, and ensures the delivery of high-quality software through robust testing procedures.
Test Member	The Test Member carries out testing, detects and reports defects, and aids in enhancing the software quality through effective testing activities.

5. Project Communications

Table 10. Project Communications

Communication Item	Who/ Target	Purpose	When, Frequency	Type, Tool, Method(s)
Daily Scrum	Phan Hong Phuc, Dinh The Anh, Vo Minh Dat, Nguyen Le Quang Thinh	Synchronize our activities, discuss progress, and plan our work for the next 24 hours.	Every day, 19h	Text, Google Meet
Sprint review	Mentor, Phan Hong Phuc, Dinh The Anh, Vo Minh Dat, Nguyen Le Quang Thinh	Showcase the features, gather feedback, refine the backlog, and improve the product and team processes.	Every weekend, 19h	Microsoft Teams

6. Configuration Management

6.1 Document Management

Google Docs and Google Sheets are powerful, web-based tools developed by Google, designed to enhance productivity and collaboration. Google Docs is a word-processing application that allows users to create, edit, and share documents online, while Google Sheets is a spreadsheet application that offers similar capabilities for managing and analyzing data. Both tools support real-time collaboration, enabling multiple users to work on the same document or spreadsheet simultaneously, regardless of their location.

Principal advantages of Google Docs and Google Sheets include:

- Real-time Collaboration: Both Google Docs and Google Sheets enable multiple users to edit and collaborate on the same document or spreadsheet at the same time. Changes are instantly visible to all users, promoting seamless teamwork and ensuring that everyone has access to the latest information.
- Accessibility: As cloud-based applications, Google Docs and Google Sheets can be accessed from any device with internet connectivity. This flexibility allows users to work from anywhere, whether they're using a desktop, laptop, tablet, or smartphone.
- Version History: These tools automatically save a revision history, allowing users to track changes, view previous versions, and revert to earlier iterations if necessary. This feature ensures that critical information is preserved and that users can always access and restore previous versions of their work.
- Comments and Discussions: Both platforms allow users to add comments and suggestions directly within the document or spreadsheet. This feature facilitates effective communication and feedback, streamlining the review and approval process and enabling team members to address concerns and provide input efficiently.
- Integration: Google Docs and Google Sheets integrate seamlessly with other Google Workspace tools, such as Google Drive, Gmail, and Google Calendar. This integration enhances productivity by allowing users to easily share documents, schedule meetings, and manage tasks within a unified ecosystem.

- Sharing and Permissions: These tools offer granular control over sharing and access permissions, allowing users to specify who can view, edit, or comment on their documents and spreadsheets. This ensures that sensitive information is protected and that only authorized users have access to specific content.

By utilizing Google Docs and Google Sheets, our team can manage documents and data more effectively, collaborate in real-time, and maintain a high level of organization and security. These tools provide a robust platform for creating, editing, and sharing content, making them indispensable for modern, collaborative work environments.

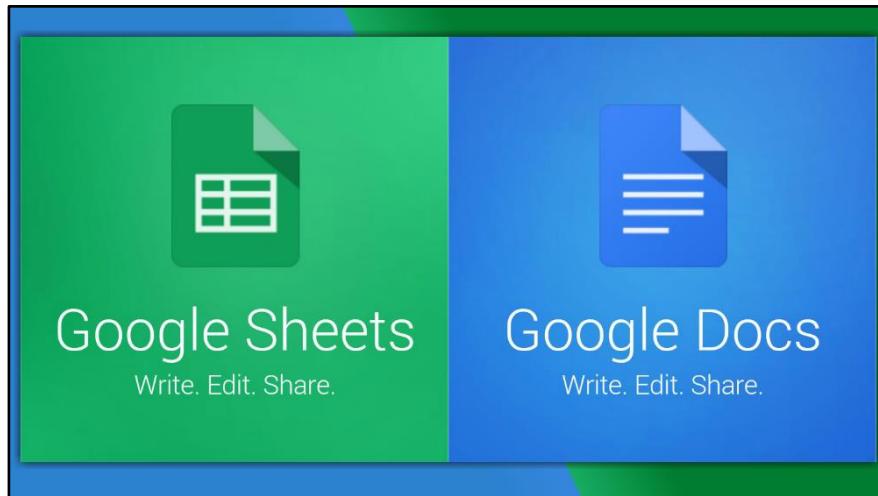


Figure 31. Document Management - Google Docs and Google Sheets

6.2 Source Code Management

Our team leverages GitHub for effective source code management, using its robust tools and collaborative features to streamline our development processes. GitHub acts as a web-based platform and version control repository, providing a comprehensive toolkit for managing software development projects. Here's our detailed workflow with GitHub:

- Repository Setup: The team leader sets up the initial repository on GitHub, ensuring all necessary configurations are in place. Team members then clone this repository to their local machines, creating their local development environment.
- Branching Strategy: For every new task, whether it's a feature or bug fix, team members create a dedicated branch from the main branch (commonly "master" or "main"). This branching strategy helps keep the development of different tasks isolated and organized.
- Local Development: Team members work on their respective branches, making code changes, adding new files, and performing necessary modifications. They commit their changes frequently to their local branch, ensuring that work is saved incrementally.
- Push to Remote Branch: Upon completing a significant part of their work, team members push their commits from the local branch to the corresponding remote branch on GitHub. This keeps the remote repository updated with the latest progress.
- Pull Request (PR) Initiation: After finishing a task, team members create a pull request (PR) to merge their branch into the main branch. This PR summarizes the changes made and sets the stage for code review.
- Peer Review Process: The pull request is reviewed by other team members. Reviewers provide feedback, request changes if necessary, and engage in discussions to ensure the code meets quality standards and project requirements.

- Incorporate Feedback: If reviewers suggest changes, the developer incorporates the feedback and updates the pull request accordingly. This iterative process continues until the code meets the required standards and is approved by the reviewers.
- Merging the PR: Once the pull request is approved, a team member with merging permissions merges the branch into the main branch. Automated tests and continuous integration (CI) pipelines are triggered to verify that the new code integrates smoothly with the existing codebase.
- Deployment Staging: For features ready for testing, the code is merged into a staging branch and deployed to a staging environment. This stage involves comprehensive testing to catch any issues before production deployment.
- Final Deployment: After successful testing in the staging environment, the code is merged into the production branch and deployed live. This final step ensures that new features and fixes are available to end-users.
- Continuous Improvement: The process is iterative, with new branches being created for each new task. Team members continually work on improving the project, following this structured workflow to maintain organization and efficiency.



Figure 32. Source Code Management - GitHub

6.3 Tools & Infrastructures

Table 11. Tools & Infrastructures

Category	Tools / Infrastructure
Technology	React (FE), Tailwind CSS (FE), Flask (BE), FFmpeg (BE)
Database	SQLite
IDEs/Editors	Visual Studio Code
Diagramming	Draw.io, Visual Paradigm
Documentation	Google Docs/Sheets
Version Control	GitHub (Source Code), Google Drive (Documents)
Deployment server	NGINX, Docker
Project management	Jira (Task/Schedule), GitHub (Code, Defects)

III. Software Requirement Specification

1. Overall Description

1.1 Product Overview

Vigision, the Real-time Intelligent Video Surveillance System, is a transformative Network Video Recorder (NVR) and AI Computer Vision software designed to revolutionize home security. It leverages the power of Deep Learning to provide enhanced features such as fall detection, object recognition. It can be self-hosted on local hardware devices, offering flexibility and convenience. The Vigision system architecture comprises several entities: Admin, Member, as well as the external entities: Cameras, Google Mail. Each entity interacts with the Vigision system uniquely, contributing to its comprehensive functionality. The context diagram below illustrates the interaction of these system entities.

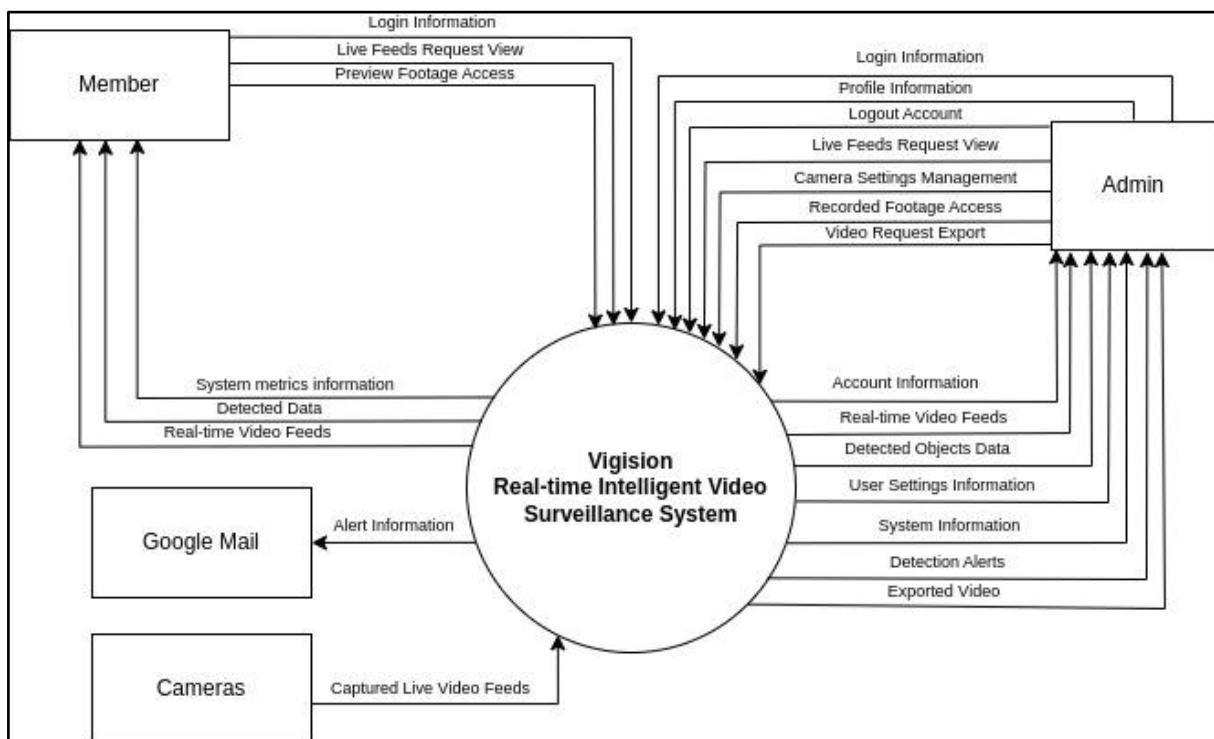


Figure 33. System Context Diagram

1.2 Business Rules

Table 12. Business Rules

ID	Rule Definition
BR-01	Admins must have access to all system features, including user management, camera management, system configuration, viewing system metrics, and accessing live and recorded camera feeds.
BR-02	Members should have restricted access, primarily for viewing purposes. They can log in and log out, view the live dashboard, access the system in full-screen mode, and view detailed live camera feeds and system metrics.
BR-03	Admins are responsible for user management, including creating new users, updating user details, and deleting users. Usernames must be at least 6 characters long and must include letters, numbers, dots, or underscores. Passwords must be at least 8 characters long and include at least one uppercase letter, one special character, and one number.

BR-04	Members cannot manage users. They are restricted to viewing live camera feeds and system metrics, and changing the application theme.
BR-05	Only Admins have the authority to manage user accounts.
BR-06	Users can toggle detection, recording, and snapshot features on their connected cameras.
BR-07	Changes to camera settings must be logged for auditing purposes.
BR-08	The system must utilize deep learning algorithms for accurate detection.
BR-09	All significant events detected by the system must be logged and accessible to users.
BR-10	Users can configure alert settings based on detection rules and thresholds.
BR-11	The system should send alerts to users only when their specific rules and conditions are met.
BR-12	All alerts must be based on user-configured rules and settings. Users can review and modify alert settings at any time.
BR-13	Recorded footage must be downloadable and securely stored.
BR-14	The statistics dashboard must accurately display real-time system performance metrics, including storage information, ensuring the dashboard reflects the current system performance and storage details for users' monitoring needs.
BR-15	The oldest 2 hours of recordings will be deleted if there is less than an hour left of storage.
BR-16	Users have full access to manage their profile, view and manage their camera streams, set up recordings, and configure alerts.
BR-17	Account Managers have comprehensive access to configure system settings, manage users, and monitor system performance.
BR-18	Detected events must be logged with a timestamp, event type, and associated camera.
BR-19	User configurations should be saved and applied consistently across the system.
BR-20	The system must adhere to strict privacy and security standards to protect user information from unauthorized access or breach.
BR-21	Users can customize event viewing preferences, such as filtering events by type or date range.
BR-22	Downloaded snapshots and recordings may be used as evidence, for further analysis, or for sharing with relevant parties.
BR-23	Downloaded snapshots and recordings are subject to applicable data protection and privacy regulations.
BR-24	Deleted events are permanently removed from the system and cannot be recovered.
BR-25	Deletion actions may be audited for security and compliance purposes.
BR-26	System metrics and storage information may be archived or logged for historical analysis or compliance purposes.
BR-27	Viewing system metrics may require sufficient system resources and network connectivity to ensure accurate and up-to-date information.
BR-28	Restarting the application should be performed with caution to avoid unnecessary disruption to surveillance operations.
BR-29	The application should log restart events for auditing and troubleshooting purposes.
BR-30	Light and dark mode changes should apply universally across the application interface to maintain visual coherence.
BR-31	Light and dark mode changes triggered by automatic mode should occur seamlessly and without disrupting user interactions with the application.

2. User Requirements

2.1 Overview

2.1.1 System Actors

Table 13. System Actors

#	Actor	Description
1	Admin	An Admin is a user who has complete access to all features and functionalities of the application. The Admin is responsible for critical tasks such as user management, including viewing the list of users, creating new users, updating user details, and deleting users. Admins can also manage cameras and camera groups, configure system settings, view system metrics, and access detailed live and recorded camera feeds. They can enable or disable detection, recording, and snapshot functionalities for cameras, manage various types of masks (motion, object), and zones, fine-tune motion detection settings, debug system issues, and manage events and exports. Additionally, Admins have the ability to change application themes and restart the application when necessary. This role is essential for maintaining the security, functionality, and efficiency of the system.
2	Member	A Member is a user with limited access, primarily focused on viewing capabilities within the application. Members can log in and log out, view the live dashboard, and access the system in full screen mode. They are allowed to view detailed live camera feeds and system metrics but do not have the administrative privileges to modify user details, camera settings, or system configurations. Members can change the application theme according to their preference. This role is designed to provide necessary access to view and monitor the system without the ability to alter any settings or configurations, ensuring that the integrity and security of the system remain intact.

2.1.2 Use Case Diagrams

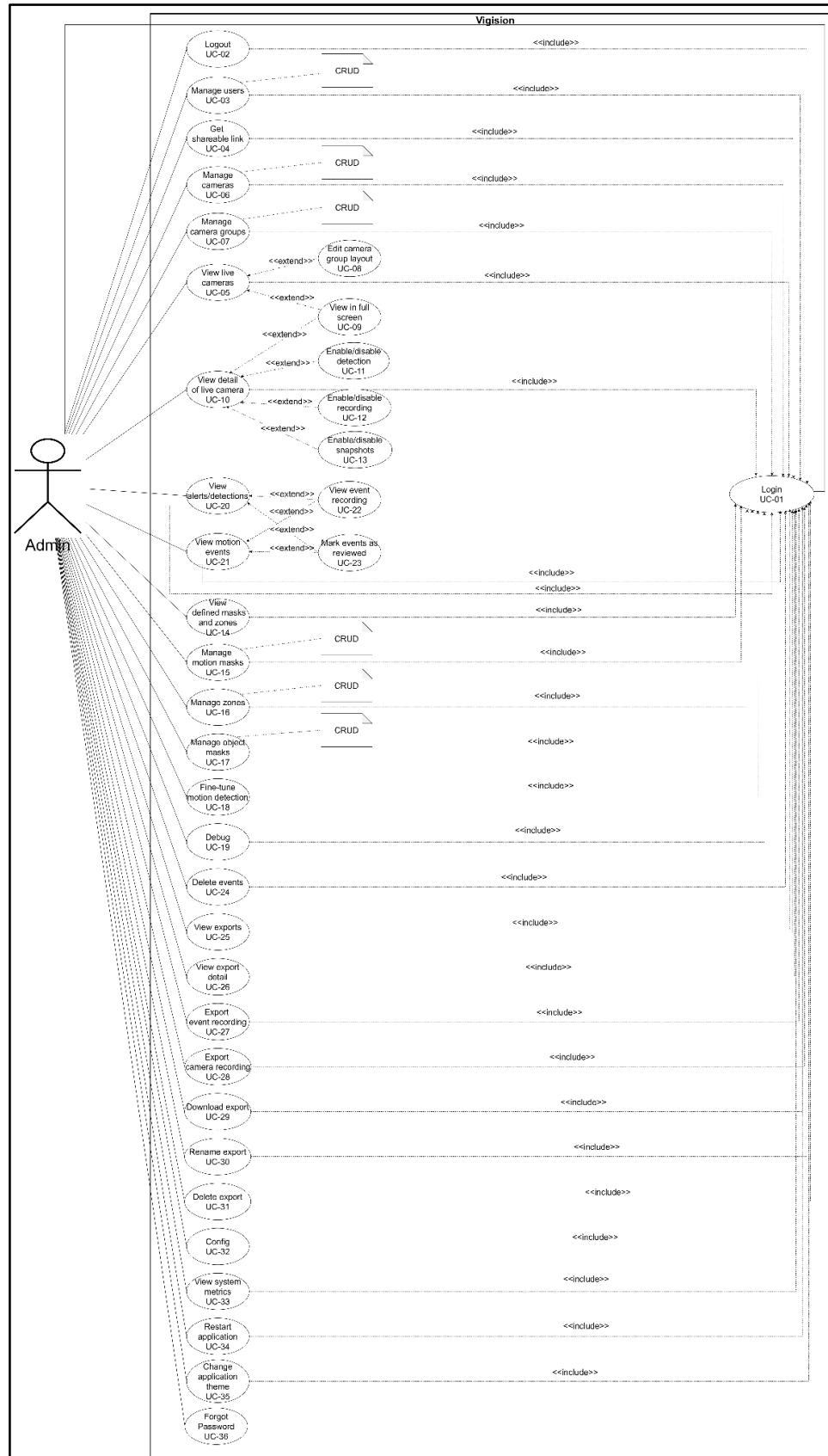


Figure 34. Use Case Diagram for Admin

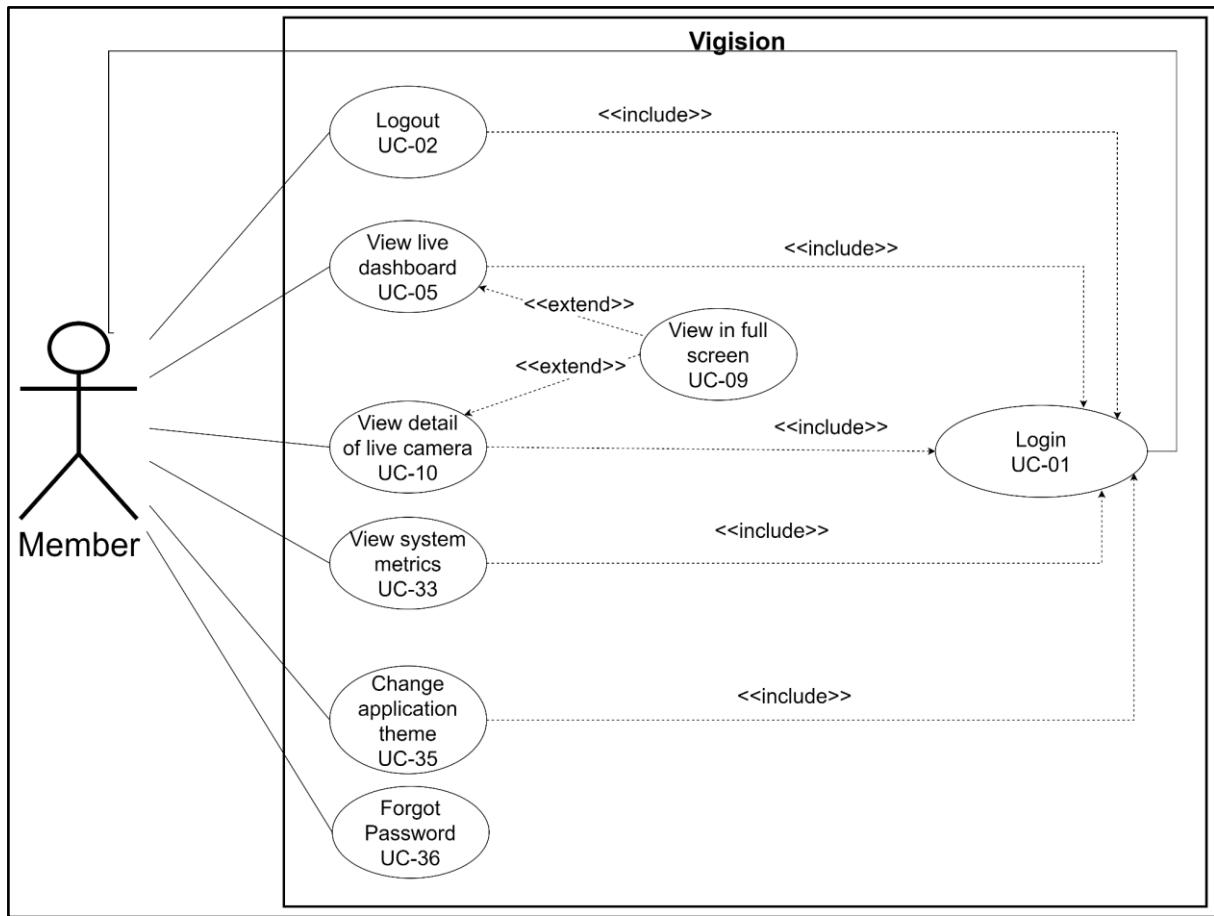


Figure 35. Use Case Diagram for Member

2.1.3 Use Case List

Table 14. Use case list

ID	Use Case	Actors	Secondary Actors
1	Login	Admin, Member	None
2	Logout	Admin, Member	None
3	Manage users	Admin	None
4	Get shareable link	Admin	None
5	View live cameras	Admin, Member	None
6	Manage cameras	Admin	None
7	Manage camera groups	Admin	None
8	Edit camera group layout	Admin	None

9	View in full screen	Admin, Member	None
10	View detail of live camera	Admin, Member	None
11	Enable/disable detection	Admin	None
12	Enable/disable recording	Admin	None
13	Enable/disable snapshots	Admin	None
14	View defined masks and zones	Admin	None
15	Manage motion masks	Admin	None
16	Manage zones	Admin	None
17	Manage object masks	Admin	None
18	Fine-tune motion detection	Admin	None
19	Debug	Admin	None
20	View alerts/detections	Admin	None
21	View motion events	Admin	None
22	View event recording	Admin	None
23	Mark events as reviewed	Admin	None
24	Delete events	Admin	None
25	View exports	Admin	None
26	View export detail	Admin	None
27	Export event recording	Admin	None
28	Export camera recording	Admin	None
29	Download export	Admin	None
30	Rename export	Admin	None
31	Delete export	Admin	None

32	Config	Admin	None
33	View system metrics	Admin, Member	None
34	Restart application	Admin	None
35	Change application theme	Admin, Member	None
36	Forgot Password	Admin, Member	None

2.2. Authentication

2.2.1 Login

Table 15. Login

ID and Name:	UC-01: Login		
Created By:	Nguyen Le Quang Thinh	Date Created:	25/05/2024
Primary Actor:	Admin, Member	Secondary Actors:	-
Trigger:	The user click login after they fill username or email and password information		
Description:	The user can access to the features and functionalities available to them based on their role and permissions. This process ensures that only authorized users can access the system.		
Pre-conditions:	1. The user has registered their account in the Vigision system. 2. The user has a device with internet access to reach the Vigision application interface. 3. The Vigision application and its authentication service are running and accessible.		
Post-conditions:	1. The user is successfully authenticated and logged into the system. 2. The user can access the features and functionalities according to their role.		
Normal Flow:	1. The user navigates to the login page of the Vigision application. 2. The user enters their username and password and clicks the 'Login' button. 3. If the credentials are valid, the system authenticates the user. 4. The user is redirected to the application's main dashboard.		
Alternative Flows:	1. The user navigates to the sign-in page of the Vigision application. 2. The user enters their invalid username and password and clicks the 'Sign In' button. 3. User will got username or email or password error message.		

Exception:	lost connection from network
Priority:	Medium
Frequency of Use:	High
Business Rules:	BR-01, BR-02, BR-20
Other Information:	-
Assumption:	-

Screen layout:

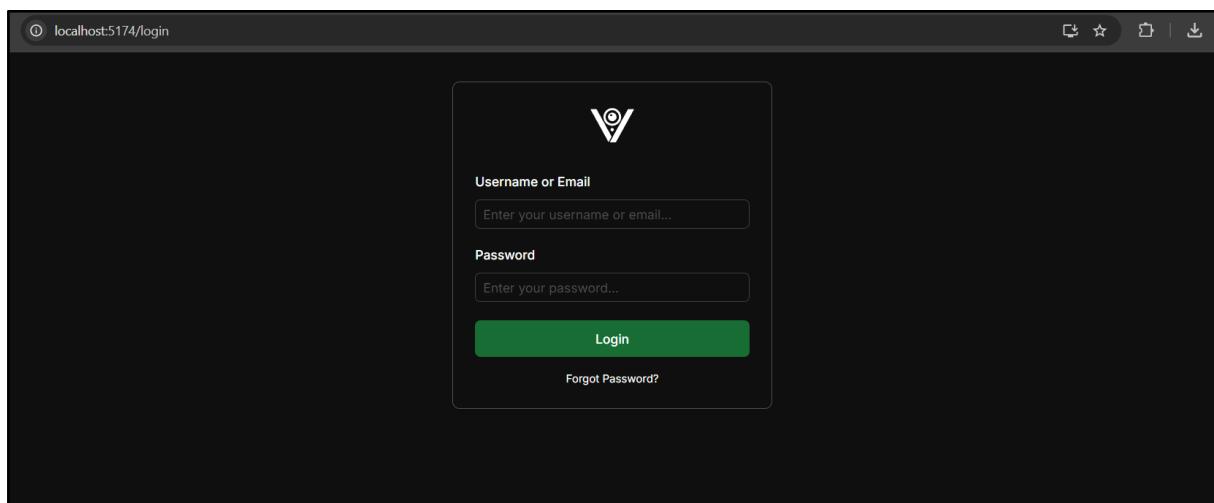


Figure 36. Login

2.2.2 Logout

Table 16. Logout

ID and Name:	UC-02: Logout		
Created By:	Nguyen Le Quang Thinh	Date Created:	25/05/2024
Primary Actor:	Admin, Member	Secondary Actors:	-
Trigger:	The user click logout from the Vigision application.		
Description:	The logout use case allows an Admin or User to safely exit the Vigision system, ensuring that their token is properly terminated and that no unauthorized access can occur through their token.		

Pre-conditions:	1. The user must be logged into the Vigision system.
Post-conditions:	1. The user's token is securely terminated. 2. The user is redirected to the login page.
Normal Flow:	1. The user clicks on the "Account" button located at the bottom left corner of the interface. 2. A menu with account options is displayed. 3. The user clicks on the "Logout" option. 4. The system terminates the user's token. 5. The user is redirected to the login page.
Alternative Flows:	-
Exception:	-
Priority:	Medium
Frequency of Use:	High
Business Rules:	BR-20
Other Information:	-
Assumption:	-

Screen layout:

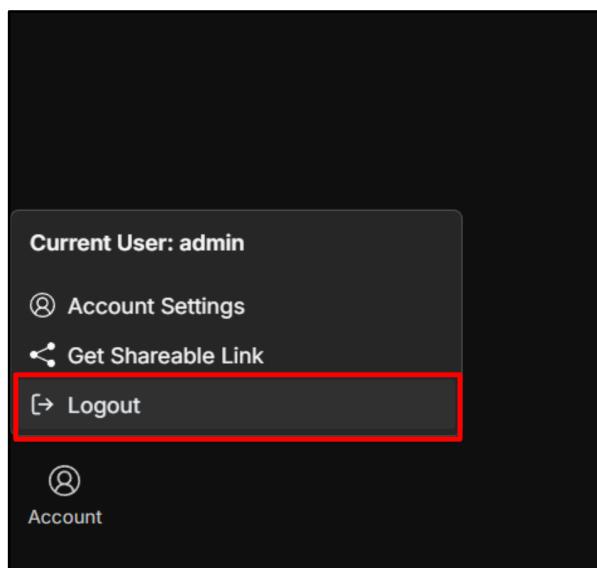


Figure 37. Logout

2.2.3 Forgot password

Table 17. Forgot password

ID and Name:	UC-36: Forgot password		
Created By:	Nguyen Le Quang Thinh	Date Created:	25/05/2024
Primary Actor:	Admin, Member	Secondary Actors:	-
Trigger:	The user clicks the "Forgot Password" link on the login page after forgot their password.		
Description:	The reset password describes the process by which a user who has forgotten their password can securely reset it. The system guides the user through the process of verifying their identity and setting a new password, ensuring continued secure access to the Vigision application.		
Pre-conditions:	1. The user has an existing account in the Vigision system. 2. The user has access to the email address associated with their account. 3. The Vigision application and its email service are operational.		
Post-conditions:	1. The user successfully resets their password. 2. The user's new password is securely updated in the system. 3. The user can log in using the new password.		
Normal Flow:	1. The user clicks on the "Forgot Password" link on the sign-in page. 2. The system prompts the user to enter the email address associated with their account then click send OTP. 3. The user enter OTP and click submit. 4. The system directs the user to a secure page where they are prompted to enter and confirm a new password. Validate and Update Password: The system validates the new password (e.g., checking it meets security criteria) and updates it in the system. 6. The password has been successfully reset.		
Alternative Flows:	1. The user enters an invalid OTP 2. The system displays an error message invalid OTP		
Exception:	-		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	BR-20		
Other Information:	-		
Assumption:	-		

Screen layout:

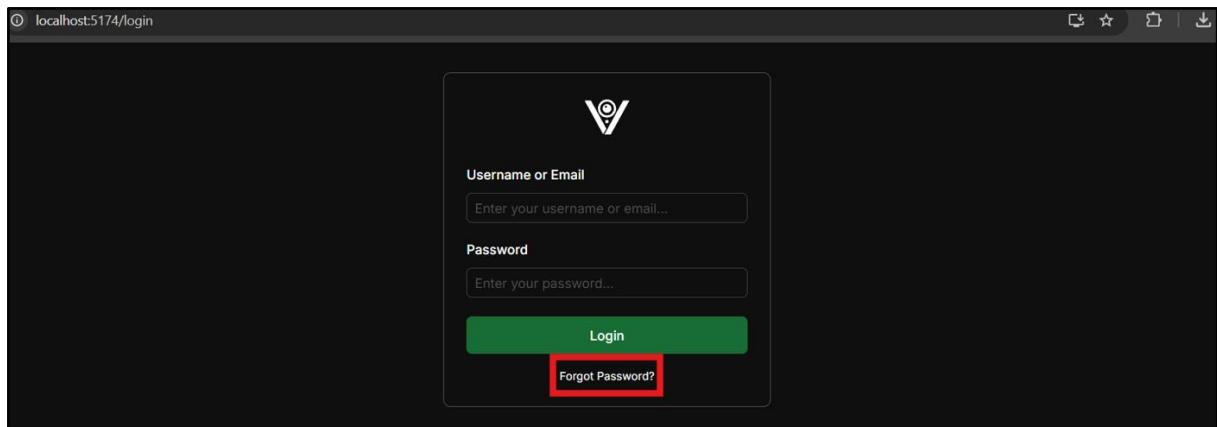


Figure 38. Forgot password - 1

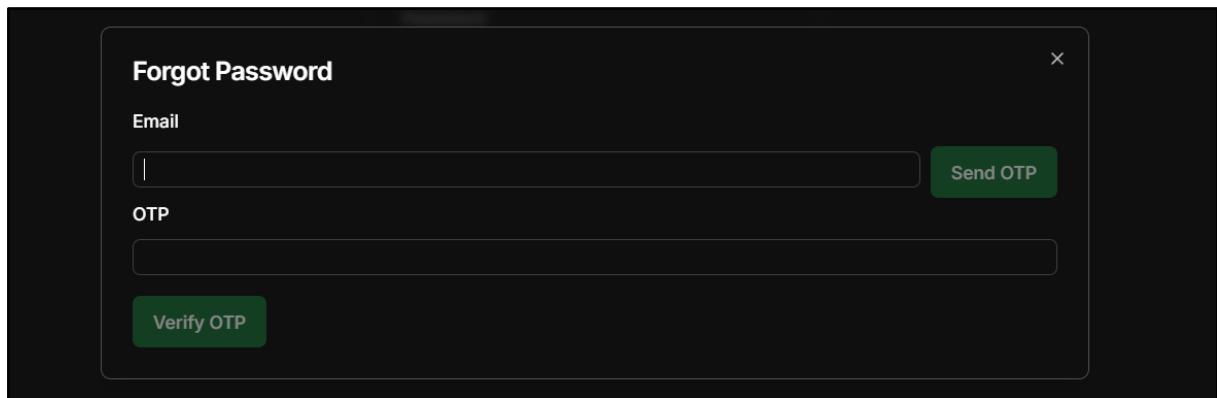


Figure 39. Forgot password - 2

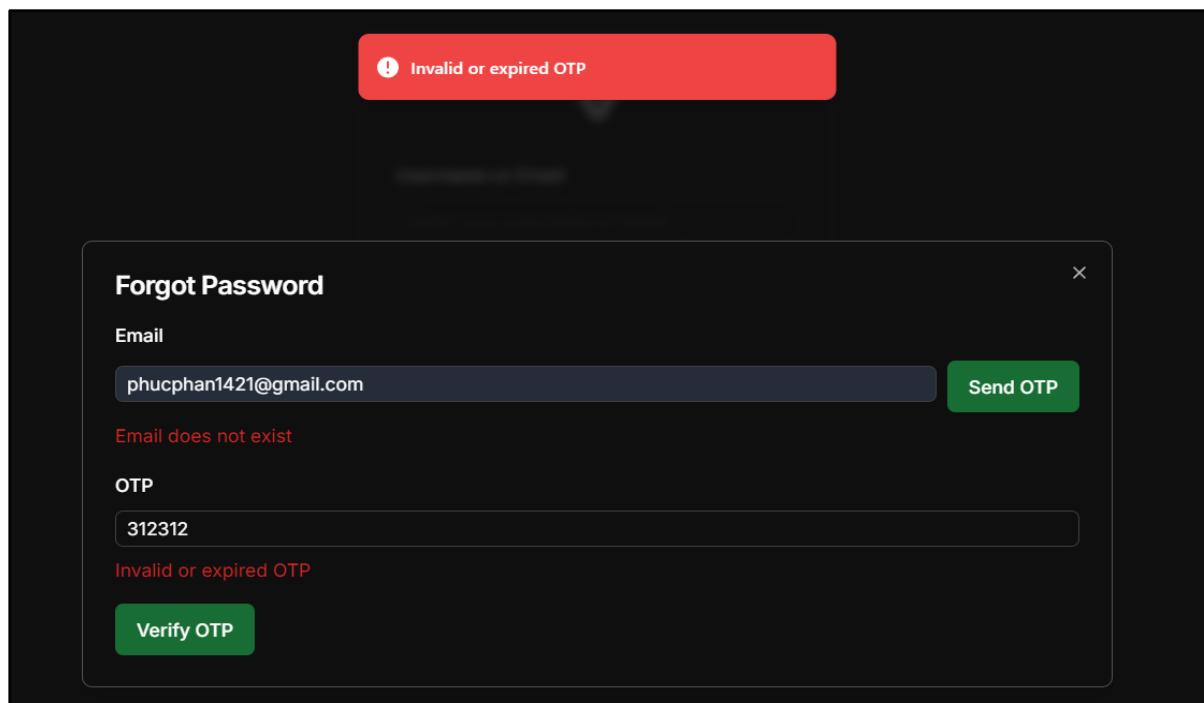


Figure 40. Forgot password - 3

2.3. User management

2.3.1 Manage users

Table 18. Manage users

ID and Name:	UC-03: Manage users		
Created By:	Nguyen Le Quang Thinh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin navigates to the Account settings page.		
Description:	<p>View Registered Users: This allows the Admin to view a comprehensive list of all registered users within the Vigision system. The list includes detailed information such as username, email, and receive alert status.</p> <p>Create New User: The Admin can create a new user account by entering information such as username, email, OTP, and password. The system validates the input, sends an OTP to the provided email, and finalizes the account creation upon OTP verification.</p> <p>Update Password: Admins or Members can update their passwords by verifying the old password and entering a new one that meets the system's security requirements.</p> <p>Delete User: Admins can delete existing user accounts, removing the user's information from the database and ensuring they no longer have system access.</p> <p>Manage Email Alerts: Admins or Members can enable or disable email alerts. When enabled, users receive email notifications for significant events detected by the Vigision system.</p>		
Pre-conditions:	The Admin must be logged into the system and have the necessary permissions to manage users.		
Post-conditions:	<p>A list of users is displayed, allowing the Admin to take further actions, such as creating, editing or deleting users.</p> <p>A new user account is created, and the new user can log in to the Vigision system.</p> <p>The user's email address is updated.</p> <p>The user's password is updated.</p> <p>The user account is deleted, and the user can no longer log in.</p> <p>The user's email alert settings are updated.</p>		
Normal Flow:	<p>UC-03-01: View list of users</p> <ol style="list-style-type: none"> 1. The Admin clicks on the "Account" button and selects "Account Settings" or navigates through "Settings" -> "Settings". 2. The Admin clicks on the "Account Settings" tab. 3. The system retrieves the list of all users from the database. 4. The system displays the list of users, including details such as username, email, receive alert status. <p>UC-03-02: Create user</p> <ol style="list-style-type: none"> 1. The Admin clicks on the "+ Add user" button. 		

	<p>2. The system displays a form for entering user information (username, email, otp and password).</p> <p>3. The Admin fills in the user information.</p> <p>4. The Admin clicks the "Send OTP" button.</p> <p>5. The system sends a OTP to the provided email address.</p> <p>6. The Admin enters the OTP received in the email.</p> <p>7. The Admin clicks the "Create User" button.</p> <p>UC-03-03: Update email</p> <ol style="list-style-type: none"> 1. The Admin/Member clicks on the "Update Email" button for a specific user. 2. The system displays a form for entering the new email address. 3. The Admin/Member enters the new email address. 4. The Admin/Member clicks the "Send OTP" button. 5. The system sends an OTP to the new email address. 6. The Admin/Member enters the OTP received in the email. 7. The Admin/Member clicks the "Update Email" button. 8. The system validates the OTP and the new email address. 9. The Admin/Member click update email 10. The system updates the user's email address and displays a success message. <p>UC-03-04: Update password</p> <ol style="list-style-type: none"> 1. The Admin/Member clicks on the "Update Password" button . 2. The system displays a form for entering the old password . 3. The Admin/Member enters the old password. 4. The Admin/Member click submit. 5. The system verify the old password 6. The system displays a form for entering the new password and confirm password 7. The Admin/Member enters the new password and confirm password. 8. The system validates the old password and checks the new password against the security requirements. 9. The Admin/Member clicks the "Save" button. 10.The system updates the user's password and displays a success message. <p>UC-03-05: Delete user</p> <ol style="list-style-type: none"> 1. The Admin clicks on the "Delete" button for a specific user. 2. The system prompts the Admin to confirm the deletion. 3. The Admin confirms the deletion. 4. The system deletes the user's information from the database. 5. The system displays a success message indicating that the user has been deleted. <p>UC-03-06: Enable/Disable Receive Alert by Email</p> <ol style="list-style-type: none"> 1. The Admin/Member clicks on the "Receive alert" toggle button. 2. The system updates the user's email alert settings. 3. The system displays a success message indicating that the email alert settings have been updated.
Alternative Flows:	<p>UC-03-02:</p> <p>Step 7: If the username already exists, the system displays an error message indicating that the username is already taken. The Admin</p>

	<p>enters a different username and proceeds with the user creation.</p> <p>Step 7: If the email already exists, the system displays an error message indicating that the email is already in use.</p> <p>The Admin enters a different email and proceeds with the user creation.</p> <p>Step 7: If the OTP is incorrect, the system displays an error message indicating that the OTP is incorrect. The Admin enters the correct OTP and proceeds with the user creation.</p> <p>UC-03-03:</p> <p>Step 9: If the new email already exists, the system displays an error message indicating that the email is already in use. The Admin/Member enters a different email and proceeds with the update.</p> <p>Step 9: If the OTP is incorrect, the system displays an error message indicating that the OTP is incorrect. The Admin/Member enters the correct OTP and proceeds with the update.</p> <p>UC-03-04:</p> <p>Step 4: If the old password is incorrect, the system displays an error message indicating that the old password is incorrect. The Admin/User enters the correct old password and proceeds with the update.</p>
Exception:	<p>UC-03-02:</p> <p>System error during the user creation process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue creating the user.</p> <p>UC-03-03:</p> <p>System error during the email update process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue updating the email.</p> <p>UC-03-04:</p> <p>System error during the password update process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue updating the password.</p> <p>UC-03-05:</p> <p>System error during the user deletion process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue deleting the user.</p> <p>UC-03-06:</p> <p>System error during the email alert update process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue updating the email alert settings.</p>
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-03, BR-05

Other Information:	-
Assumption:	-

Screen layout:

The screenshot shows a dark-themed user management interface. On the left is a vertical sidebar with icons for Live, Review, Export, Settings, and Account. The main area is titled 'Users' and lists three entries:

- admin No email provided
- newname phucphan1421@gmail.com
- newname_2 phucphce171166@fpt.edu.vn

Each entry has a row of buttons at the end: 'Update Password', 'Update Email', 'Receive alert: Off' (set to 'Off'), and 'Delete'. In the top right corner, there is a '+ Add User' button.

Figure 41. Manage users - View list of users

This screenshot is identical to Figure 41, but it features a red rectangular box highlighting the '+ Add User' button located in the top right corner of the header area.

Figure 42. Manage users - Create user

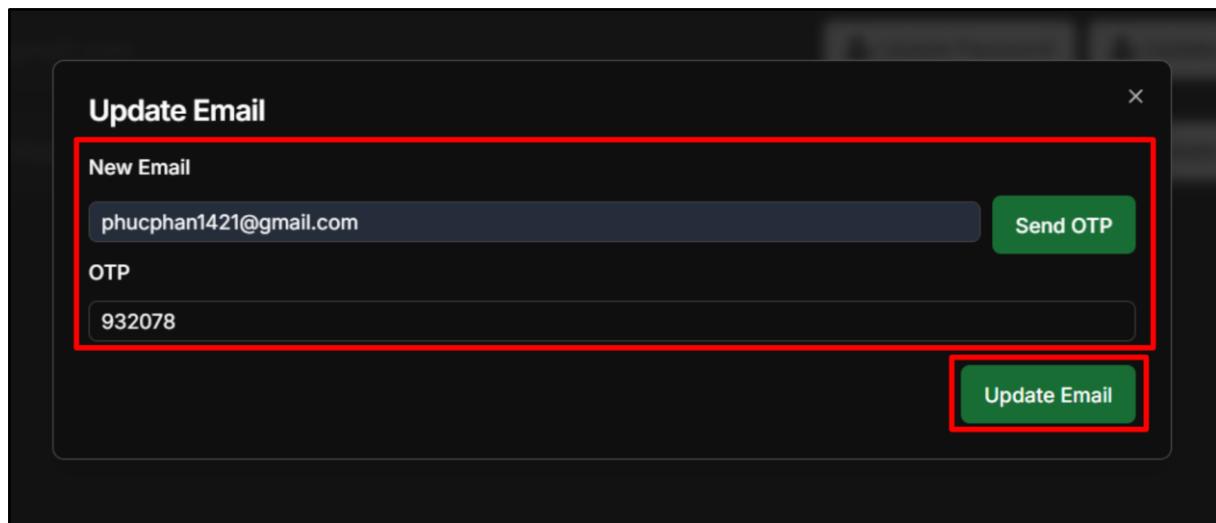


Figure 43. Manage users - Update user email

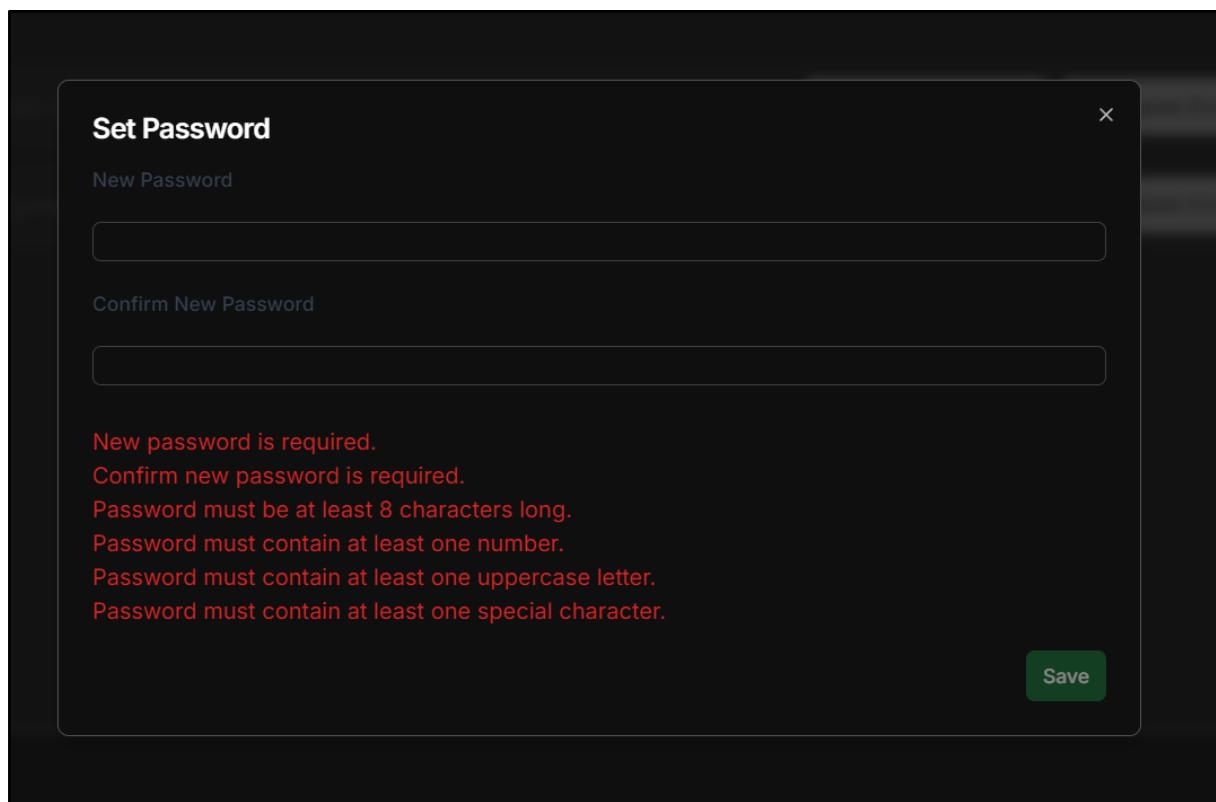


Figure 44. Manage users - Update password

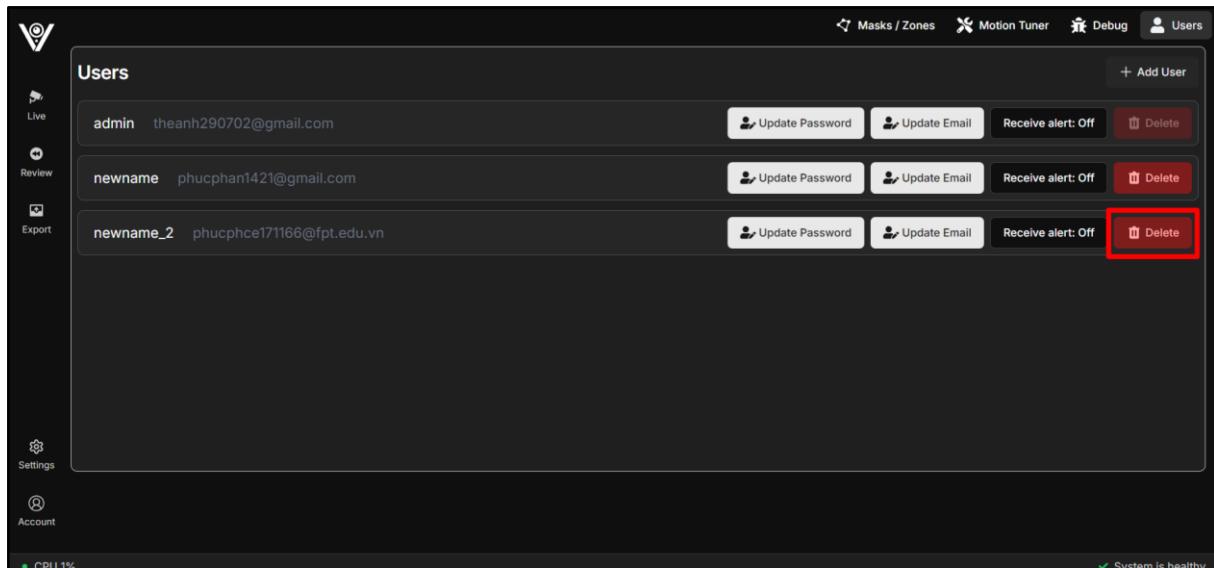


Figure 45. Manage users - Delete user

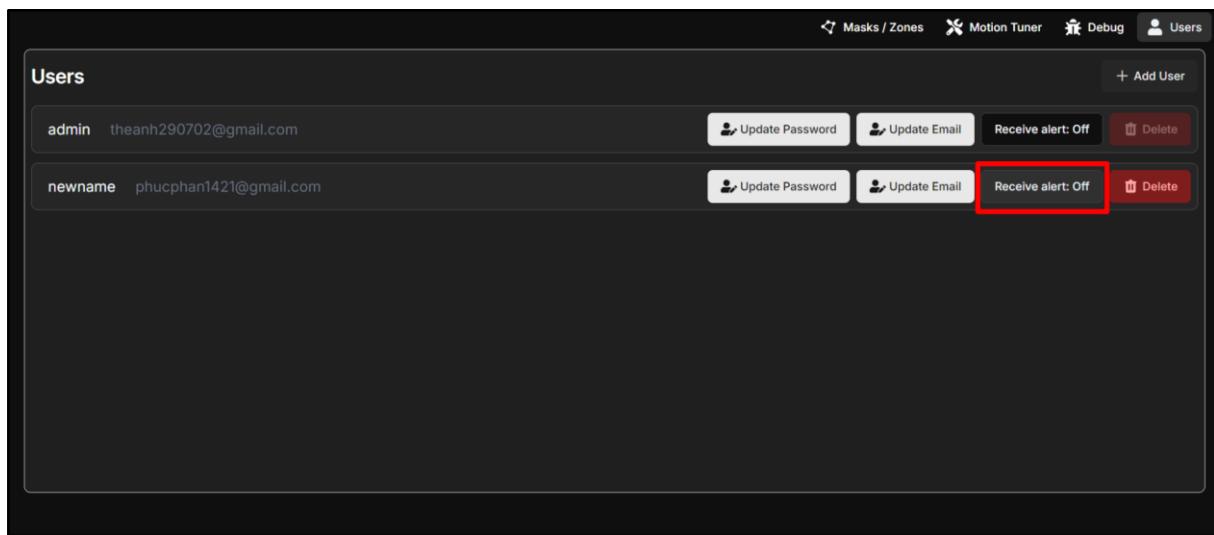


Figure 46. Manage users - Toggle receive alert

2.4. Cameras & Surveillance

2.4.1 View live cameras

Table 19. View live cameras

ID and Name:	UC-05: View live cameras		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin/Member	Secondary Actors:	-
Trigger:	User visits the Vigision web application.		
Description:	View Live Streams allows Admin or Member to view real-time video feeds from all cameras connected to the Vigision system. Upon accessing the		

	Vigision web application, users are presented with a grid layout displaying live previews of each camera's stream. These previews are essentially thumbnails that offer a glimpse into the ongoing activity captured by the cameras. If camera is offline or not properly connected to the system, the application will display a placeholder image to inform the user of the camera's unavailability.
Pre-conditions:	<ol style="list-style-type: none"> 1. The Admin/Member has successfully logged in to the Vigision application (for domain users). 2. The cameras are connected to the NVR and configured in the system. 3. The cameras are powered on and broadcasting video streams.
Post-conditions:	<ol style="list-style-type: none"> 1. The Admin/Member sees a live preview of each connected camera on the home page. 2. The previews are displayed in a grid layout, with each camera thumbnail representing a live stream.
Normal Flow:	<ol style="list-style-type: none"> 1. Admin/Member opens the Vigision application and lands on the default page. 2. The system retrieves live video streams from all connected cameras. 3. The system displays a preview of each camera's live stream on the web page.
Alternative Flows:	<ol style="list-style-type: none"> 1. If a camera is offline or not connected, the system displays a placeholder image or a notification indicating the camera is unavailable. 2. If no cameras are connected or configured, the system displays a message indicating that no cameras are available for viewing.
Exception:	-
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-02, BR-06, BR-09
Other Information:	-
Assumption:	-

Screen layout:

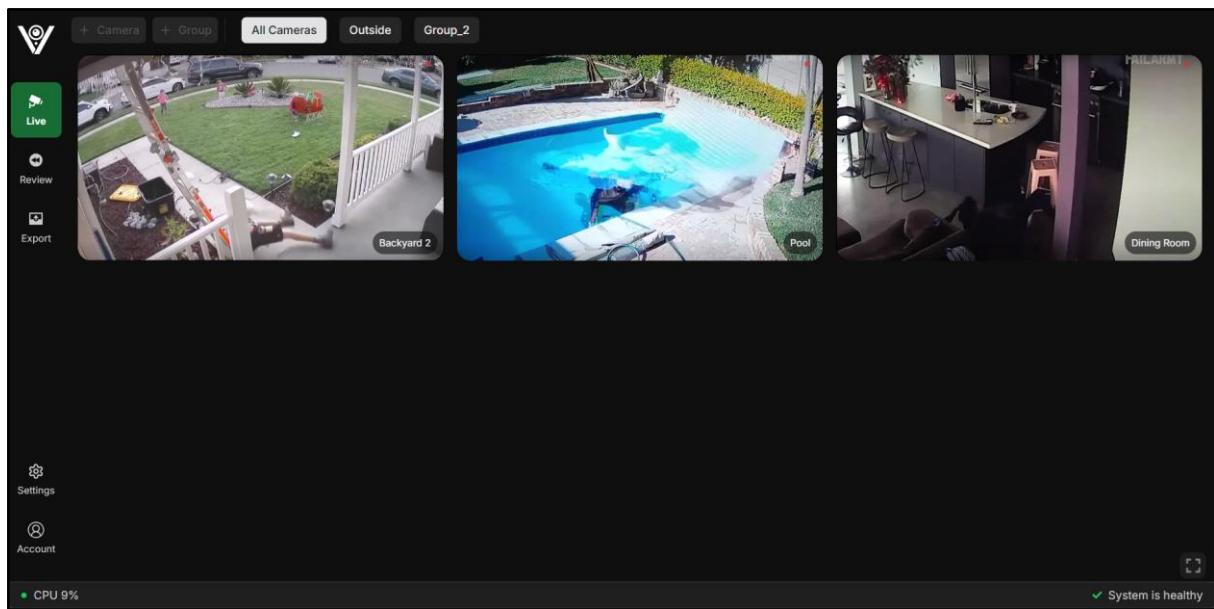


Figure 47. View live cameras

2.4.2 Manage cameras

Table 20. Manage cameras

ID and Name:	UC-06: Manage cameras		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin/Member	Secondary Actors:	-
Trigger:	One of the following triggers: User visits the Vigision web application. The Admin click "+" button. The Admin pen icon to update an existing camera. The Admin click trash can icon		
Description:	<ul style="list-style-type: none"> ● This use case allows the Admin to view a comprehensive list of all cameras configured within the Vigision system. The list includes details such as camera ID, name, status, and configuration settings. ● This use case allows the Admin to add a new camera to the Vigision system. The Admin provides the necessary configuration details for the camera, such as ID, name, and IP address. Once added, the new camera becomes part of the surveillance system and appears in the live dashboard and camera management list. ● This use case allows the Admin to update the configuration details of an existing camera. The Admin can modify settings such as the camera's name, IP address, and detection parameters. Updating camera settings is essential for maintaining accurate and effective surveillance. ● This use case allows the Admin to delete an existing camera from the 		

	Vigision system. Deleting a camera removes its configuration from the database and ensures that it is no longer part of the surveillance system.
Pre-conditions:	<ol style="list-style-type: none"> 1. The Admin/Member has successfully logged in to the Vigision application (for domain users). 2. The cameras are connected to the NVR and configured in the system. 3. The cameras are powered on and broadcasting video streams.
Post-conditions:	<p>UC-06-01: The Admin is able to view and review the list of all configured cameras.</p> <p>UC-06-02: A new camera is added to the system and is available for monitoring and management.</p> <p>UC-06-03: The selected camera's configuration is updated and saved in the system.</p> <p>UC-06-04: The selected camera is deleted from the system and no longer appears in the live dashboard or camera management list.</p>
Normal Flow:	<p>UC-06-01: View list of camera</p> <ol style="list-style-type: none"> 1. The Admin clicks on the "+Camera" option. 2. The system retrieves the list of all cameras from the database. 3. The system displays the list of cameras, including details such as camera ID, name, status, and configuration settings. 4. The Admin reviews the list of cameras and selects individual cameras for further actions if needed. <p>UC-06-02: Create camera</p> <ol style="list-style-type: none"> 1. The Admin clicks on the "+" button. 2. The system displays a form for entering the new camera's configuration details (e.g., camera ID, name, IP address). 3. The Admin fills in the required details and submits the form. 4. The system validates the input and saves the new camera's configuration to the database. 5. The system displays a success message indicating that the new camera has been added. 6. The new camera appears in the list of cameras and is available on the live dashboard. <p>UC-06-03: Update camera</p> <ol style="list-style-type: none"> 1. The Admin clicks on the pen icon for a specific camera in the list. 2. The system displays a form pre-filled with the current configuration details of the selected camera. 3. The Admin modifies the desired configuration details. 4. The Admin submits the form. 5. The system validates the input and updates the camera's configuration in the database. 6. The system displays a success message indicating that the camera's configuration has been updated. 7. The updated camera configuration is reflected in the live

	<p>dashboard and camera management list.</p> <p>UC-06-04: Delete camera</p> <ol style="list-style-type: none"> 1. The Admin clicks on the trash can icon button for a specific camera in the list. 2. The system prompts the Admin to confirm the deletion. 3. The Admin confirms the deletion. 4. The system deletes the camera's configuration from the database. 5. The system displays a success message indicating that the camera has been deleted. 6. The deleted camera is removed from the live dashboard and camera management list.
Alternative Flows:	<p>UC-06-02:</p> <p>Step 3: If the camera ID already exists, the system displays an error message indicating that the camera ID is already in use. The Admin enters a different camera ID and resubmits the form.</p>
Exception:	<p>UC-06-02:</p> <p>System error during the camera creation process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue adding the new camera and logs the error.</p> <p>UC-06-03:</p> <p>System error during the camera update process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue updating the camera and logs the error.</p> <p>UC-06-04:</p> <p>System error during the camera delete process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue deleting the camera and logs the error.</p>
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-06, BR-07, BR-09, BR-13, BR-19
Other Information:	-
Assumption:	-

Screen layout:

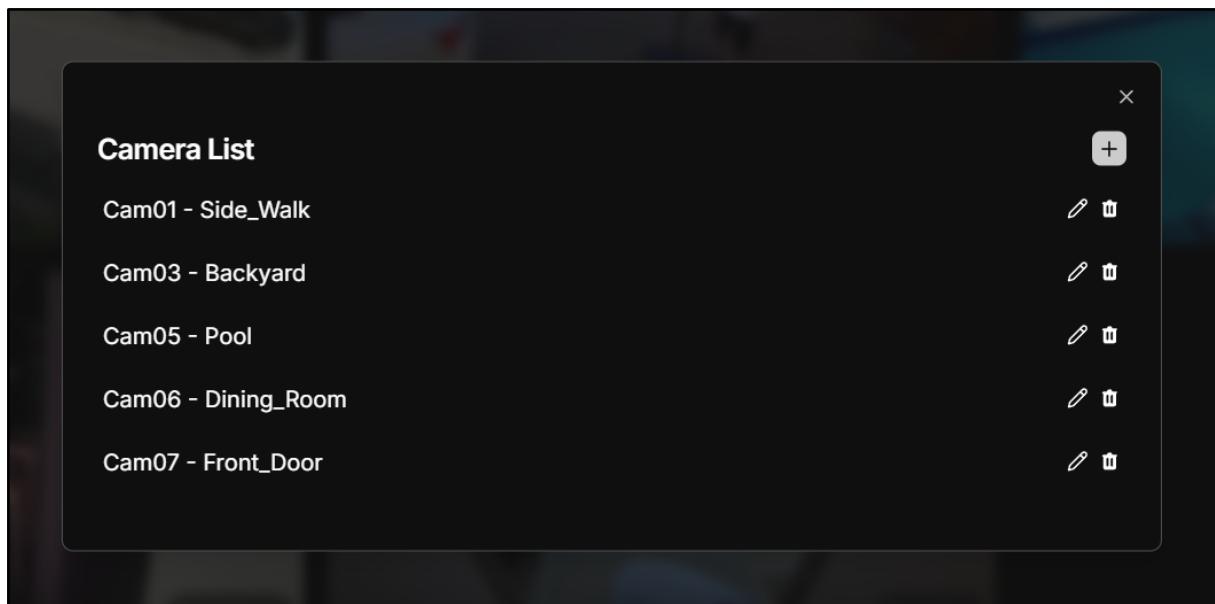


Figure 48. Manage cameras - View list of camera

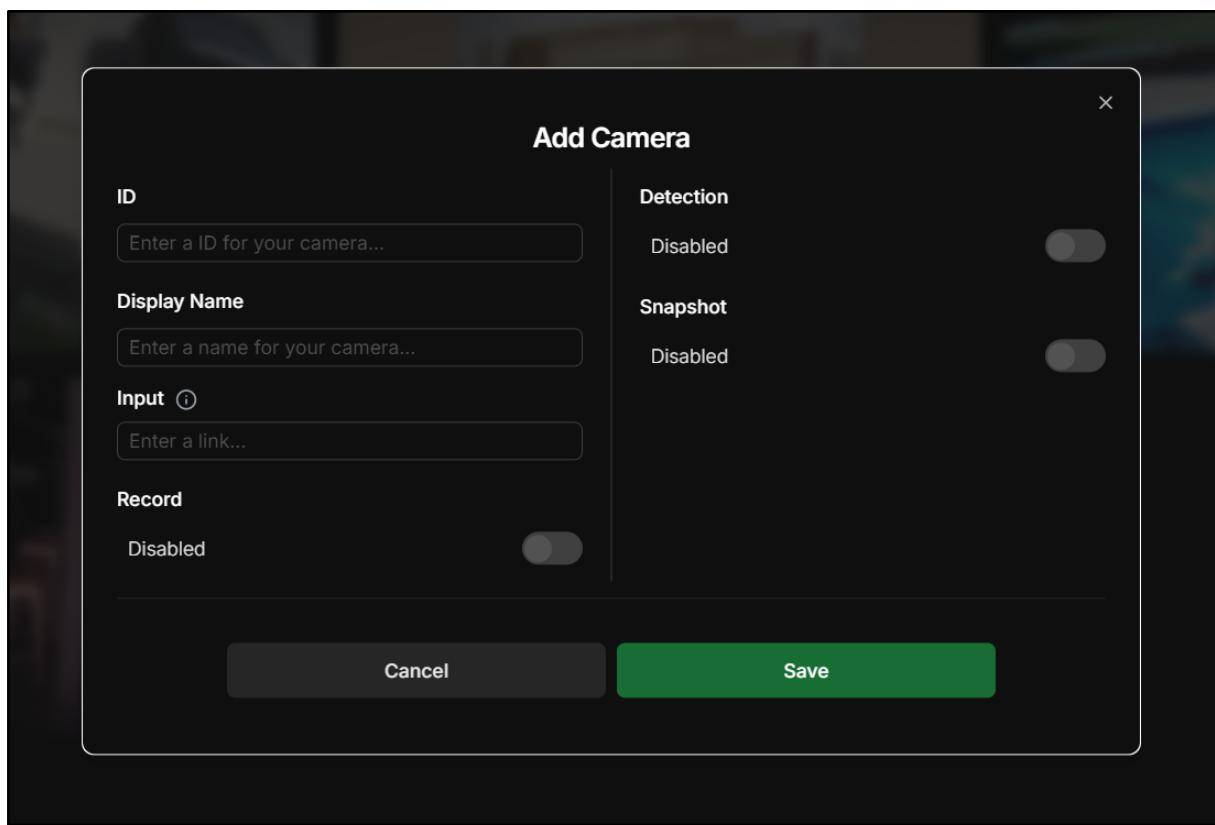


Figure 49. Manage cameras - Create camera

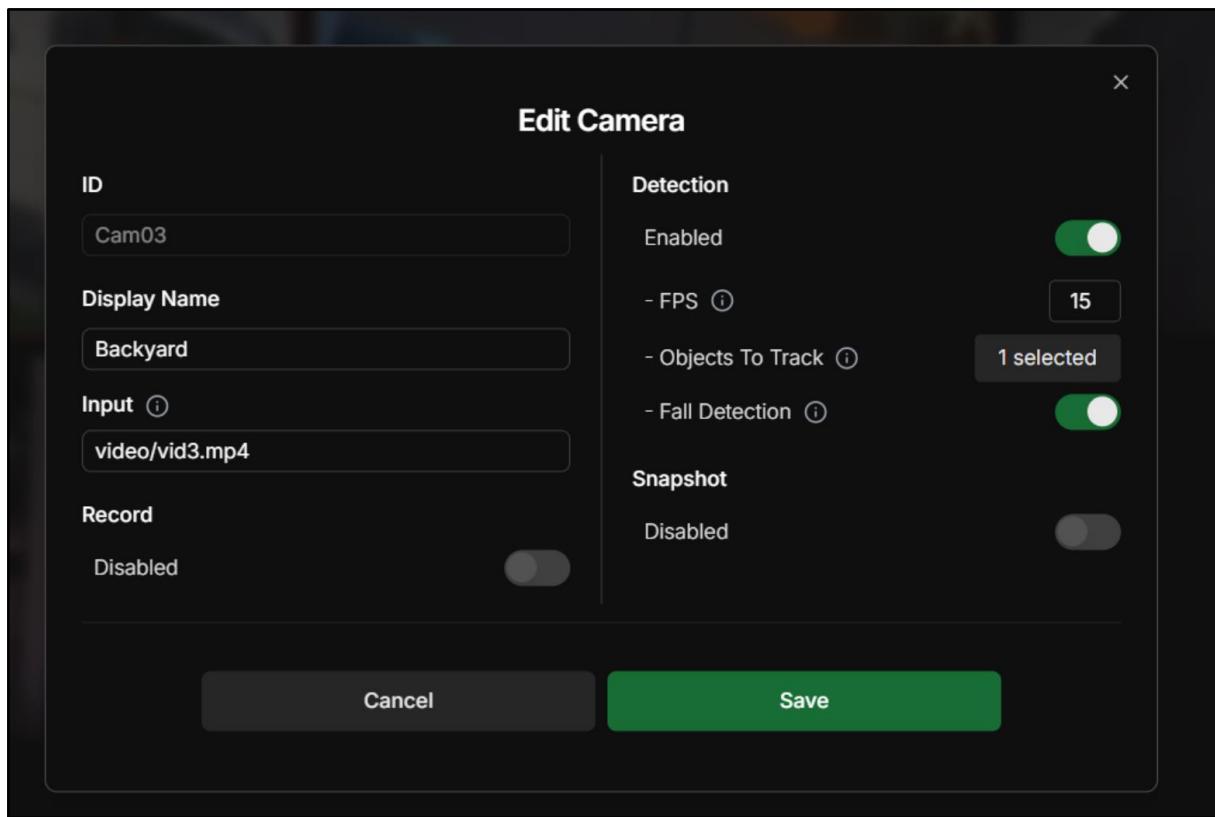


Figure 50. Manage cameras - Update camera

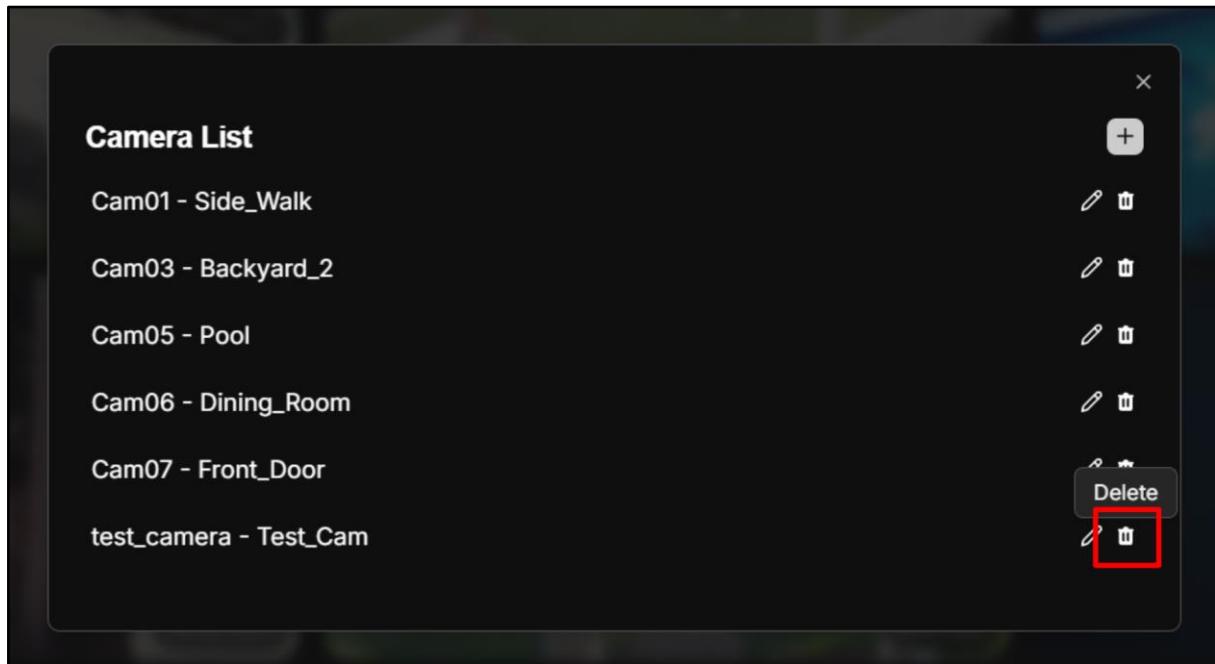


Figure 51. Manage cameras - Delete camera

2.4.3 Manage camera groups

Table 21. Manage camera groups

ID and Name:	UC-07: Manage camera groups		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	UC-07-01: View list of camera group The Admin click "+ Group" button UC-07-02: Create camera group The Admin pen icon to update an existing camera group. UC-07-03: Update camera group The Admin click update pen icon UC-07-04 Delete camera group The Admin click trash can icon		
Description:	UC-07-01: View list of camera group This use case allows the Admin to view a comprehensive list of all camera groups configured within the Vigision system. The list includes details such as group ID, name, and the cameras included in each group. UC-07-02: Create camera group This use case allows the Admin to create a new camera group within the Vigision system. The Admin provides the necessary configuration details for the group, such as group ID, name, and the cameras to be included in the group. Once added, the new camera group becomes part of the surveillance system and appears in the live dashboard and camera group management list. UC-07-03: Update camera group This use case allows the Admin to update the configuration details of an existing camera group. The Admin can modify settings such as the group's name and the cameras included in the group. Updating camera group settings is essential for maintaining organized and efficient surveillance. UC-07-04 Delete camera group This use case allows the Admin to delete an existing camera group from the Vigision system. Deleting a group removes its configuration from the database and ensures that it is no longer part of the surveillance system.		
Pre-conditions:	The Admin must be logged into the system		
Post-conditions:	UC-07-01: View list of camera group The Admin is able to view and review the list of all configured camera groups. UC-07-02: Create camera group A new camera group is added to the system and is available for		

	<p>monitoring and management.</p> <p>UC-07-03: Update camera group The selected camera's group configuration is updated and saved in the system.</p> <p>UC-07-04 Delete camera group The selected camera group is deleted from the system and no longer appears in the live dashboard or camera group management list.</p>
Normal Flow:	<p>UC-07-01: View list of camera group</p> <ol style="list-style-type: none"> 1. The Admin clicks on the "+Group" button. 2. The system retrieves the list of all camera groups from the database. 3. The system displays the list of camera groups, including details such as group ID, name, and the cameras included in each group. 4. The Admin reviews the list of camera groups and selects individual groups for further actions if needed. <p>UC-07-02: Create camera group</p> <ol style="list-style-type: none"> 1. The Admin clicks on the "+" button. 2. The system displays a form for entering the new camera group's configuration details (e.g., group ID, name, cameras to include). 3. The Admin fills in the required details and submits the form. 4. The system validates the input and saves the new camera's configuration to the database. 5. The system displays a success message indicating that the new camera group has been added. 6. The new camera group appears in the list of cameras group and is available on the live dashboard. <p>UC-07-03: Update camera group</p> <ol style="list-style-type: none"> 1. The Admin clicks on the pen icon for a specific camera group in the list. 2. The system displays a form pre-filled with the current configuration details of the selected camera group. 3. The Admin modifies the desired configuration details. 4. The Admin submits the form. 5. The system validates the input and updates the camera's group configuration in the database. 6. The system displays a success message indicating that the camera's group configuration has been updated. 7. The updated camera group configuration is reflected in the live dashboard and camera group management list. <p>UC-07-04 Delete camera group</p> <ol style="list-style-type: none"> 1. The Admin clicks on the trash can icon button for a specific camera group in the list. 2. The system prompts the Admin to confirm the deletion. 3. The Admin confirms the deletion. 4. The system deletes the camera's group configuration from the database. 5. The system displays a success message indicating that the camera group has been deleted. 6. The deleted camera is removed from the live dashboard and

	camera group management list.
Alternative Flows:	<p>UC-07-01: View list of camera group Step 3: If the group ID already exists, the system displays an error message indicating that the group ID is already in use. The Admin enters a different group ID and resubmits the form.</p> <p>UC-07-02: Create camera group</p> <p>UC-07-03: Update camera group</p> <p>UC-07-04 Delete camera group</p>
Exception:	<p>UC-07-01: View list of camera group System error during the camera group creation process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue adding the new camera group and logs the error.</p> <p>UC-07-03: Update camera group System error during the camera group update process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue updating the camera group and logs the error.</p> <p>UC-07-04 Delete camera group System error during the camera group deletion process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue delete the camera group and logs the error.</p>
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-06, BR-07, BR-09, BR-13, BR-19
Other Information:	-
Assumption:	-

Screen layout:

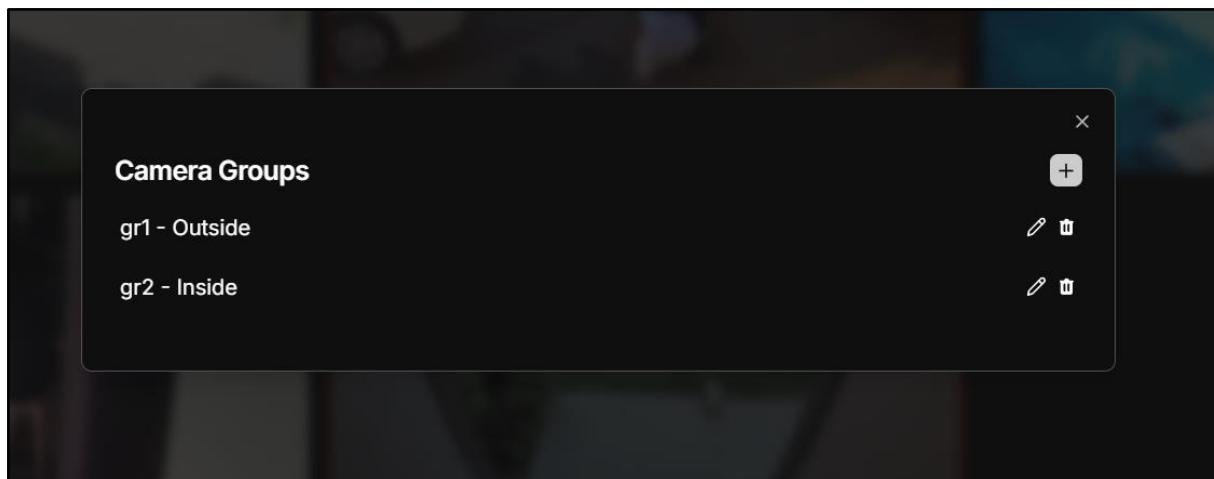


Figure 52. Manage cameras group - View list of camera group

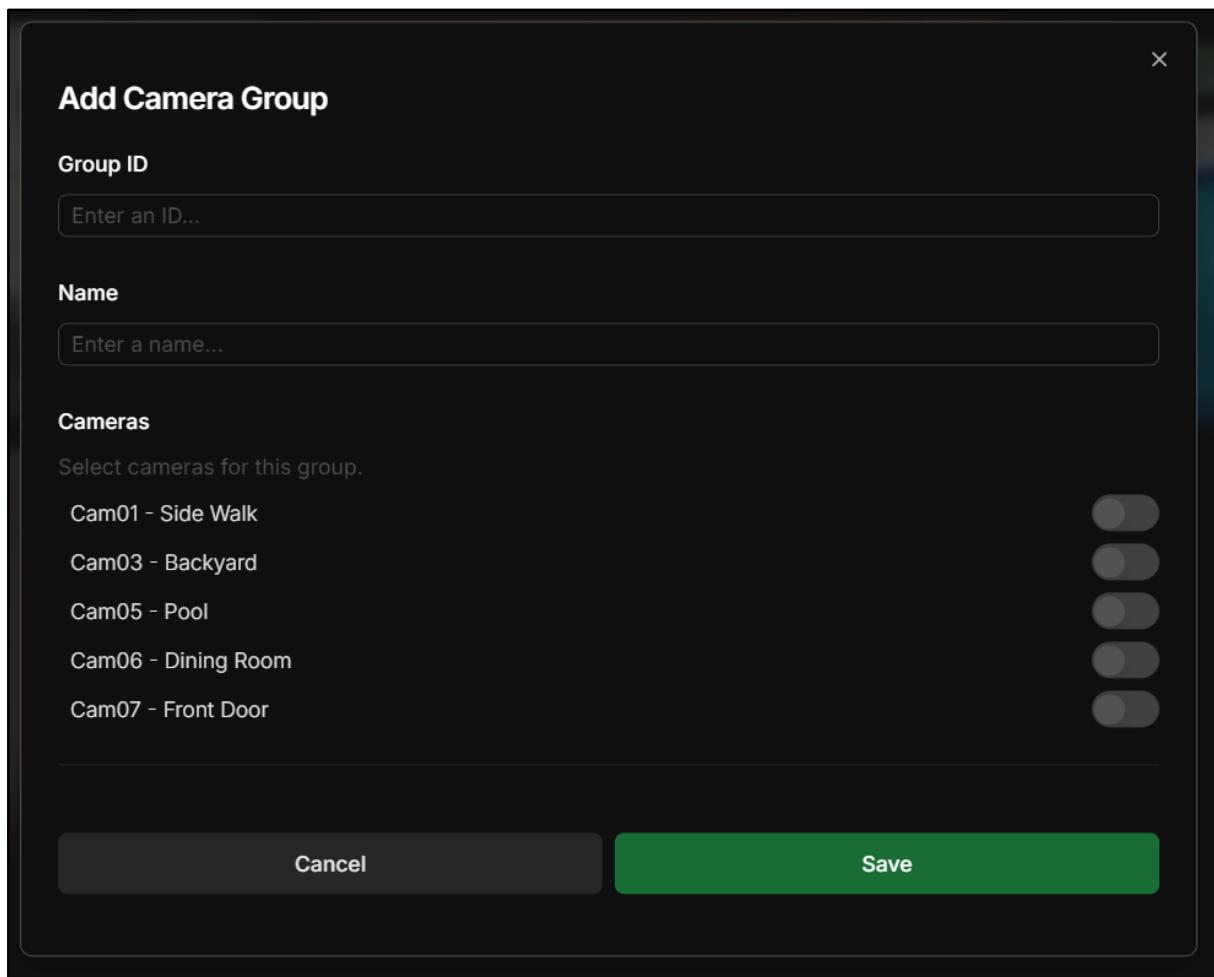


Figure 53. Manage cameras group - Create camera group

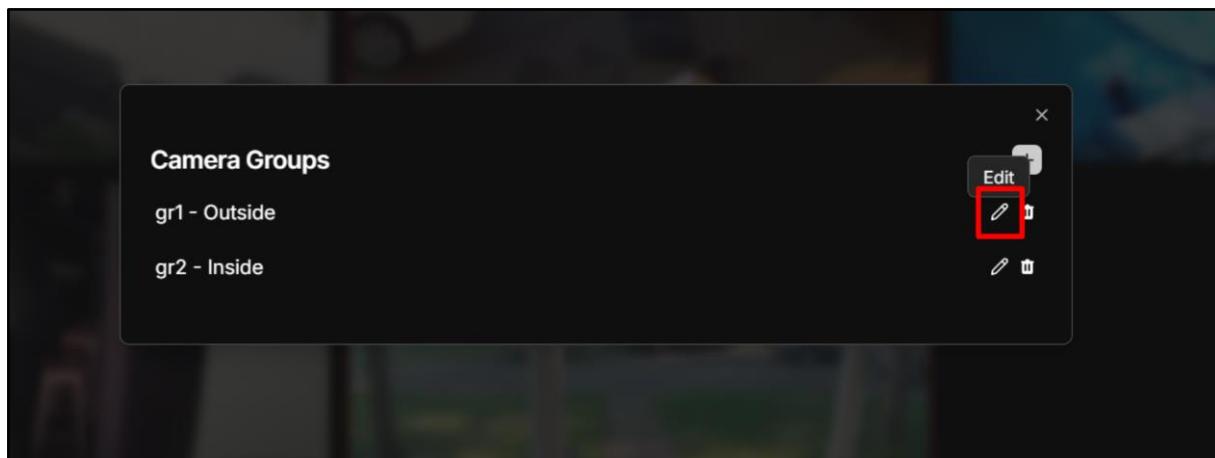


Figure 54. Manage cameras group - Update camera group

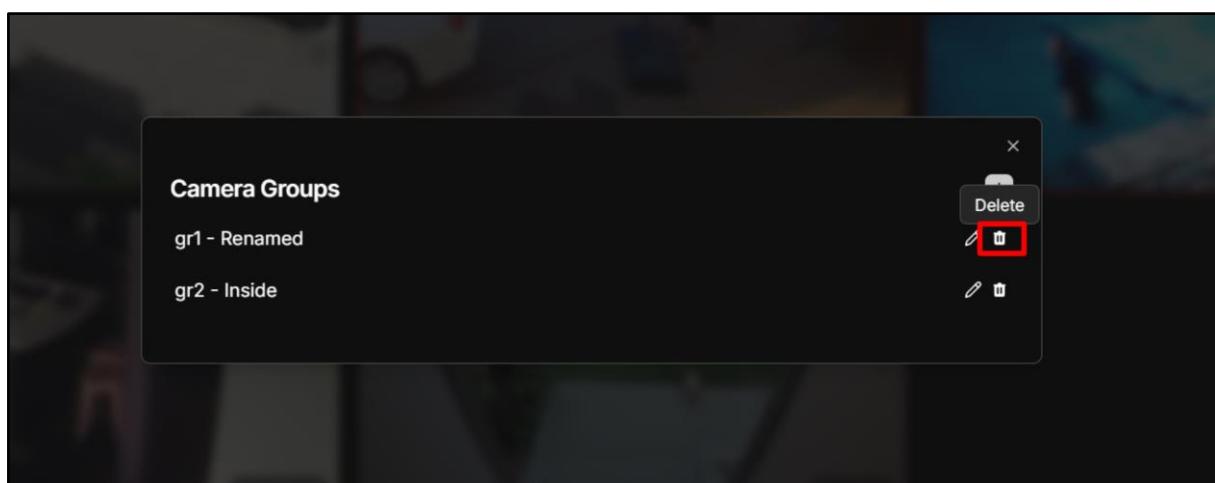


Figure 55. Manage cameras group - Delete camera group

2.4.4 Edit camera group layout

Table 22. Edit camera group layout

ID and Name:	UC-08: Edit camera group layout		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to edit the layout of a camera group from the live dashboard or camera group management section.		
Description:	This use case allows the Admin to modify the layout of cameras within a group. The Admin can rearrange the camera feeds to create a customized view that meets their monitoring needs. Editing the camera group layout helps improve the efficiency and effectiveness of surveillance.		

Pre-conditions:	The Admin must be logged into the system
Post-conditions:	The camera group layout is updated and saved in the system.
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the camera group layout editing section. 2. The system displays the current layout of the selected camera group. 3. The Admin drags and drops the camera feeds to rearrange the layout. 4. The Admin clicks the "Save" button to apply the new layout. 5. The system updates the camera group layout in the database. 6. The system displays a success message indicating that the layout has been updated. 7. The new layout is reflected in the live dashboard and camera group management list.
Alternative Flows:	-
Exception:	System error during the layout update process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue updating the layout and logs the error.
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-06, BR-19
Other Information:	-
Assumption:	-

Screen layout:

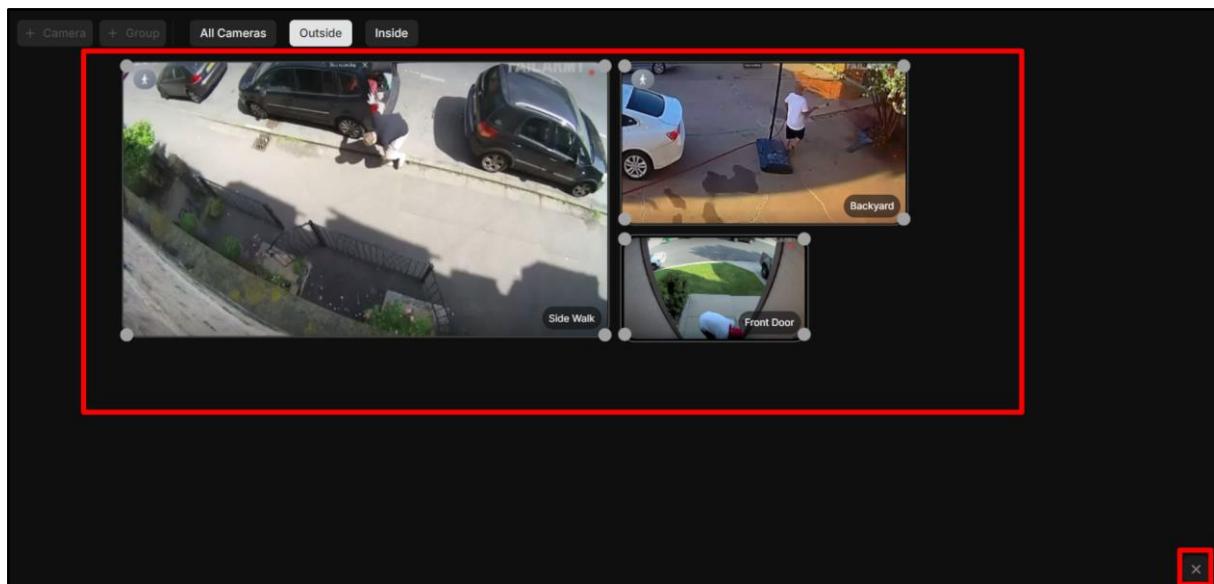


Figure 56. Edit camera group layout

2.4.5 View in full screen

Table 23. View in full screen

ID and Name:	UC-09: View in full screen		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin, Member	Secondary Actors:	-
Trigger:	Admin/Member clicks on the "Full Screen" icon while viewing a camera stream.		
Description:	This use case allows Admin/Member to maximize their viewing experience by expanding the live video stream from a selected camera to fill the entire screen. This full-screen mode provides a more immersive view, allowing users to focus on details and monitor activity more effectively.		
Pre-conditions:	1. The Admin/Member has accessed the Vigision application. 2. The camera is connected to the NVR and configured in the system. 3. The Admin/Member is currently viewing the live stream of the selected camera.		
Post-conditions:	The live video stream from the selected camera is displayed in full-screen mode.		
Normal Flow:	1. Admin/Member clicks on the "Full Screen" button or icon while viewing a camera stream. 2. The system expands the video stream to fill the entire screen. 3. The Admin/Member can now interact with the video stream in full-screen mode.		

Alternative Flows:	
Exception:	-
Priority:	Low
Frequency of Use:	Low
Business Rules:	BR-01, BR-02, BR-06
Other Information:	-
Assumption:	-

Screen layout

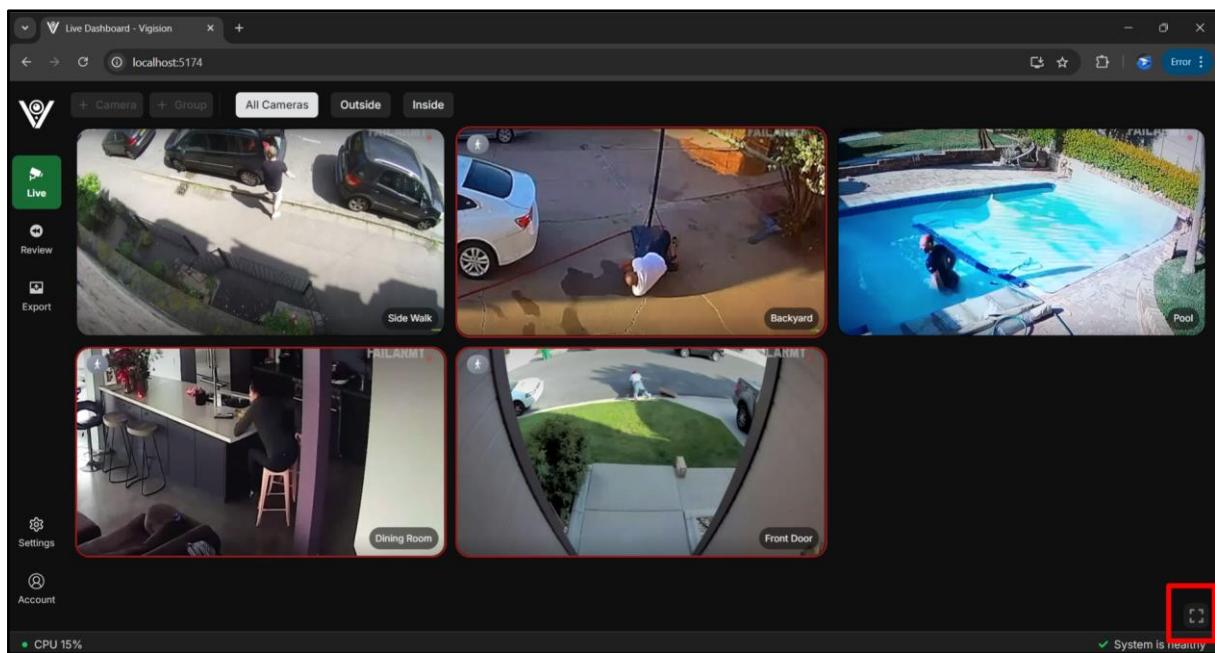


Figure 57. View full screen of camera stream

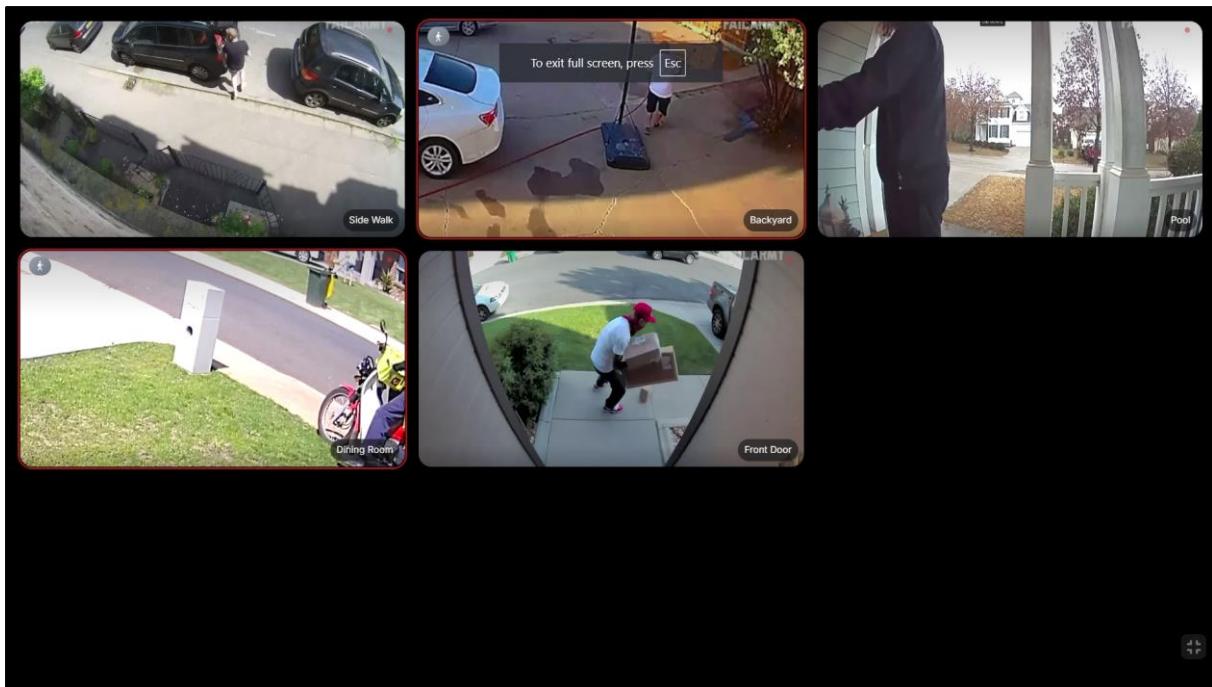


Figure 58. View full screen of camera stream

2.4.6 View detail of live camera

Table 24. View detail of live camera

ID and Name:	UC-10: View detail of live camera		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin/Member	Secondary Actors:	-
Trigger:	Admin/Member clicks on a specific camera thumbnail		
Description:	View Live Camera Details allows Admin/Member to access a dedicated page displaying comprehensive information about a specific camera. The information displayed on this page includes the camera's name, location, current status (online/offline), and other pertinent details that provide users with a deeper understanding of the camera's configuration and operational status. Additionally, Admin/Member can interact with the information presented, such as modifying camera settings or accessing historical data related to the camera's performance.		
Pre-conditions:	1. The Admin/Member has accessed the Vigision application. 2. The camera is connected to the NVR and configured in the system.		
Post-conditions:	1. The Admin/Member is viewing the detailed information page for the selected camera. 2. The page displays various details about the camera.		

Normal Flow:	<ol style="list-style-type: none"> 1. Admin/Member clicks on a specific camera thumbnail or selects a camera from the list. 2. The system retrieves the detailed information for the selected camera. 3. The system displays the detailed information page for the selected camera. 4. The page displays various details about the camera, including its name, model, location, status, and other relevant information. 5. The Admin/Member can interact with the information displayed on the page, such as toggling settings or viewing historical data.
Alternative Flows:	<ol style="list-style-type: none"> 1. If the selected camera is offline or not connected, the system displays an error message indicating the camera is unavailable.
Exception:	-
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-02, BR-06, BR-09
Other Information:	-
Assumption:	-

Screen layout

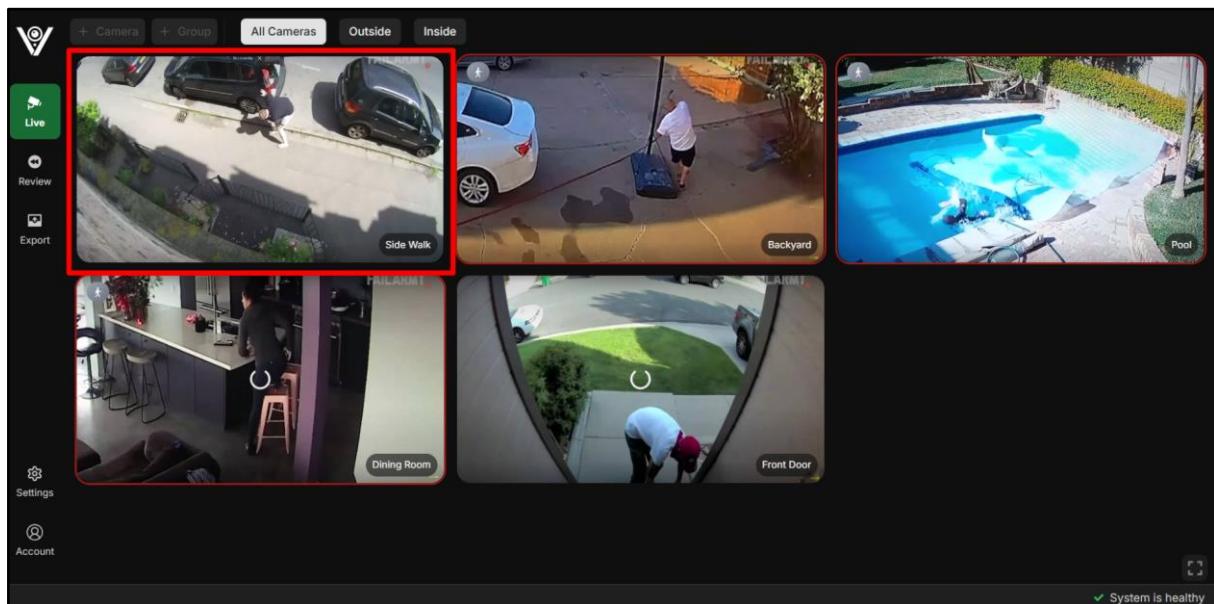


Figure 59. View live camera details



Figure 60. View live camera details

2.4.7 Enable/disable detection

Table 25. Enable/disable detection

ID and Name:	UC-11: Enable/disable detection		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin clicks on the "Detection" toggle button on the camera detail screen.		
Description:	By accessing the camera detail screen, Admin can easily toggle the detection mode on or off. When enabled, the camera actively monitors for motion and triggers alerts or recordings based on user-defined settings. Conversely, disabling detection stops the camera from actively monitoring for motion, preventing unnecessary alerts and recordings.		
Pre-conditions:	1. The Admin has accessed the Vigision application. 2. The Admin is viewing the camera detail screen for a specific camera. 3. The camera is connected to the NVR and configured in the system.		
Post-conditions:	1. The detection mode for the selected camera is updated. 2. The system reflects the updated detection status on the camera detail screen.		
Normal Flow:	1. Admin clicks on the "Detection" toggle button on the camera detail screen. 2. The system sends a command to enable or disable detection. 3. The camera receives the command and updates its detection status.		

Alternative Flows:	-
Exception:	-
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-06, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout



Figure 61. Enable detection



Figure 62. Disable detection

2.4.8 Enable/disable recording

Table 26. Enable/disable recording

ID and Name:	UC-12: Enable/disable recording		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin clicks on the "Recording" toggle button for a specific camera.		
Description:	Admin can enable or disable recording. When enabled, the camera continuously saves video footage, providing a comprehensive record of events. Disabling recording stops the camera from saving footage.		
Pre-conditions:	1. The Admin has accessed the Vigision application. 2. The camera is connected to the NVR and configured in the system. 3. Sufficient storage space is available on the NVR for recording.		
Post-conditions:	1. The recording status for the selected camera is updated. 2. The system reflects the updated recording status in the interface.		
Normal Flow:	1. User clicks on the "Recording" toggle button for a specific camera. 2. The system sends a command to enable or disable recording. 3. The camera updates its recording status. 4. The system updates the recording status in the interface.		
Alternative Flows:	If there is not enough storage space available on the NVR, the system may disable the recording toggle button		

Exception:	-
Priority:	High
Frequency of Use:	Medium
Business Rules:	BR-01, BR-06, BR-07, BR-13, BR-19
Other Information:	-
Assumption:	-

Screen layout



Figure 63. Enable recording

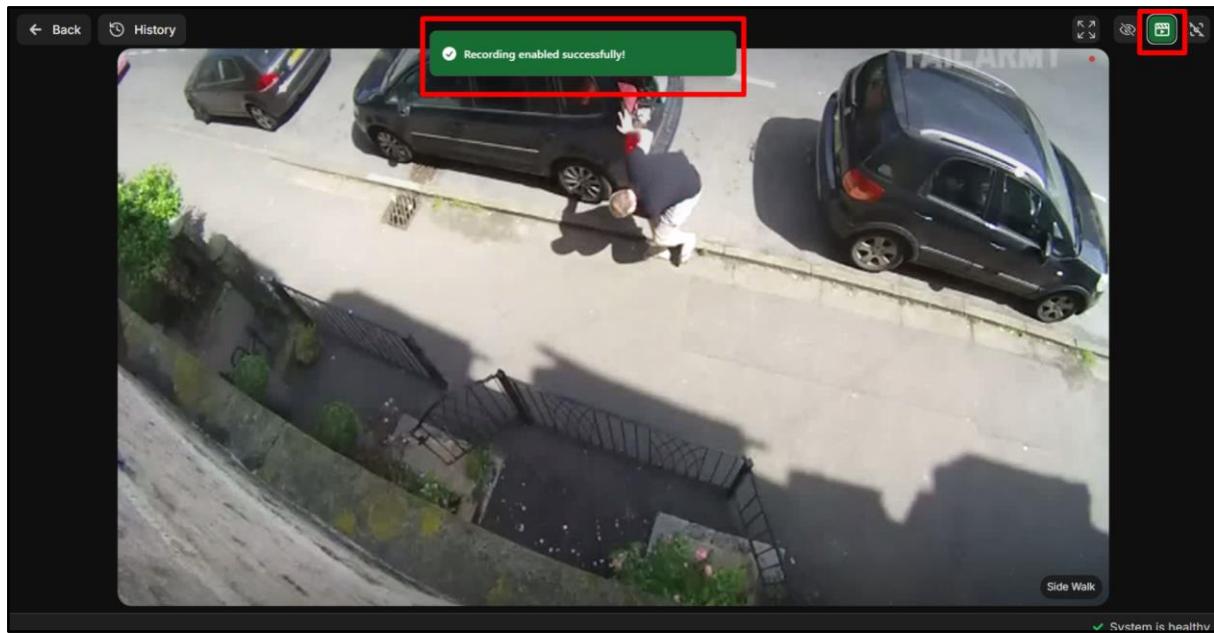


Figure 64. Disable recording

2.4.9 Enable/disable snapshots

Table 27. Enable/disable snapshots

ID and Name:	UC-13: Enable/disable snapshots		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin clicks on the "Snapshot" toggle button for a specific camera.		
Description:	This use case allows Admin to control the snapshot feature for a specific camera. Admin can enable or disable automatic snapshot capture when motion is detected. When enabled, the camera will take a still image whenever motion is detected, providing a visual record of the event. Disabling the snapshot feature stops the camera from taking these automatic pictures.		
Pre-conditions:	1. The Admin has accessed the Vigision application. 2. The camera is connected to the NVR and configured in the system. 3. The motion detection feature is enabled for the camera.		
Post-conditions:	1. The snapshot status for the Admin camera is updated. 2. The system reflects the updated snapshot status in the interface.		
Normal Flow:	1. Admin clicks on the "Snapshot" toggle button for a specific camera. 2. The system sends a command to enable or disable snapshot capture. 3. The camera receives the command and updates its snapshot status. 4. The system receives updates the snapshot status in the interface.		

Alternative Flows:	If motion detection is disabled for the camera, the system may disable the snapshot toggle button.
Exception:	-
Priority:	High
Frequency of Use:	Medium
Business Rules:	BR-01, BR-06, BR-09, BR-13, BR-19
Other Information:	-
Assumption:	-

Screen layout

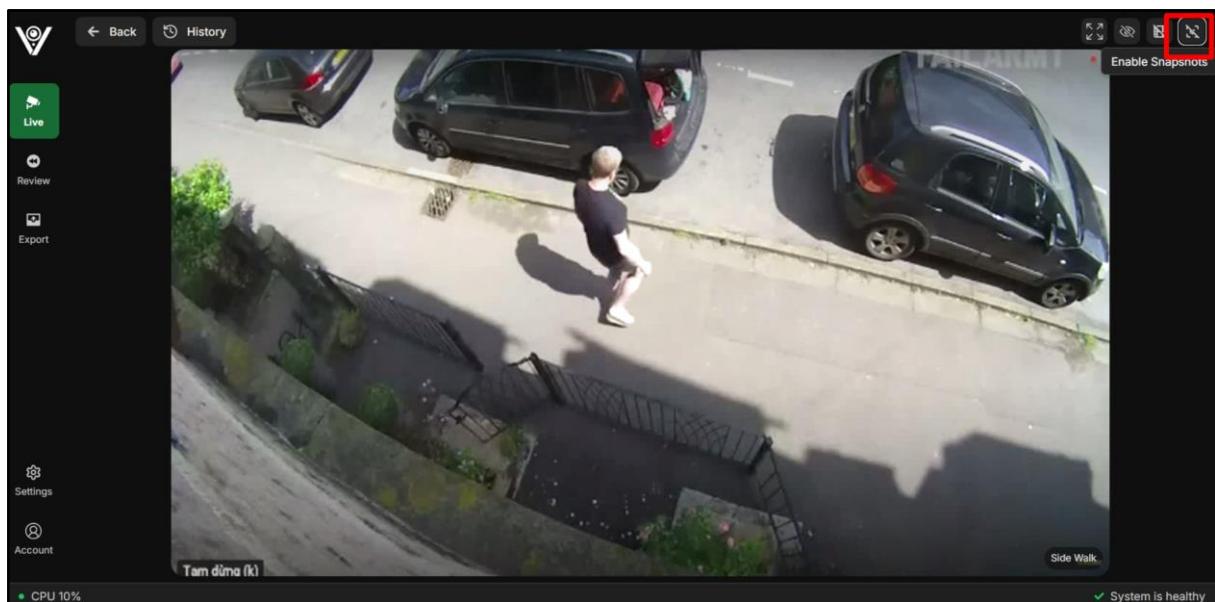


Figure 65. Enable snapshot



Figure 66. Disable snapshot

2.5. AI Feature Setting and Debugging

2.5.1 View defined masks and zones

Table 28. View defined masks and zones

ID and Name:	UC-14: View defined masks and zones		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin navigates to the "Masks and Zones" section or clicks a button/link to view them.		
Description:	The Admin can view all defined masks and zones within a selected camera stream. They can choose to view specific types of masks (motion masks, object masks) or view all types together.		
Pre-conditions:	1. Admin is logged in (for Domain user). 2. A camera stream is selected and displayed.		
Post-conditions:	1. The system displays a list of all defined masks and zones for the selected camera stream, categorized by type (motion, object) or showing all types together. 2. The Admin can view details about each mask or zone, such as its name, shape, and settings.		
Normal Flow:	1. Admin navigates to the "Masks and Zones" section or clicks a button/link to view them. 2. Admin selects a camera stream (default is the first camera). 3. The system displays a list of defined masks and zones for the selected camera stream.		

	4. Admin can filter the list by type (motion, object, or all). 5. Admin can click on a specific mask or zone to view its details.
Alternative Flows:	-
Exception:	-
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-06, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout:

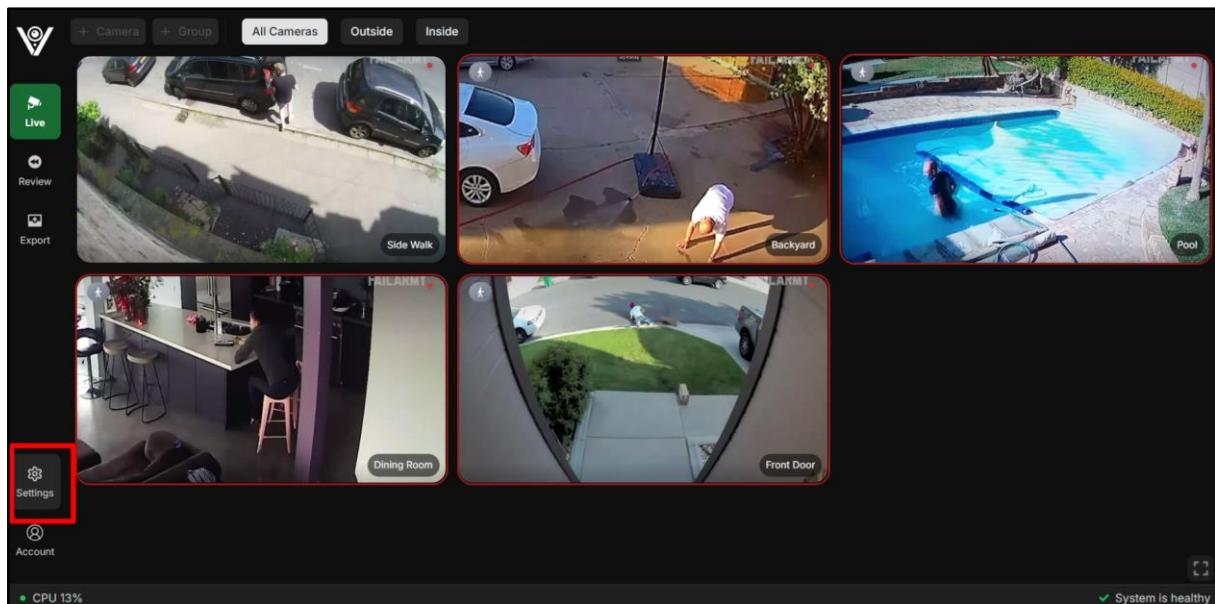


Figure 67. View defined masks and zones

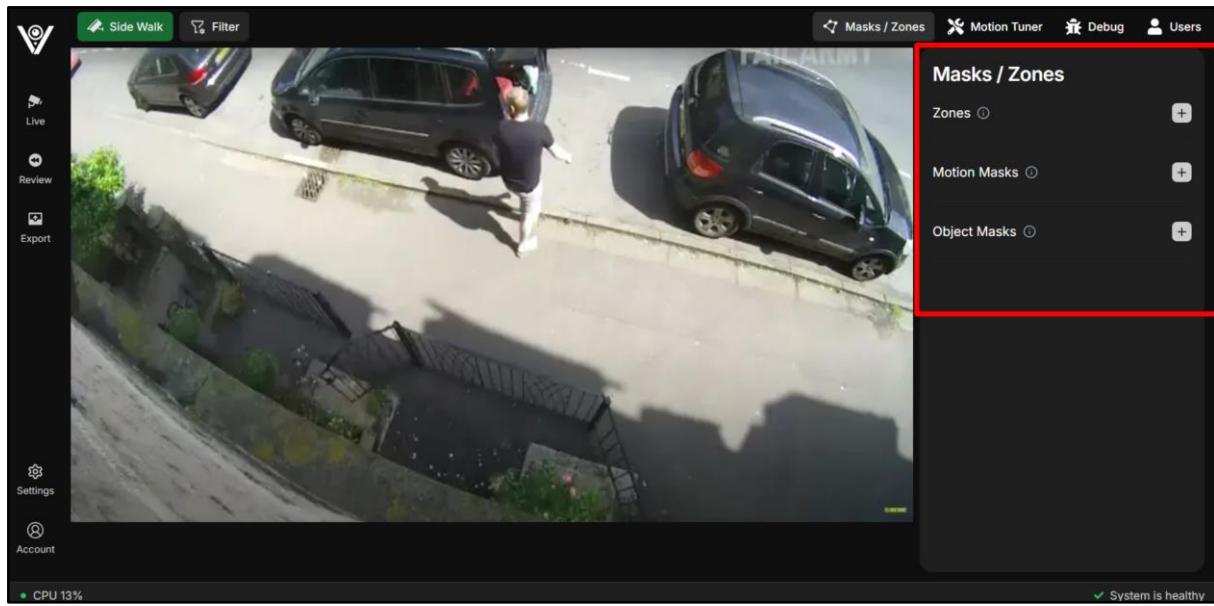


Figure 68. View defined masks and zones

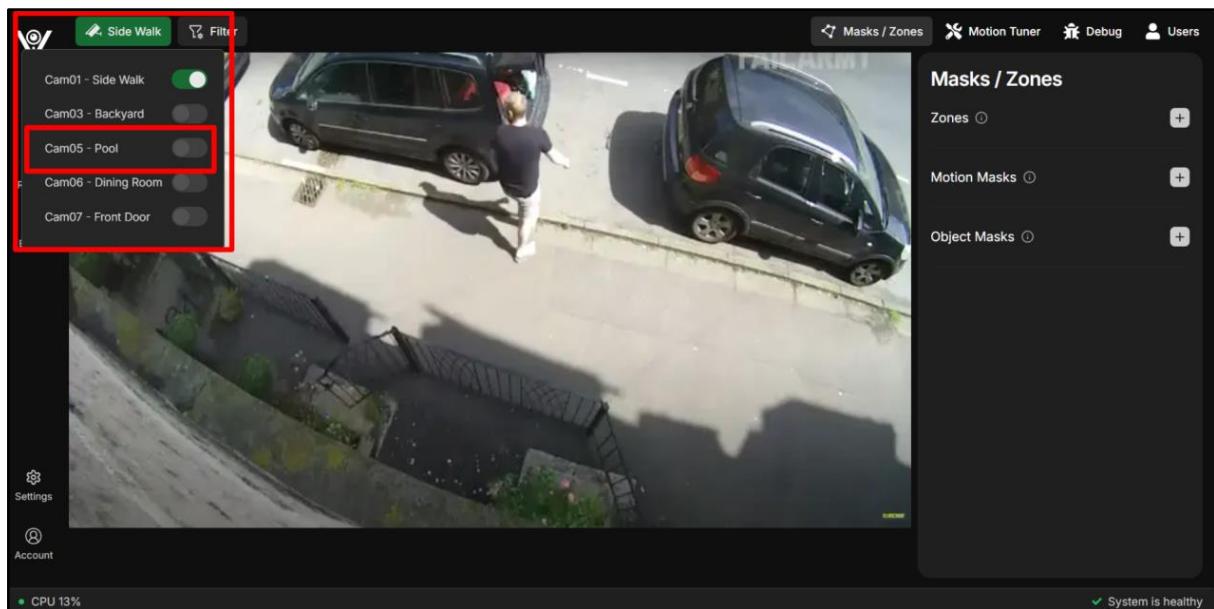


Figure 69. View defined masks and zones

2.5.2 Manage motion masks

Table 29. Manage motion masks

ID and Name:	UC-15: Manage motion masks		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	UC-15-01: Create Admin clicks on the create new motion mask button in the masks &		

	<p>zones page.</p> <p>UC-15-02: Update Admin clicks the "Update" button for an existing motion mask.</p> <p>UC-15-03: Delete Admin clicks the "Delete" button for an existing motion mask.</p>
Description:	<p>UC-15-01: Create Admin can create new motion masks to prevent unwanted motion from triggering detection. This helps in reducing false alarms and improving the accuracy of the system.</p> <p>UC-15-02: Update Admin edits an existing motion mask on a selected camera stream to modify its shape or area, refining the exclusion of unwanted motion from triggering detection.</p> <p>UC-15-03: Delete Admin deletes an existing motion mask on a selected camera stream to remove it from the system and allow the detection of motion within the previously masked area.</p>
Pre-conditions:	<ol style="list-style-type: none"> 1. Admin is logged in (for Domain user). 2. A camera stream is selected and displayed. 3. An existing motion mask is selected for editing.
Post-conditions:	<p>UC-15-01: create</p> <ol style="list-style-type: none"> 1. A new motion mask is created and saved. 2. The Admin can view the newly created motion mask on the camera stream. <p>UC-15-02: Update</p> <ol style="list-style-type: none"> 1. The selected motion mask is updated with the user's changes. 2. The Admin can view the updated motion mask on the camera stream. <p>UC-15-03: Delete</p> <ol style="list-style-type: none"> 1. The selected motion mask is permanently deleted from the system. 2. The Admin can no longer view the deleted motion mask on the camera stream. 3. The system resumes detecting motion within the previously masked area.
Normal Flow:	<p>UC-15-01: Create</p> <ol style="list-style-type: none"> 1. Admin clicks on the create new motion mask button. 2. Admin clicks and drags the mouse to draw a polygon on the camera stream. 3. Admin can undo or reset the drawing by clicking the "Undo" or "Reset" buttons. 4. Admin clicks on the "Save" button to confirm the motion mask. 5. The system saves the motion mask and displays it on the camera stream. <p>UC-15-02: Update</p> <ol style="list-style-type: none"> 1. Admin selects an existing motion mask for editing.

	<p>2. Admin clicks the "Edit" button for the selected motion mask.</p> <p>3. The system displays the selected motion mask with editable points on the camera stream.</p> <p>4. Admin drags the points or adds new ones to adjust the shape and area of the motion mask. Or Admin can undo/reset the drawing by clicking undo/reset buttons.</p> <p>5. Admin clicks "Save" to confirm the changes.</p> <p>6. The system updates the motion mask with the user's changes.</p> <p>7. The system displays the updated motion mask on the camera stream.</p> <p>UC-15-03: Delete</p> <p>1. Admin selects an existing motion mask for deletion.</p> <p>2. Admin clicks the "Delete" button for the selected motion mask.</p> <p>3. The system displays a confirmation dialog asking the Admin to confirm the deletion.</p> <p>4. Admin clicks "Confirm" to proceed with the deletion.</p> <p>5. The system permanently deletes the motion mask.</p> <p>6. The system updates the camera stream display to reflect the removal of the motion mask.</p>
Alternative Flows:	<p>UC-15-01: Create Admin clicks on the "Cancel" button. The system cancels the motion mask creation process.</p> <p>UC-15-02: Updae Admin clicks "Cancel" before saving the changes. The system discards the changes and retains the original motion mask.</p> <p>UC-15-03: Delete Admin clicks "Cancel" in the confirmation dialog. The system cancels the deletion and retains the motion mask.</p>
Exception:	
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-06, BR-07, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout

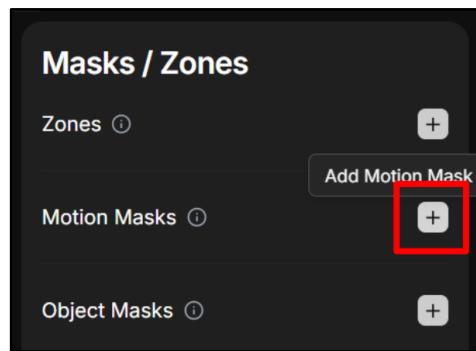


Figure 70. Create new motion mask

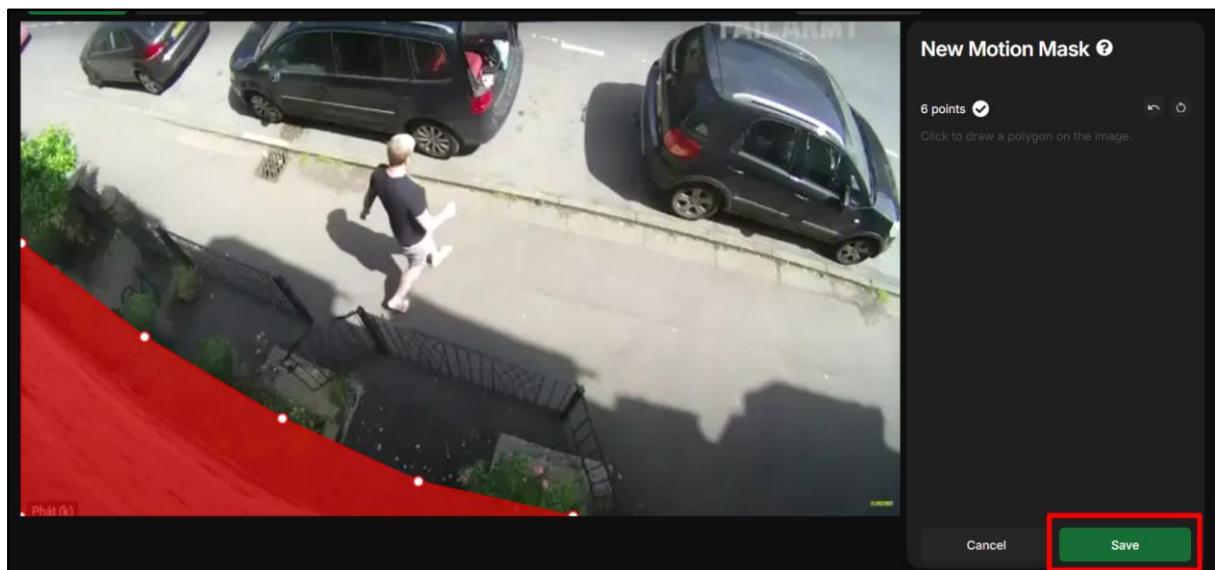


Figure 71. Create new motion mask

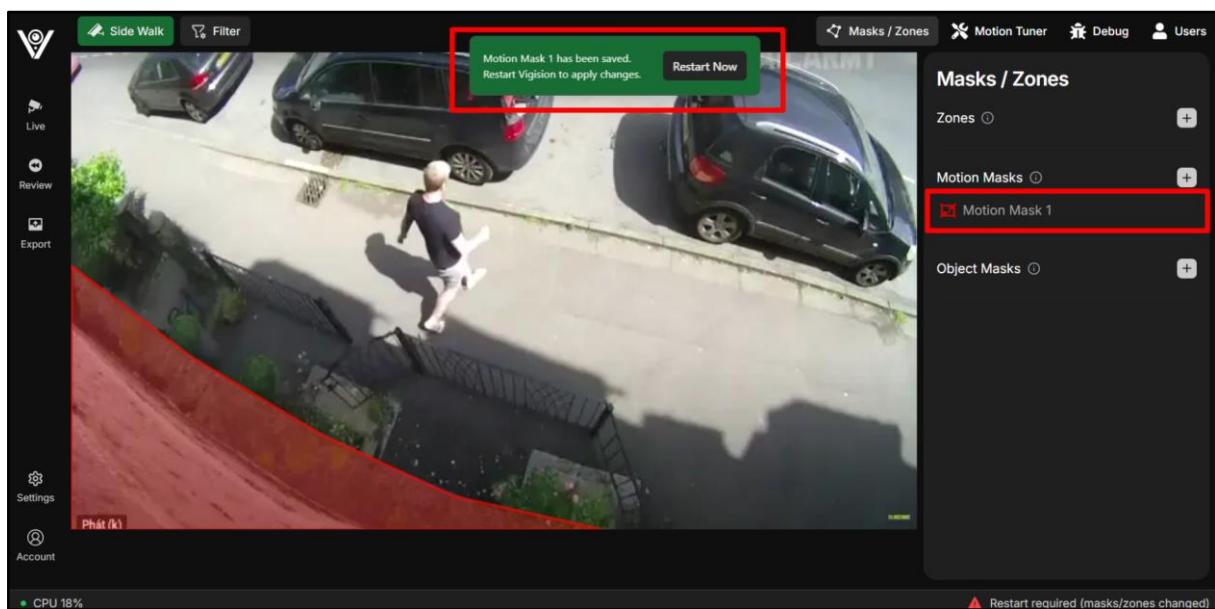


Figure 72. Create new motion mask

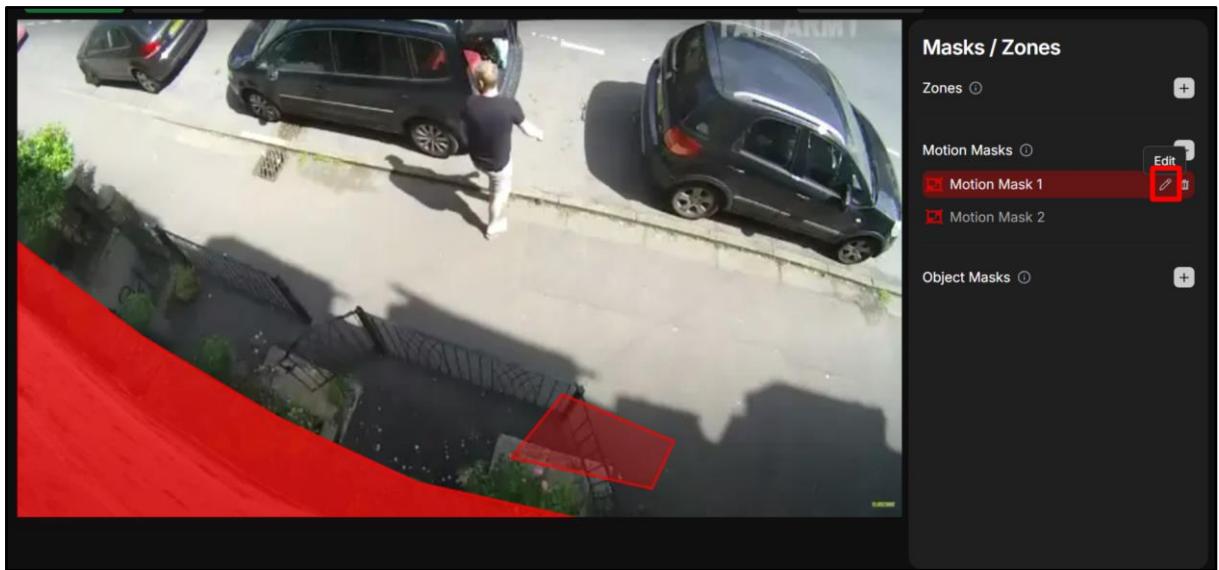


Figure 73. Edit a motion mask

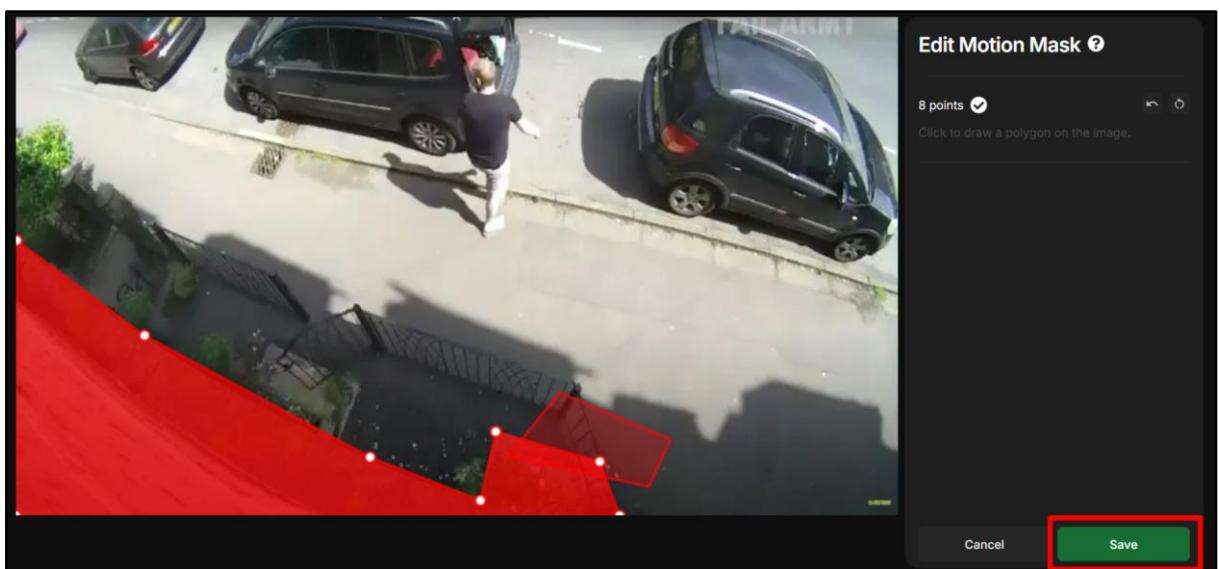


Figure 74. Edit a motion mask

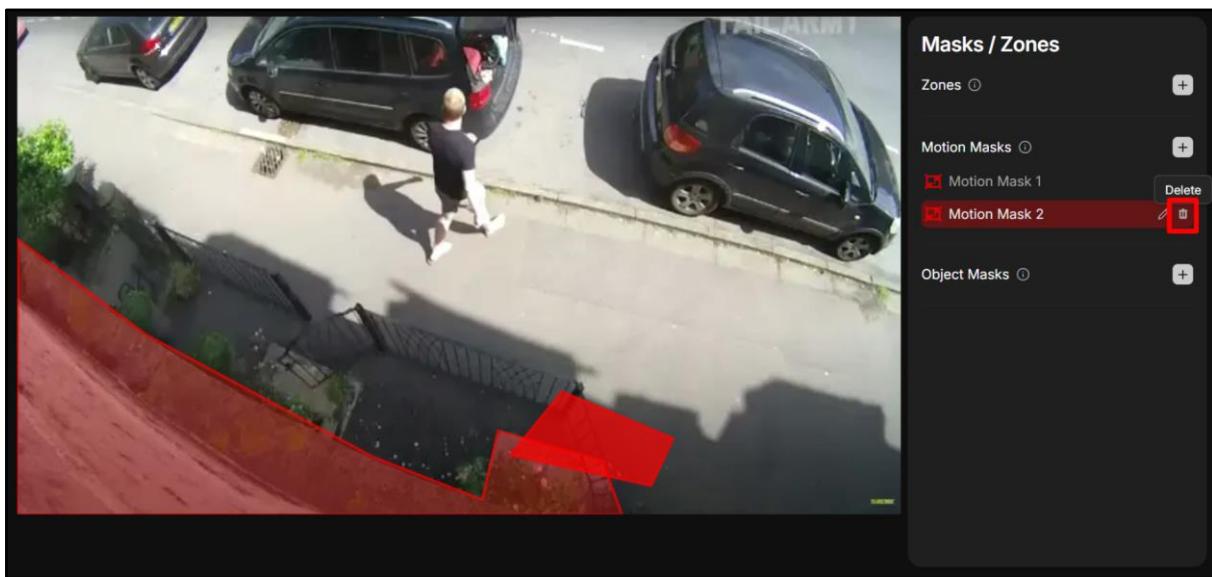


Figure 75. Delete a motion mask

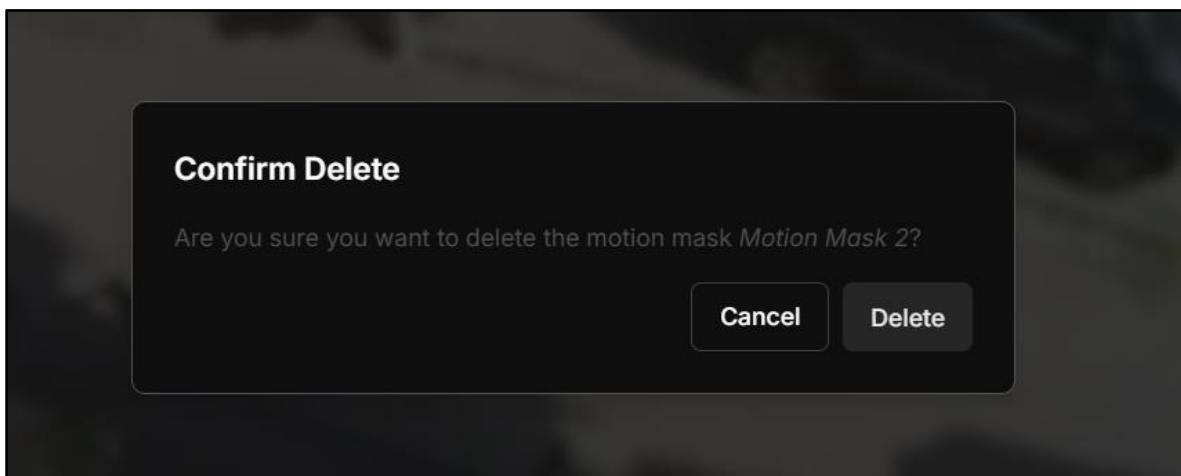


Figure 76. Delete a motion mask

2.5.3 Manage zones

Table 30. Manage zones

ID and Name:	UC-16: Manage zones		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	One of the following triggers: Admin clicks the create new zone button.		

	<p>Admin clicks the "Edit" button for an existing zone.</p> <p>Admin clicks the "Delete" button for an existing zone.</p>
Description:	<p>This use case allows the Admin to manage zones within the Vigision system. It includes the following actions:</p> <p>Create Zone: Admin creates a new zone within a selected camera stream to define a specific area of interest and configure object detection and alert settings within that area.</p> <p>Update Zone: Admin edits an existing zone on a selected camera stream to modify its shape, area, object detection settings, and alert settings.</p> <p>Delete Zone: Admin deletes an existing zone on a selected camera stream to remove it from the system and stop object detection and alert triggers within that area.</p>
Pre-conditions:	<p>UC-16-01: Create</p> <ol style="list-style-type: none"> 1. Admin is logged in. 2. A camera stream is selected and displayed. <p>UC-16-02: Update</p> <ol style="list-style-type: none"> 1. Admin is logged in. 2. A camera stream is selected and displayed. 3. An existing zone is selected for editing. <p>UC-16-03: Delete</p> <ol style="list-style-type: none"> 1. Admin is logged in. 2. A camera stream is selected and displayed. 3. An existing zone is selected for deletion.
Post-conditions:	<p>UC-16-01: Create</p> <ol style="list-style-type: none"> 1. A new zone is created and saved. 2. The Admin can view the newly created zone on the camera stream. 3. The system updates the configuration based on the new zone. <p>UC-16-02: Update</p> <ol style="list-style-type: none"> 1. The selected zone is updated with the user's changes. 2. The Admin can view the updated zone on the camera stream. 3. The system updates the object detection and alert settings based on the zone's new configuration. <p>UC-16-03: Delete</p> <ol style="list-style-type: none"> 1. The selected zone is permanently deleted from the system. 2. The Admin can no longer view the deleted zone on the camera stream. 3. The system stops detecting objects and triggering alerts within the previously defined zone area.

Normal Flow:	<p>UC-16-01: Create Zone</p> <ol style="list-style-type: none"> 1. Admin clicks on the create new zone button. 2. Admin clicks and drags the mouse to draw a polygon on the camera stream. Or Admin can undo, reset the drawing by clicking the “undo” and “reset” buttons. 3. Admin enters a unique zone name (at least 2 characters, not a camera name or existing zone name). 4. Admin specifies the number of frames an object must be in the zone before it's considered inside (default: 3). 5. Admin sets the minimum time (in seconds) an object must be in the zone to trigger an event (default: 0). 6. Admin selects the list of objects that apply to this zone. 7. Admin chooses whether to enable alerts and/or detection for objects entering this zone. 8. Admin clicks "Save" to confirm the zone creation. 9. The system saves the new zone and updates the object detection and alert settings. 10. The system displays the newly created zone on the camera stream. <p>UC-16-02: Update Zone</p> <ol style="list-style-type: none"> 1. Admin selects an existing zone for editing. 2. Admin clicks the "Edit" button for the selected zone. 3. The system displays the selected zone with editable points and configuration options on the camera stream. 4. User drags the points or adds new ones to adjust the shape and area of the zone. Or the user can undo/reset the drawing by click undo/reset buttons. 5. Admin modifies the zone name, frame count, minimum time, object list, alert settings, and detection settings as needed. 6. Admin clicks "Save" to confirm the changes. 7. The system updates the zone with the user's changes. 8. The system displays the updated zone on the camera stream. 9. The system updates the object detection and alert settings based on the zone's new configuration. <p>UC-16-03: Delete Zone</p> <ol style="list-style-type: none"> 1. Admin actor selects an existing zone for deletion. 2. Admin clicks the "Delete" button for the selected zone. 3. The system displays a confirmation dialog asking the user to confirm the deletion. 4. Admin clicks "Confirm" to proceed with the deletion. 5. The system permanently deletes the zone. 6. The system updates the camera stream display to reflect the removal of the zone.
Alternative Flows:	<p>UC-16-01: Create Zone</p> <ol style="list-style-type: none"> 1. Admin clicks on the "Cancel" button. The system cancels the motion mask creation process.

	<p>2. Admin attempts to create a zone with invalid input (e.g., overlapping with existing zones, incomplete polygon, duplicate zone name, invalid frame count or time settings). The system displays an error message and prompts the user to correct the input.</p> <p>UC-16-02: Update Zone</p> <ol style="list-style-type: none"> 1. Admin clicks "Cancel" before saving the changes. The system discards the changes and retains the original zone configuration. 2. Admin attempts to edit the zone with invalid input (e.g., overlapping with existing zones, incomplete polygon, duplicate zone name, invalid frame count or time settings). The system displays an error message and prompts the Admin to correct the input. <p>UC-16-03: Delete Zone</p> <p>Admin clicks "Cancel" in the confirmation dialog. The system cancels the deletion and retains the zone.</p>
Exception:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-06, BR-07, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout

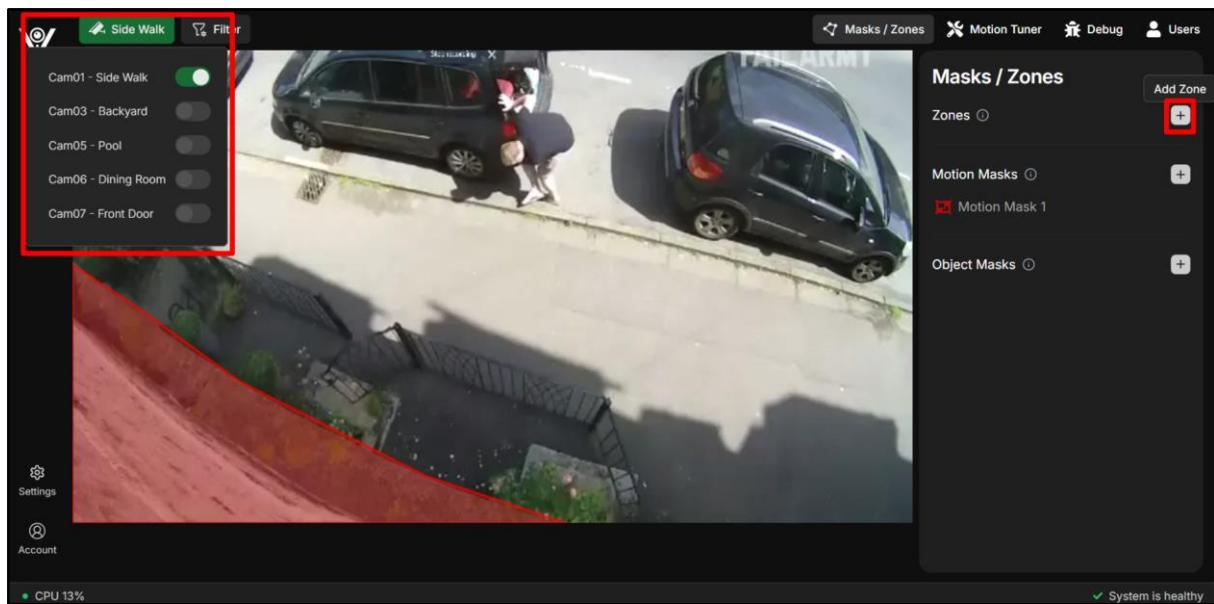


Figure 77. Manage zones - Create new zone

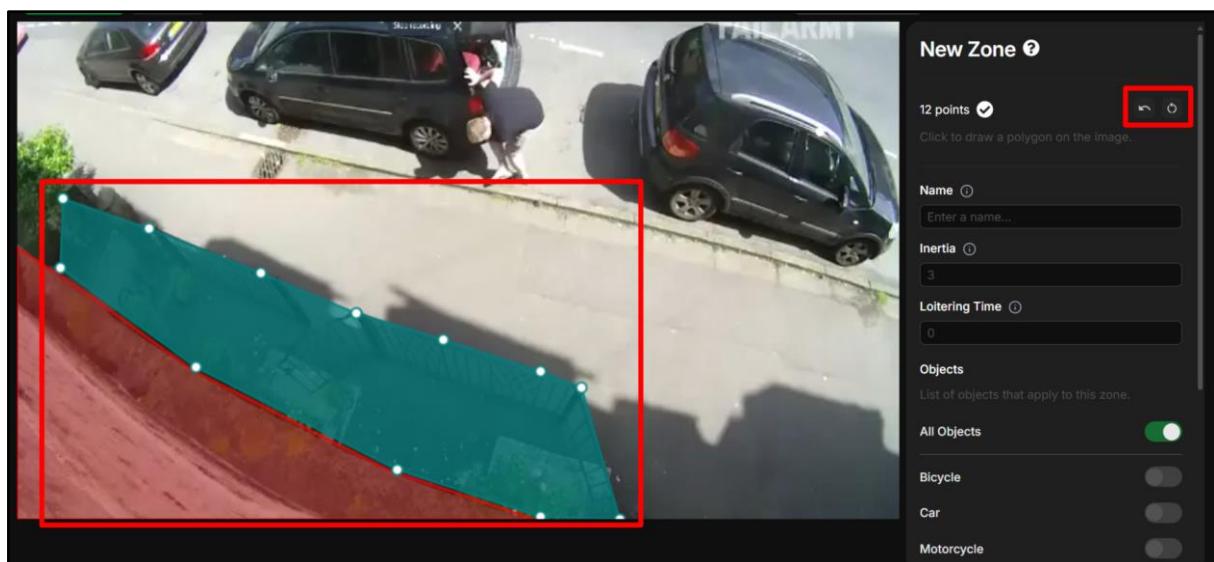


Figure 78. Manage zones - Create new zone

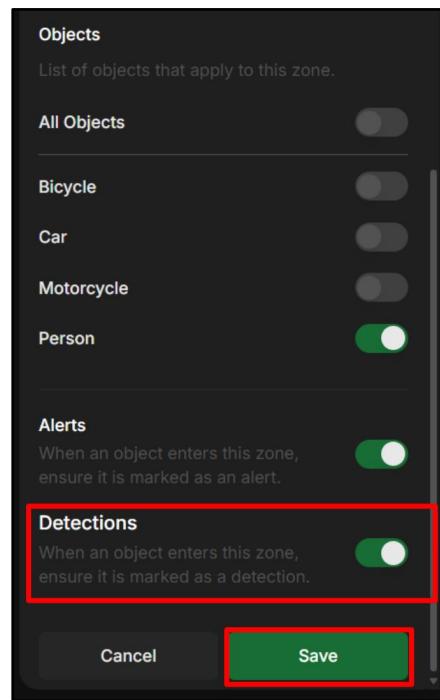


Figure 79. Manage zones - Create new zone

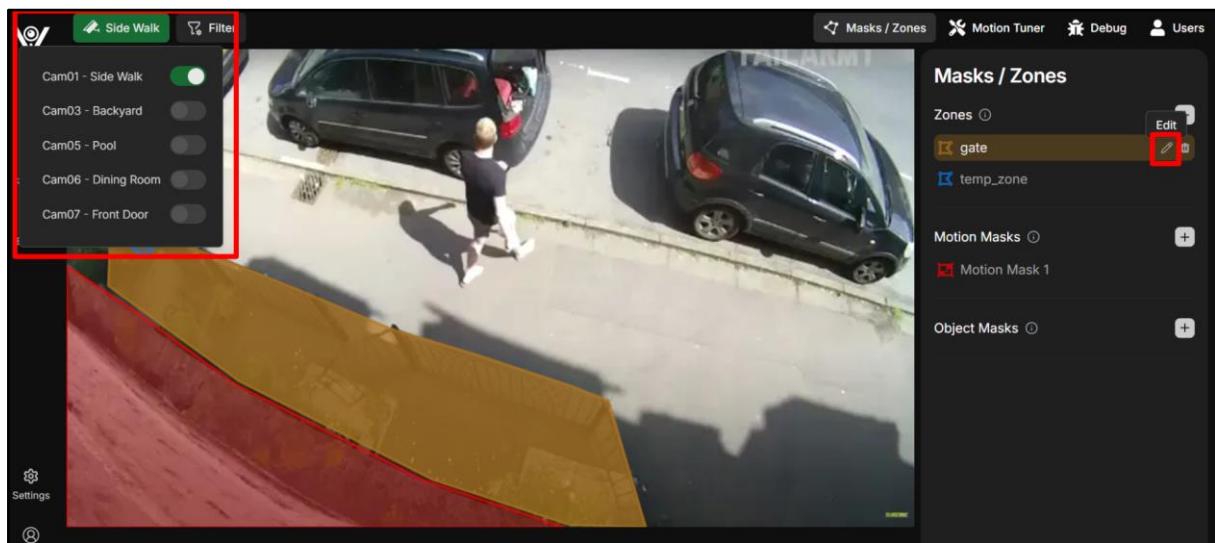


Figure 80. Manage zones - Edit a zone

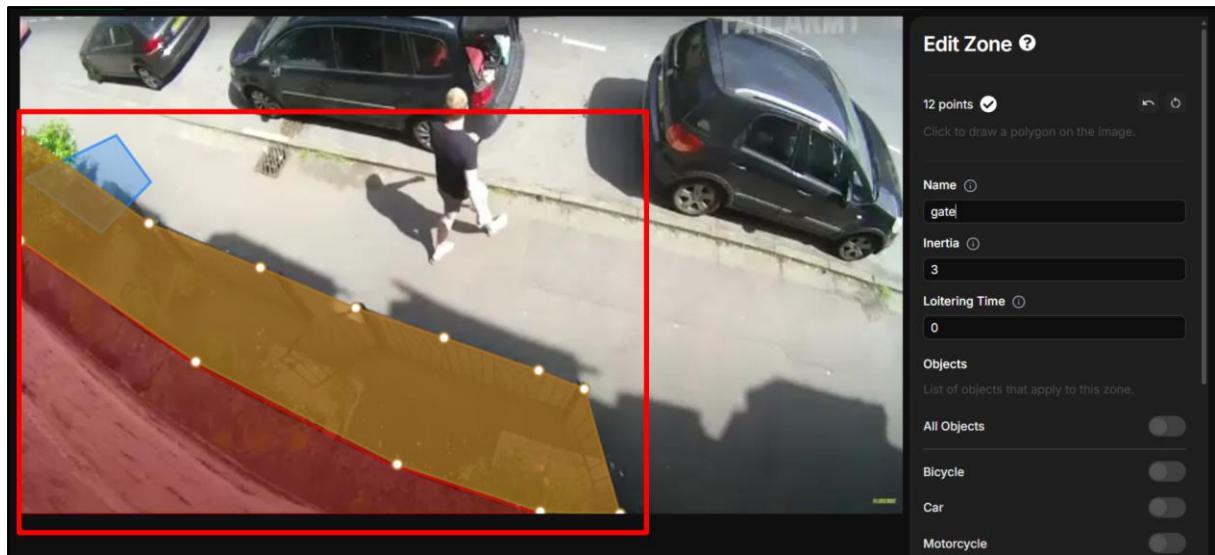


Figure 81. Manage zones - Edit a zone

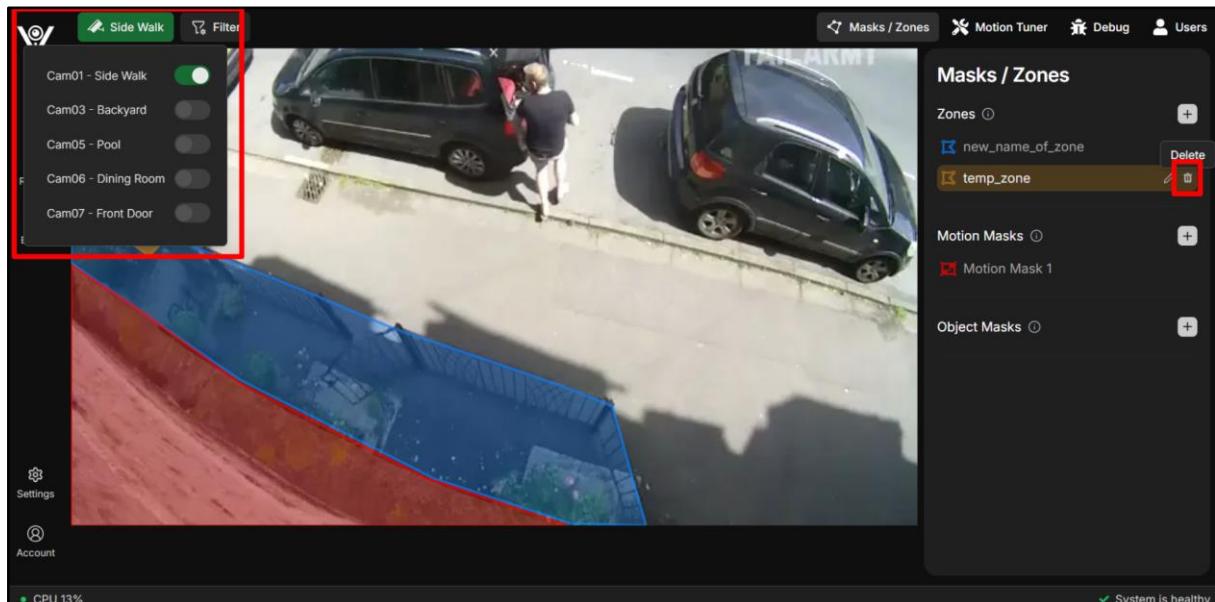


Figure 82. Manage zones - Delete a zone

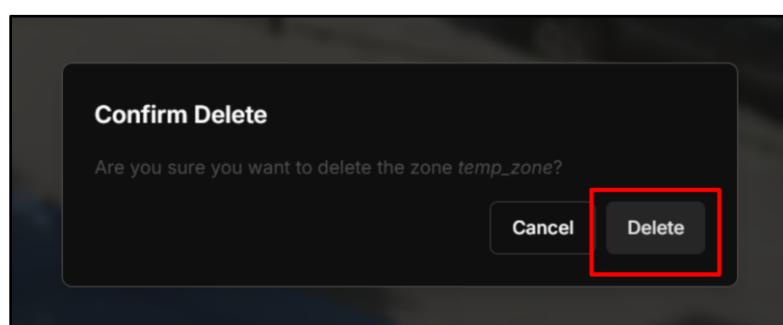


Figure 83. Manage zones - Delete a zone

2.5.4 Manage object masks

Table 31. Manage object masks

ID and Name:	UC-17: Manage object masks		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	UC-17-01: Create Admin clicks the create new object mask button. UC-17-02: Update Admin clicks the "Edit" button for an existing object mask. UC-17-03: Delete Admin clicks the "Delete" button for an existing object mask.		
Description:	UC-17-01: Create Admin creates a new object mask within a selected camera stream to filter out false positives for a specific object type based on location. Object filter masks help eliminate incorrect detections of a specific type of object by considering its position. UC-17-02: Update Admin edits an existing object mask on a selected camera stream to modify its shape or area, refining the exclusion of false positives for a specific object type within that area. UC-17-03: Delete Admin deletes an existing object mask on a selected camera stream to remove it from the system and allow the detection of the associated object type within the previously masked area.		
Pre-conditions:	1. Admin is logged in. 2. A camera stream is selected and displayed. 3. An existing object mask is selected for editing.		
Post-conditions:	UC-17-01: Create 1. A new object mask is created and saved. 2. The Admin can view the newly created object mask on the camera stream. 3. The system updates the object detection settings to exclude the masked area for the selected object type. UC-17-02: Update 1. The selected object mask is updated with the user's changes. 2. The Admin can view the updated object mask on the camera stream. 3. The system updates the object detection settings to reflect the changes in the masked area for the selected object type. UC-17-03: Delete 1. The selected object mask is permanently deleted from the system.		

	<p>2. The Admin can no longer view the deleted object mask on the camera stream.</p> <p>3. The system resumes detecting the associated object type within the previously masked area.</p>
Normal Flow:	<p>UC-17-01: Create</p> <ol style="list-style-type: none"> 1. Admin clicks on the create new object mask button. 2. Admin clicks and drags the mouse to draw a polygon on the camera stream. 3. Admin selects the object type that applies to this object mask. 4. Admin clicks "Save" to confirm the object mask creation. 5. The system saves the new object mask and updates the object detection settings. 6. The system displays the newly created object mask on the camera stream. <p>UC-17-02: Update</p> <ol style="list-style-type: none"> 1. Admin selects an existing object mask for editing. 2. Admin clicks the "Edit" button for the selected object mask. 3. The system displays the selected object mask with editable points on the camera stream. 4. Admin drags the points or adds new ones to adjust the shape and area of the object mask. 5. Admin may also change the object type associated with the mask. 6. Admin clicks "Save" to confirm the changes. 7. The system updates the object mask with the user's changes. 8. The system displays the updated object mask on the camera stream. 9. The system updates the object detection settings to reflect the changes in the masked area for the selected object type. <p>UC-17-03: Delete</p> <ol style="list-style-type: none"> 1. Admin selects an existing object mask for deletion. 2. Admin clicks the "Delete" button for the selected object mask. 3. The system displays a confirmation dialog asking the user to confirm the deletion. 4. Admin clicks "Confirm" to proceed with the deletion. 5. The system permanently deletes the object mask. 6. The system updates the camera stream display to reflect the removal of the object mask.
Alternative Flows:	<p>UC-17-01: Create</p> <p>Admin clicks "Cancel" before saving the object mask. The system discards the drawing and does not create a new object mask.</p> <p>UC-17-02: Update</p> <p>Admin clicks "Cancel" before saving the changes. The system discards the changes and retains the original object mask.</p> <p>UC-17-03: Delete</p> <p>Admin clicks "Cancel" before saving the changes. The system discards the changes and retains the original object mask.</p>
Exception:	-

Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-06, BR-07, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout

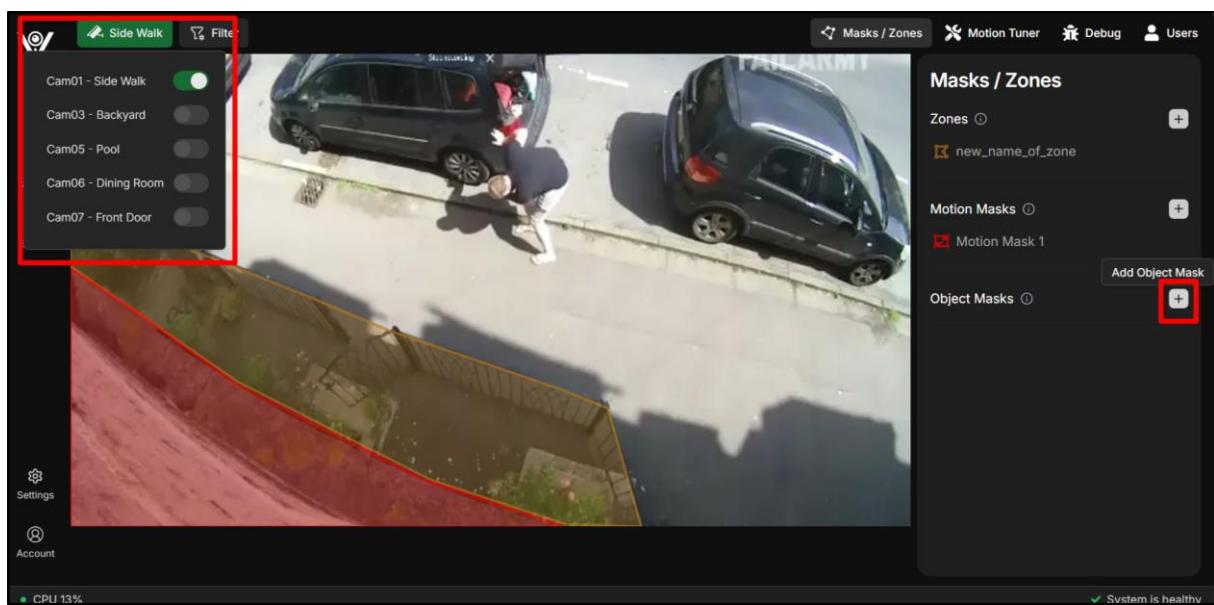


Figure 84. . Manage object masks - Create new object mask

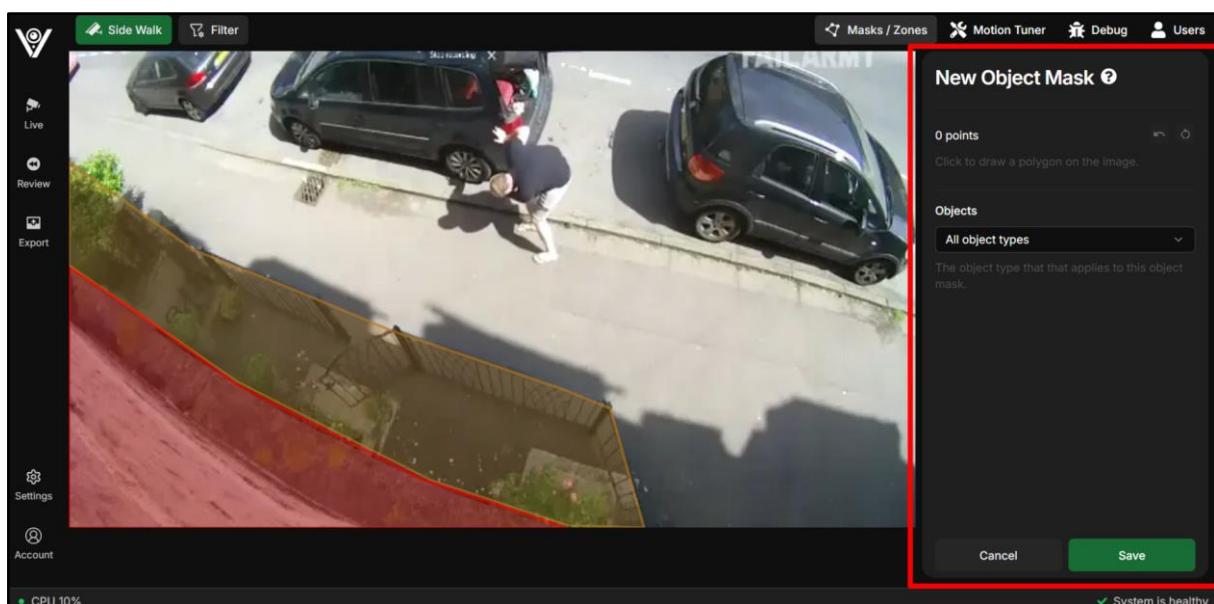


Figure 85. . Manage object masks - Create new object mask



Figure 86. Manage object masks - Create new object mask

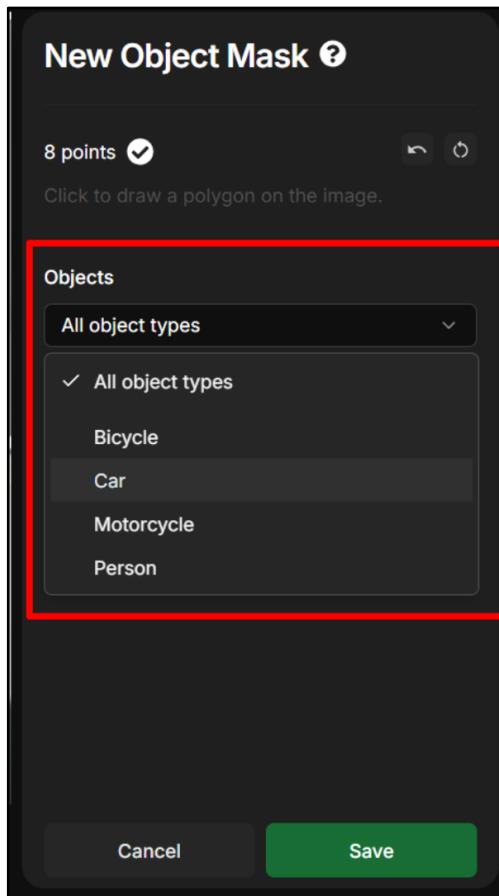


Figure 87. Manage object masks - Create new object mask

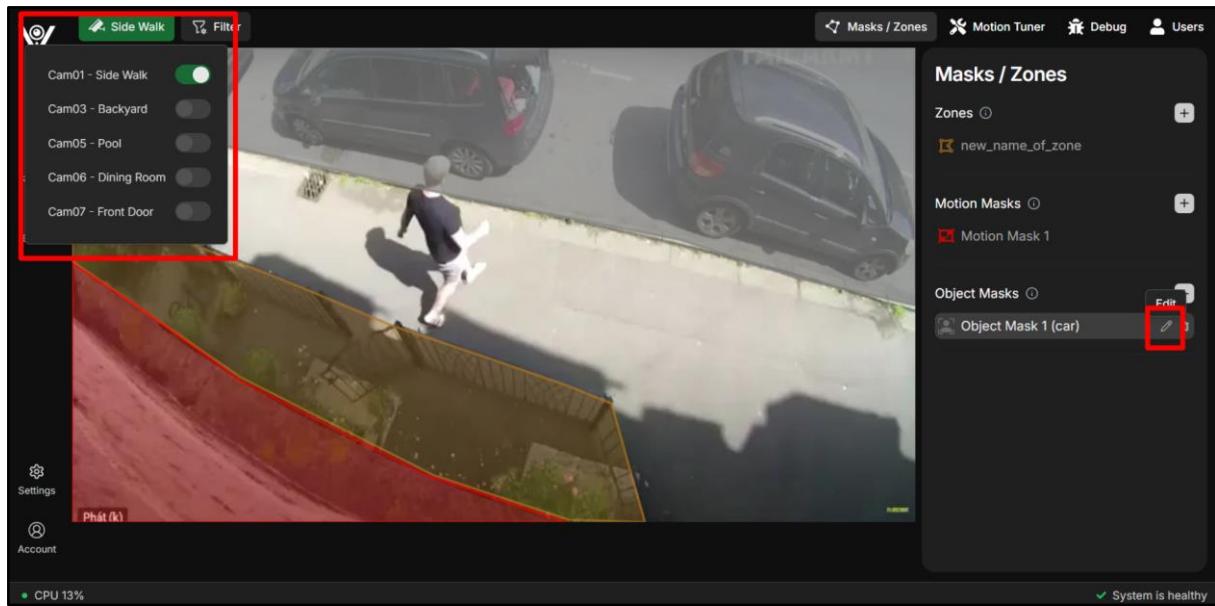


Figure 88. . Manage object masks - Edit an object mask

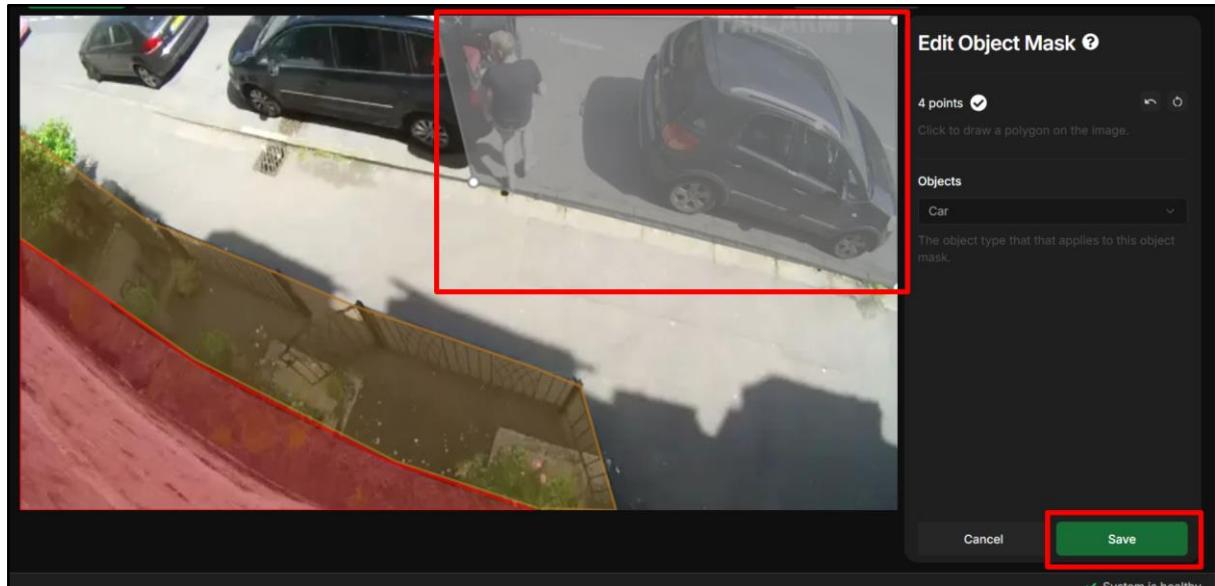


Figure 89. . Manage object masks - Edit an object mask

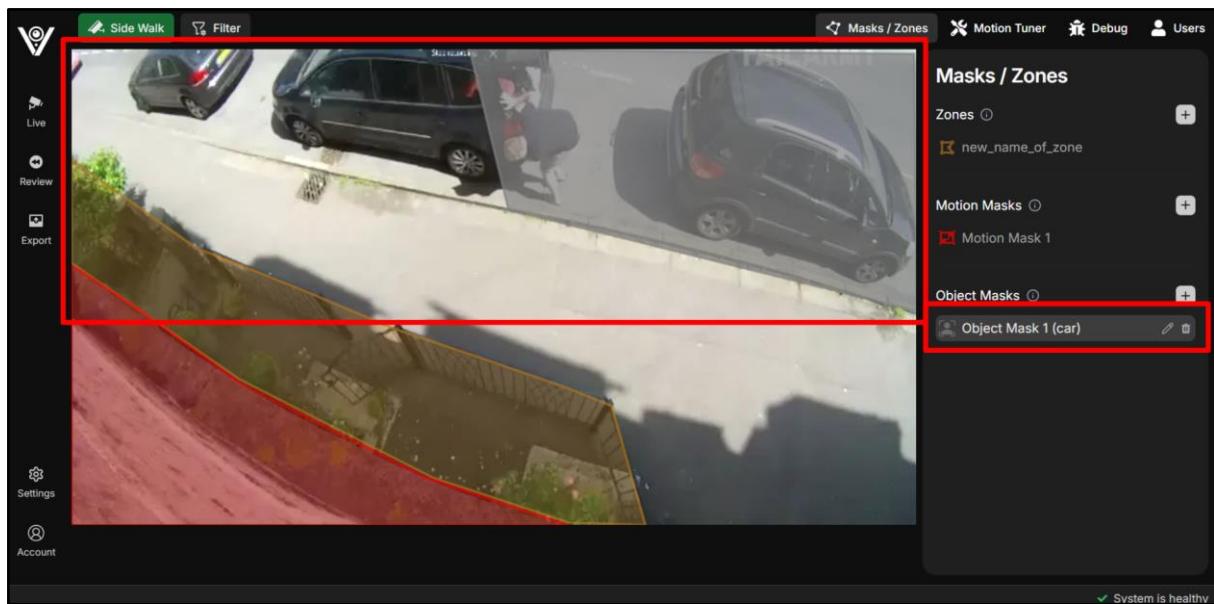


Figure 90. Manage object masks - Delete an object mask

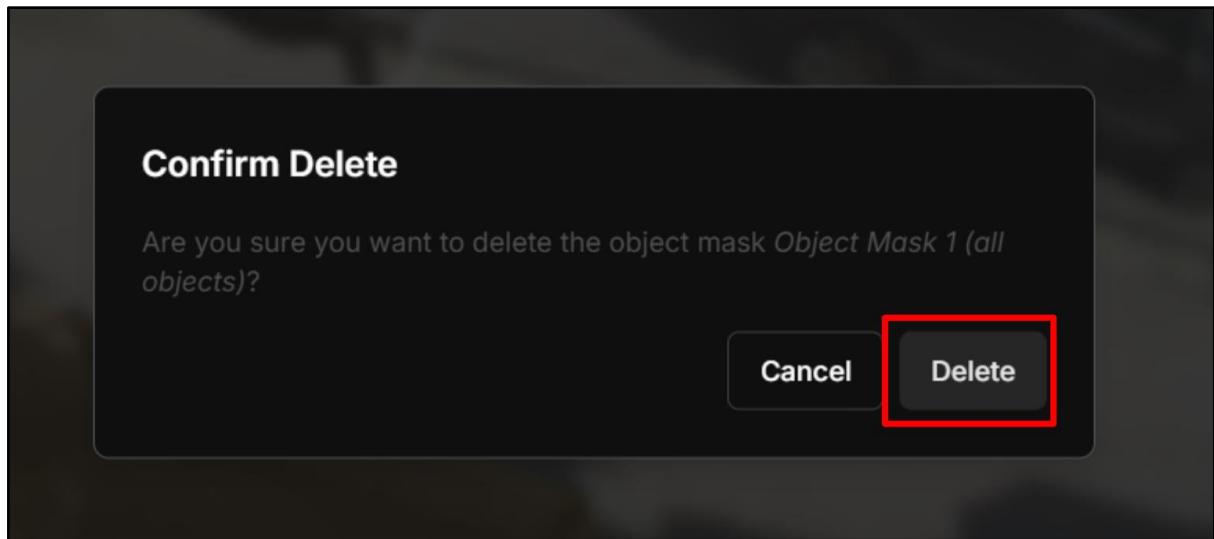


Figure 91. Manage object masks - Delete an object mask

2.5.5 Fine-tune motion detection

Table 32. Fine-tune motion detection

ID and Name:	UC-18: Fine-tune motion detection		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to fine-tune motion detection settings from the camera settings or live dashboard.		

Description:	This use case allows the Admin to adjust the motion detection settings for a specific camera. The Admin can fine-tune parameters such as sensitivity, detection zones, and threshold values to optimize the motion detection performance. Fine-tuning motion detection helps reduce false alarms and ensures that the system accurately detects significant events.
Pre-conditions:	The Admin must be logged into the system
Post-conditions:	The motion detection settings for the selected camera are fine-tuned and saved.
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the camera detail page or the live dashboard. 2. The Admin selects the camera for which they want to fine-tune motion detection. 3. The Admin clicks on the "Fine-Tune Motion Detection" button. 4. The system displays the current motion detection settings for the selected camera. 5. The Admin adjusts the motion detection parameters (e.g., sensitivity, detection zones, threshold values). 6. The Admin clicks the "Save" button. 7. The system updates the motion detection settings in the database. 8. The system displays a success message indicating that the motion detection settings have been fine-tuned. 9. The updated settings are applied to the selected camera.
Alternative Flows:	-
Exception:	System error during the motion detection fine-tuning process (e.g., database connection issue). The system displays a generic error message indicating that there was an issue fine-tuning the motion detection settings and logs the error.
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-06, BR-07, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout:



Figure 92. Fine-tune motion detection

2.5.6 Debug

Table 33. Debug

ID and Name:	UC-19: Debug		
Created By:	Phan Hong Phuc	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin navigates to the "Debug" page for a selected camera stream.		
Description:	Admin accesses the debug page for a selected camera stream to view and interact with various debugging options including bounding boxes, motion boxes, timestamps, regions, zones, motion masks, pose estimation, and fall detection.		
Pre-conditions:	1. Admin is logged in. 2. A camera stream is selected and displayed.		
Post-conditions:	1. The Admin is presented with the debug page for the selected camera stream, displaying available debug options. 2. The Admin can toggle on some debug options to make the selected debugging objects visible. Or toggle off to hide.		
Normal Flow:	1. Admin navigates to the "Debug" page. 2. Admin selects a camera stream. 3. The system displays the selected camera stream. 4. The debug page presents a list of available debug options, including: - Bounding Boxes toggle		

	<ul style="list-style-type: none"> - Motion Boxes toggle - Regions toggle - Zones toggle - Motion Masks toggle - Timestamp toggle - Pose estimation - Fall detection <p>5. Admin can interact with the debug options by clicking the toggles or adjusting other settings.</p>
Alternative Flows:	-
Exception:	-
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-06, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout

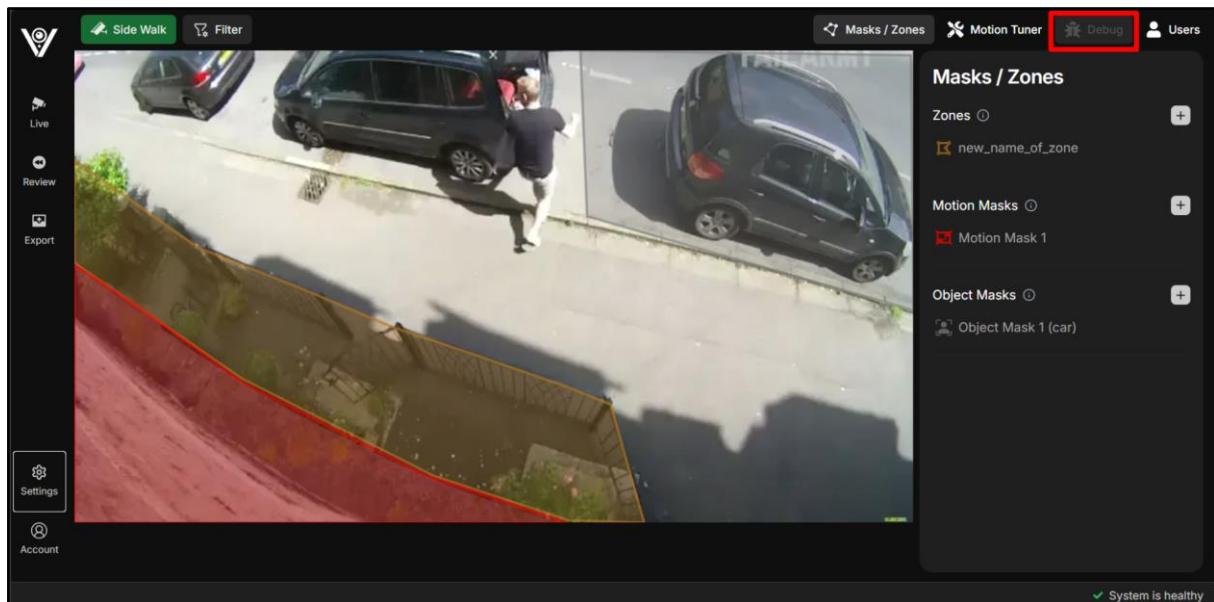


Figure 93. Debug - View debug options



Figure 94. Debug - View debug options



Figure 95. Debug - View debug options



Figure 96. Debug - Debug bounding boxes



Figure 97. Debug - Debug motion boxes



Figure 98. Debug - Add timestamp on video



Figure 99. Debug - Debug regions

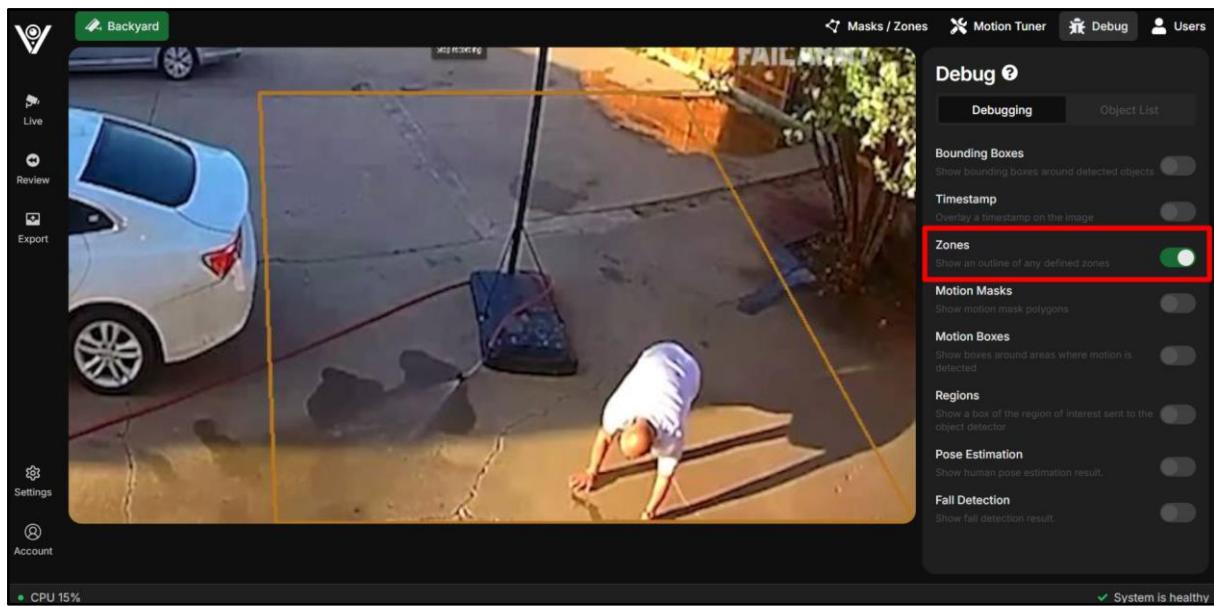


Figure 100. Debug - Debug zones

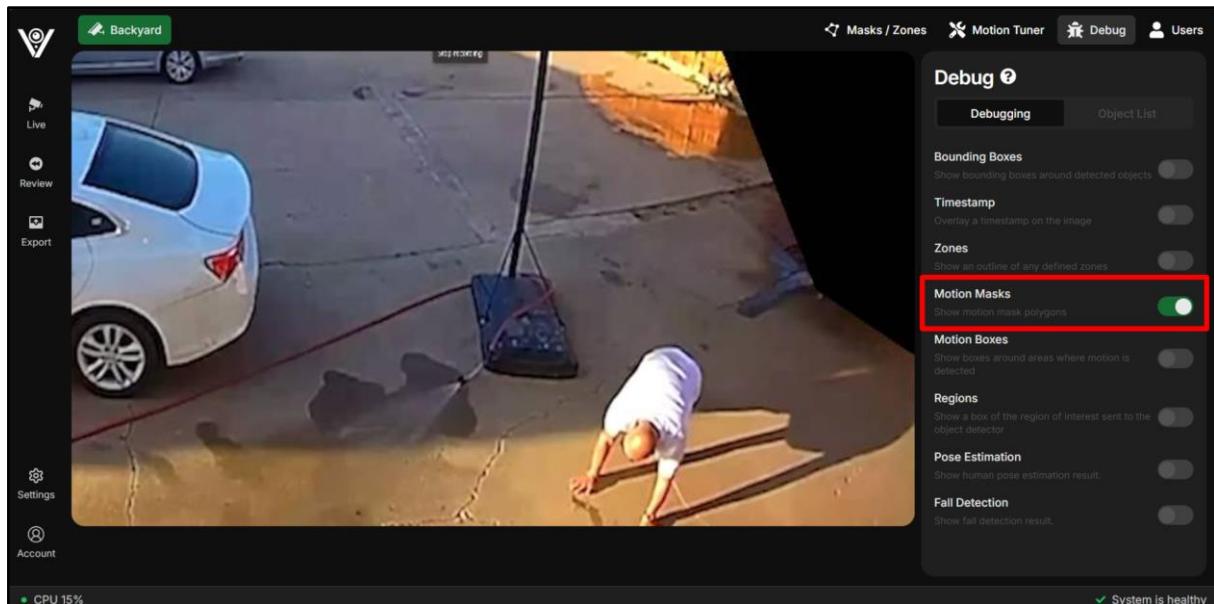


Figure 101. Debug - Debug motion masks

2.6 Events

2.6.1 View alerts/detections

Table 34. View alerts/detections

ID and Name:	UC-20: View alerts/detections		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-

Trigger:	Admin wants to review recorded events captured by the surveillance system.
Description:	This use case allows the Admin to access and review recorded events captured by the Vigision surveillance system. Events could include instances of motion detection, object detection, or human fall detection.
Pre-conditions:	1. Admin is logged in. 2. There must be recorded events available in the system.
Post-conditions:	The Admin has reviewed the recorded events and can take further actions based on them.
Normal Flow:	1. Admin navigates to the "Events" section within the Vigision application. 2. The application displays a list of recorded events, sorted by date and time.
Alternative Flows:	1. If there are no recorded events available, the application informs the user that there are no events to display. 2. If the Admin encounters a playback error while trying to view a recording, the application provides an error message and suggests troubleshooting steps.
Exception:	-
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-08, BR-09, BR-18, BR-19, BR-21
Other Information:	-
Assumption:	-

Screen layout:

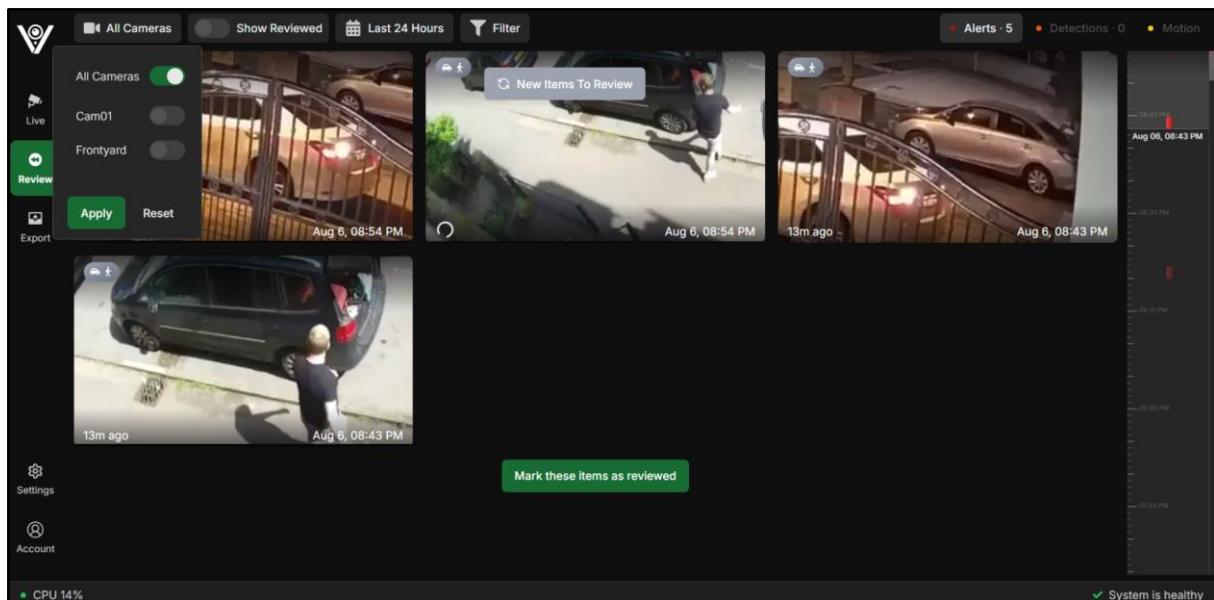


Figure 102. View events

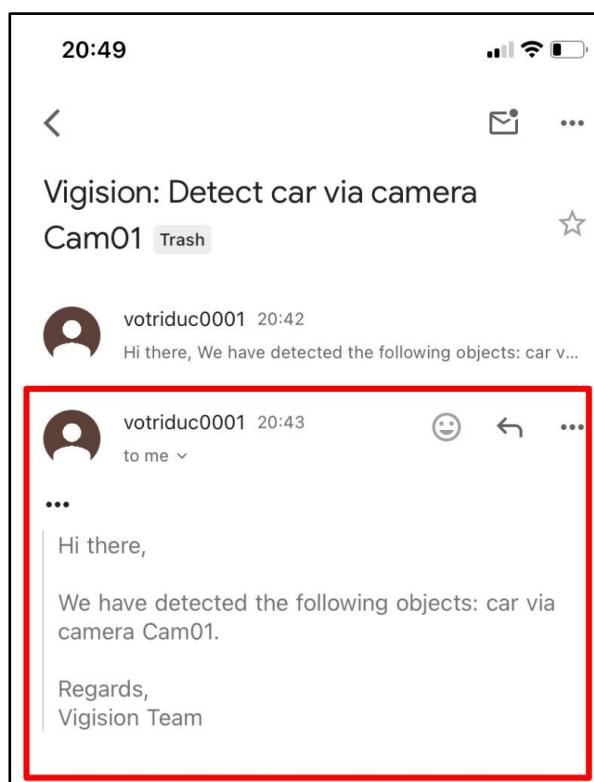


Figure 103. View events

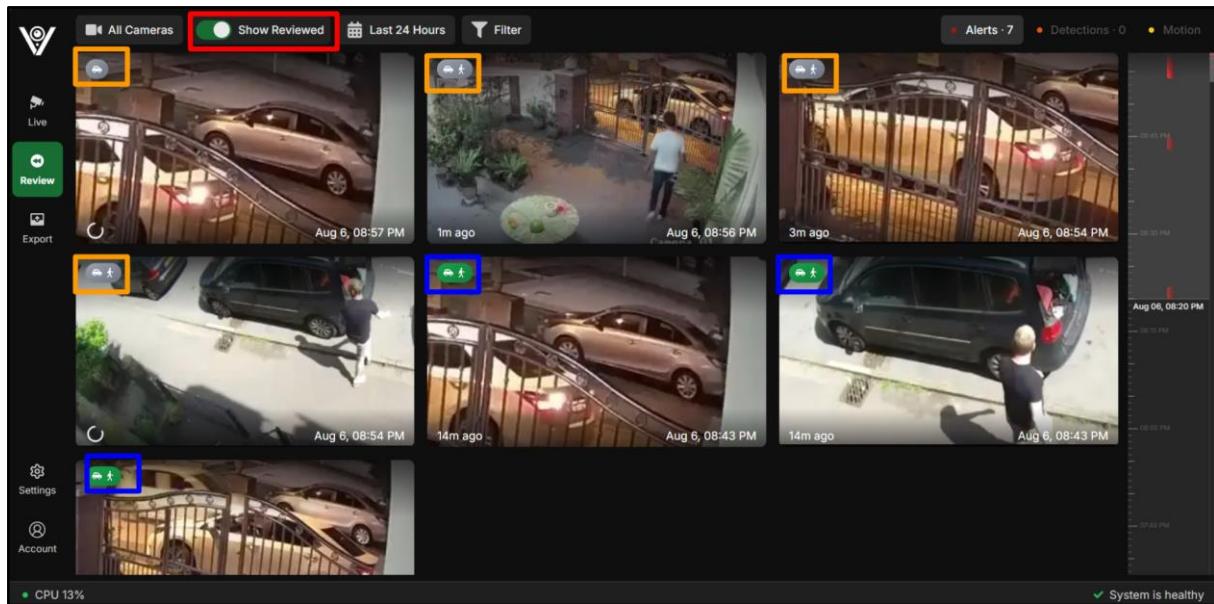


Figure 104. View events

2.6.2 View motion events

Table 35. View motion events

ID and Name:	UC-21: View motion events		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to view motion events from the events management section.		
Description:	This use case allows the user to view a list of motion events recorded by the Vigision system. Motion events are triggered when the system detects movement within the camera feed. Viewing these events helps users analyze motion patterns and identify potential security incidents.		
Pre-conditions:	The Admin must be logged into the system		
Post-conditions:	The Admin can view and review all recorded motion events.		
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the events management section. 2. The Admin selects the "Motion Events" tab. 3. The system retrieves and displays a list of motion events recorded by the system. 4. The Admin reviews the list of motion events, including details such as date, time, camera, and duration of motion. 5. The Admin can click on individual motion events to view more details and 		

	take further actions if needed.
Alternative Flows:	-
Exception:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-08, BR-09, BR-18, BR-19, BR-21
Other Information:	-
Assumption:	-

Screen layout:

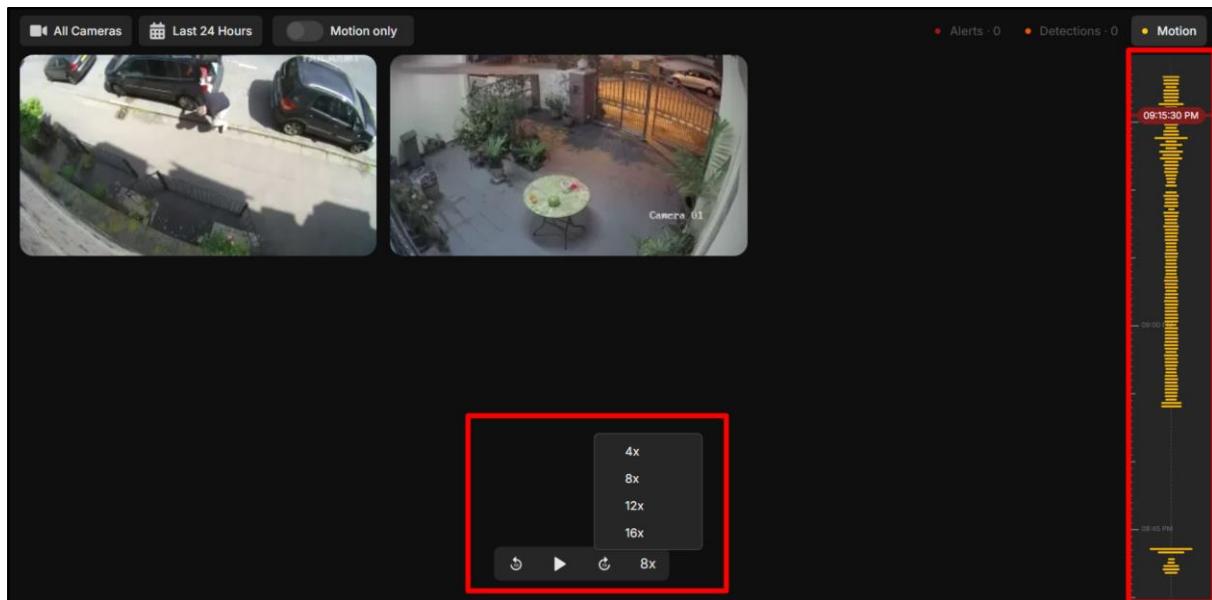


Figure 105. View motion events

2.6.3 View event recording

Table 36. View event recording

ID and Name:	UC-22: View event recording		
Created By:	Dinh The Anh	Date Created:	25/05/2024

Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects an event to view its recording from the events management section.		
Description:	This use case allows the user to view the recorded video of a specific event. Event recordings capture the video footage related to alerts, detections, and motion events. Viewing these recordings helps users analyze the events in detail and take appropriate actions.		
Pre-conditions:	The Admin must be logged into the system		
Post-conditions:	The Admin can view the recorded video of the selected event.		
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the events management section. 2. The Admin selects an event from the list. 3. The system retrieves and displays the recorded video of the selected event. 4. The Admin reviews the video footage to analyze the event. 5. The Admin can take further actions such as marking the event as reviewed, exporting the recording, or deleting the event. 		
Alternative Flows:	-		
Exception:	-		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	BR-01, BR-08, BR-09, BR-18, BR-19, BR-22, BR-23		
Other Information:	-		
Assumption:	-		

Screen layout:

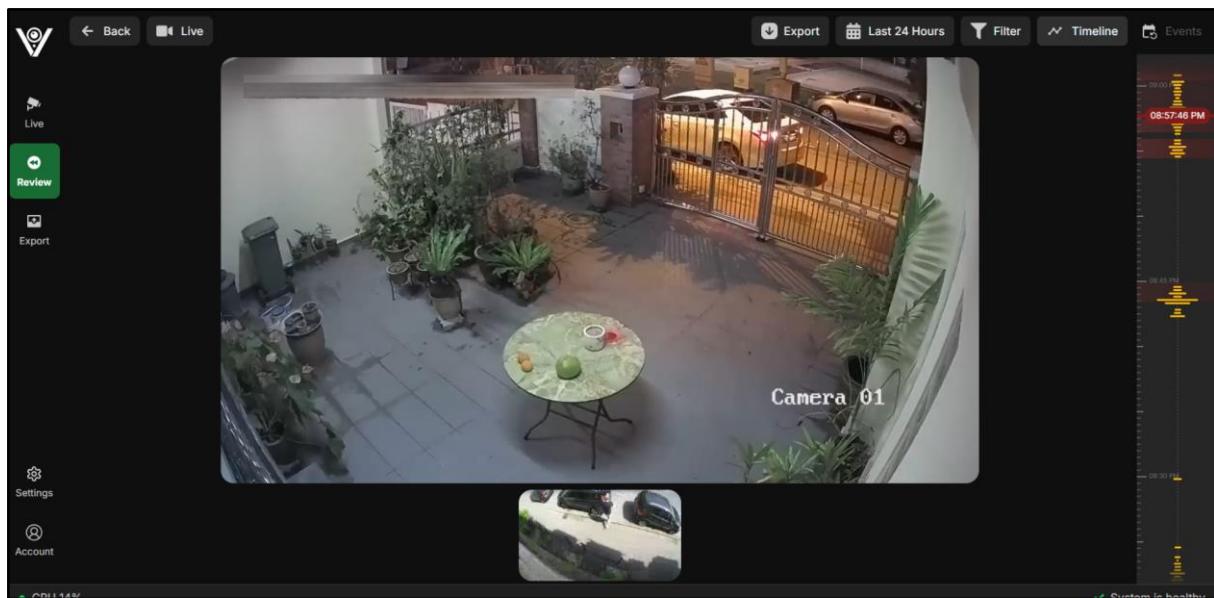


Figure 106. View event recording

2.6.4 Mark events as reviewed

Table 37. Mark events as reviewed

ID and Name:	UC-23: Mark events as reviewed		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to mark one or multiple events as reviewed from the events management section.		
Description:	This use case allows Admin to mark events as reviewed, indicating that they have been checked and processed. Marking events as reviewed helps users keep track of which events have been addressed and which ones still require attention.		
Pre-conditions:	The Admin must be logged into the system		
Post-conditions:	The selected events are marked as reviewed in the system.		
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the events management section. 2. The Admin selects one or multiple events from the list. 3. The Admin clicks on the "Mark as Reviewed" button. 4. The system updates the status of the selected events to "Reviewed". 5. The system displays a success message indicating that the events have been marked as reviewed. 		

Alternative Flows:	-
Exception:	System error while updating the event status (e.g., database connection issue). The system displays a generic error message indicating that there was an issue marking the events as reviewed and logs the error.
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-08, BR-09, BR-18, BR-19, BR-21
Other Information:	-
Assumption:	-

Screen layout:

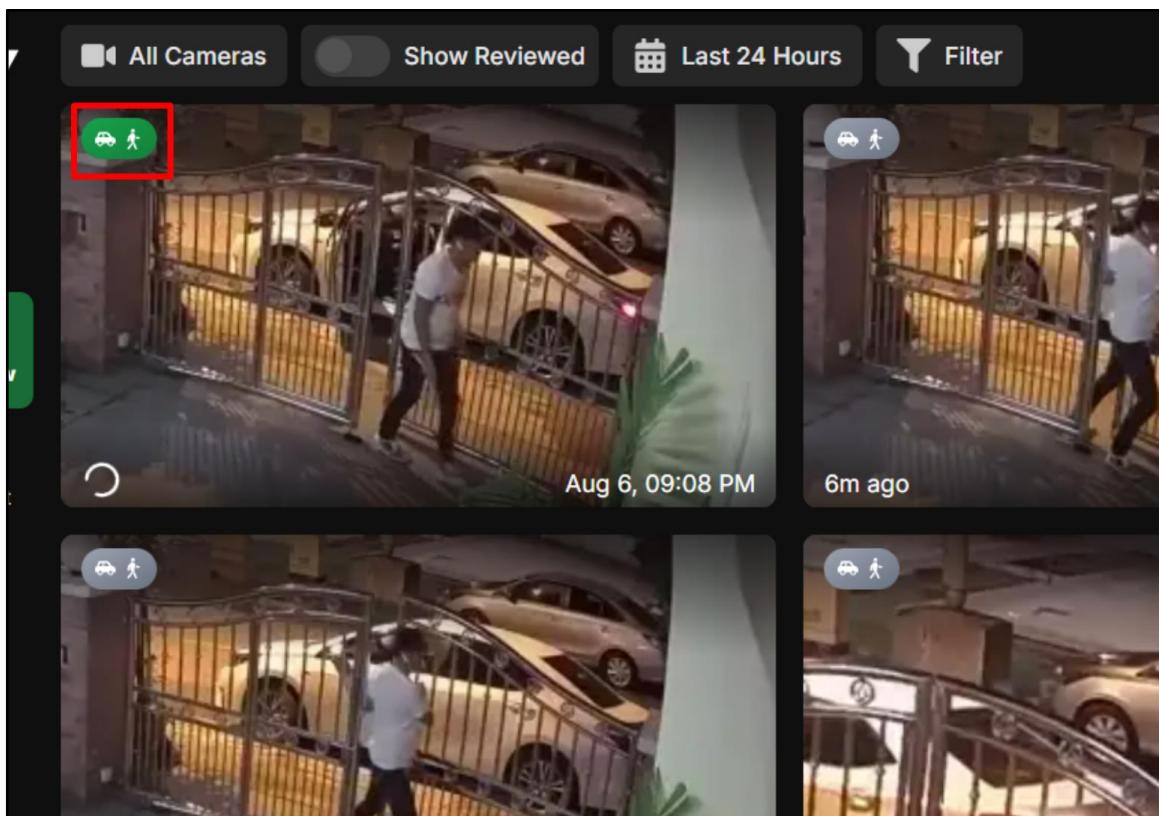


Figure 107. Mark events as reviewed

2.6.5 Delete events

Table 38. Delete events

ID and Name:	UC-24: Delete events		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin selects specific events from the list of recorded events to delete them.		
Description:	This use case allows the user to remove unwanted or irrelevant events recorded by the Vigision surveillance system from the event log.		
Pre-conditions:	1. Admin is logged in. 2. Recorded events must be available in the system.		
Post-conditions:	The selected events are successfully deleted from the event log.		
Normal Flow:	1. Admin navigates to the "Events" section within the Vigision application. 2. The application presents a list of recorded events, sorted by date and time. 3. Admin selects one or more specific events from the list that they want to delete. 4. The application prompts the user to confirm the deletion action. 5. Admin confirms the deletion. 6. The application removes the selected events from the event log.		
Alternative Flows:	1. If the Admin accidentally selects events for deletion, they can cancel the action before confirming. 2. If the Admin wants to delete all events within a certain timeframe or meeting specific criteria, they can use advanced filtering options before initiating the deletion process.		
Exception:	-		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	BR-01, BR-08, BR-09, BR-14, BR-18, BR-19, BR-21, BR-25, BR-26		
Other Information:	-		
Assumption:	-		

Screen layout:

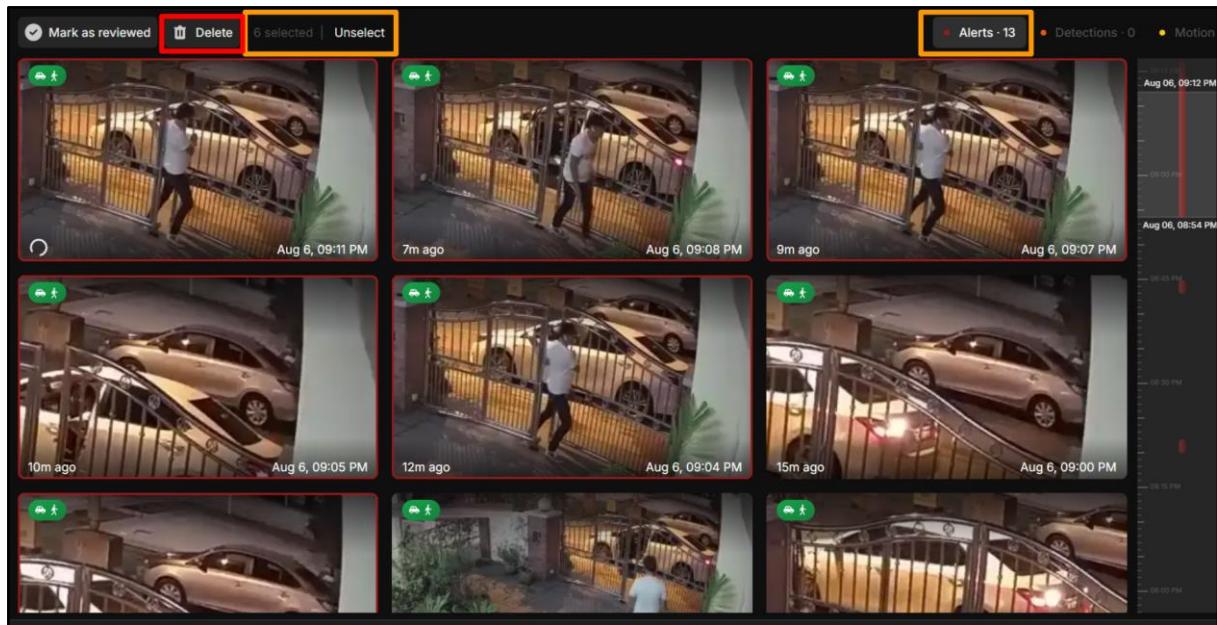


Figure 108. Delete events

2.7 Exports

2.7.1 View exports

Table 39. View exports

ID and Name:	UC-25: View exports		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to view exports from the exports management section.		
Description:	This use case allows the Admin to view a list of all exports available in the Vigision system. Exports include event recordings and camera recordings that have been saved for external use. Viewing these exports helps users manage and review the saved surveillance data.		
Pre-conditions:	The Admin must be logged into the system and have the necessary permissions to view exports.		
Post-conditions:	The Admin can view and review all available exports.		

Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the exports management section. 2. The system retrieves and displays a list of available exports categorized by type (event recordings, camera recordings). 3. The Admin reviews the list of exports, including details such as date, time, camera, and type of export. 4. The Admin can click on individual exports to view more details and take further actions if needed.
Alternative Flows:	-
Exception:	System error while fetching exports (e.g., database connection issue). The system displays a generic error message indicating that there was an issue retrieving the exports and logs the error.
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-09, BR-13, BR-19
Other Information:	-
Assumption:	-

Screen layout:

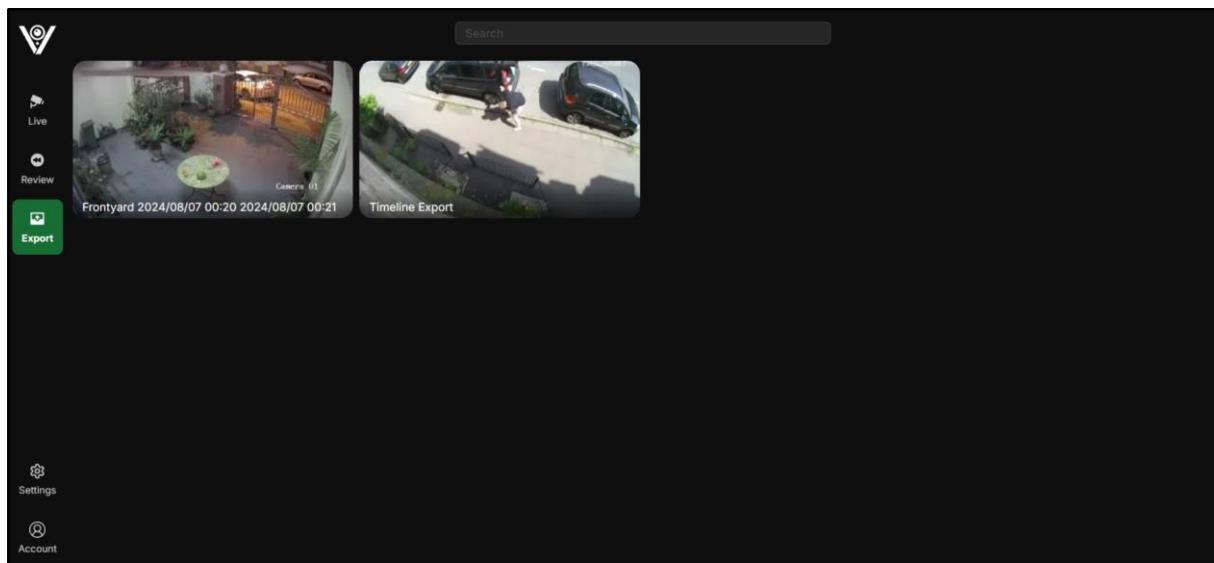


Figure 109. View exports

2.7.2 View export detail

Table 40. View export detail

ID and Name:	UC-26: View export detail		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects an export to view its details from the exports management section.		
Description:	This use case allows the Admin to view detailed information about a specific export. Export details include metadata such as the date, time, camera, and duration of the recording, as well as options for further actions such as downloading or renaming the export. Viewing export details helps users manage and review the saved surveillance data comprehensively.		
Pre-conditions:	The Admin must be logged into the system and have the necessary permissions to view exports.		
Post-conditions:	The Admin can view detailed information about the selected export.		
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the exports management section. 2. The Admin selects an export from the list. 3. The system retrieves and displays detailed information about the selected export. 4. The Admin reviews the export details, including metadata such as date, time, camera, and duration of the recording. 5. The Admin can take further actions such as downloading, renaming, or deleting the export. 		
Alternative Flows:	-		
Exception:	System error while fetching the export details (e.g., database connection issue). The system displays a generic error message indicating that there was an issue retrieving the export details and logs the error.		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	BR-01, BR-09, BR-13, BR-19		
Other Information:	-		
Assumption:	-		

Screen layout:

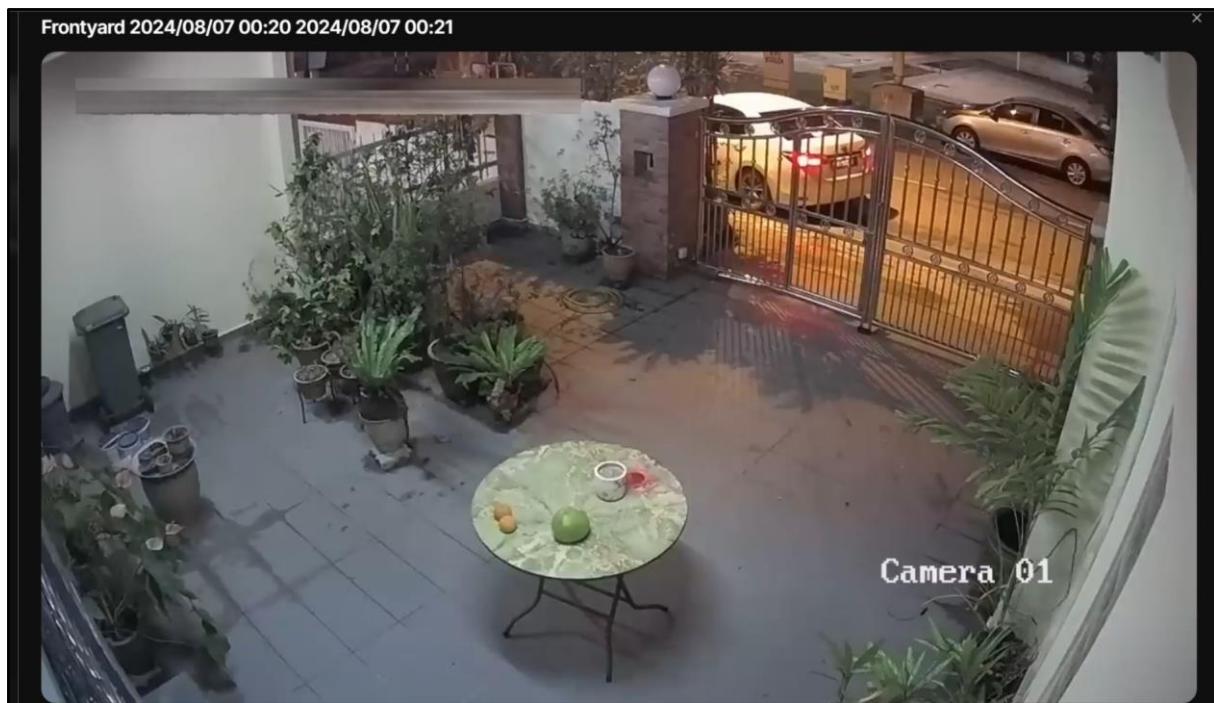


Figure 110. View export detail

2.7.3 Export event recording

Table 41. Export event recording

ID and Name:	UC-27: Export event recording		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin selects a specific event from the list of recorded events to download the recording associated with that event.		
Description:	This use case enables the Admin to download recordings of specific events captured by the Vigision surveillance system for offline storage, analysis, or sharing.		
Pre-conditions:	1. Admin is logged in. 2. Recorded events must be available in the system, and recordings must be associated with the selected event.		
Post-conditions:	The Admin has successfully downloaded the recording associated with the selected event.		
Normal Flow:	1. Admin navigates to the "Events" section within the Vigision application. 2. The application presents a list of recorded events, sorted by date and time.		

	<p>3. Admin selects a specific event from the list to download the recording.</p> <p>4. The application displays the recording associated with the selected event.</p> <p>5. Admin initiates the download process for the recording.</p> <p>6. The application prompts the user to choose a location for saving the downloaded recording.</p> <p>7. Admin confirms the download, and the recording is saved to the specified location on their device.</p>
Alternative Flows:	<p>1. If the selected event has no associated recording, the application notifies the user accordingly.</p> <p>2. If the Admin encounters any issues during the download process (e.g., interrupted connection), the application provides an error message and suggests alternative actions.</p>
Exception:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-08, BR-09, BR-12, BR-17, BR-19, BR-22, BR-23, BR-24
Other Information:	-
Assumption:	-

Screen layout:

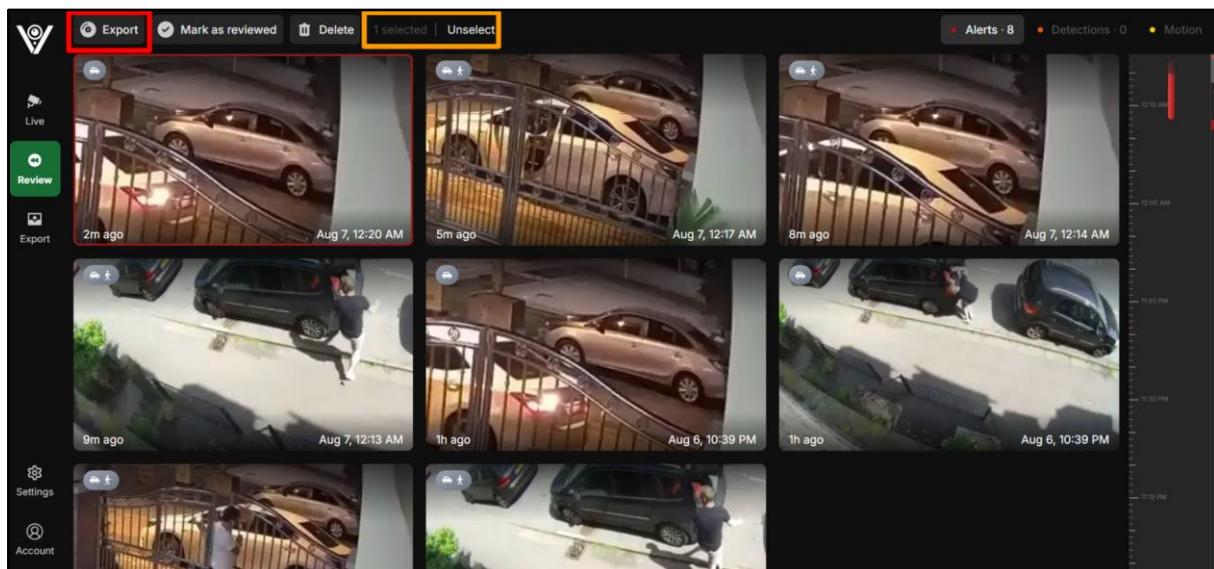


Figure 111. Export event recording

2.7.4 Export camera recording

Table 42. Export camera recording

ID and Name:	UC-28: Export camera recording		
Created By:	Vo Minh Dat	Date Created:	26/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to export a camera recording from the camera management or live dashboard section.		
Description:	This use case allows the Admin to export the recording of a specific camera. Exporting camera recordings helps users save and share significant surveillance data for further analysis, reporting, or evidence. The export process involves selecting the camera, configuring export settings, and saving the recording in the desired format.		
Pre-conditions:	The Admin must be logged into the system and have the necessary permissions to export camera recordings.		
Post-conditions:	The selected camera recording is exported and saved in the desired format.		
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the camera management or live dashboard section. 2. The Admin selects a camera from the list. 3. The Admin clicks on the "Export" button. 4. The system displays export settings such as file format and destination. 5. The Admin configures the export settings and clicks the "Export" button. 6. The system processes the export and saves the recording in the desired format. 7. The system displays a success message indicating that the camera recording has been exported. 8. The exported recording appears in the list of available exports. 		
Alternative Flows:	-		
Exception:	-		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	BR-01, BR-09, BR-12, BR-13, BR-19, BR-22, BR-23		

Other Information:	-
Assumption:	-

Screen layout:

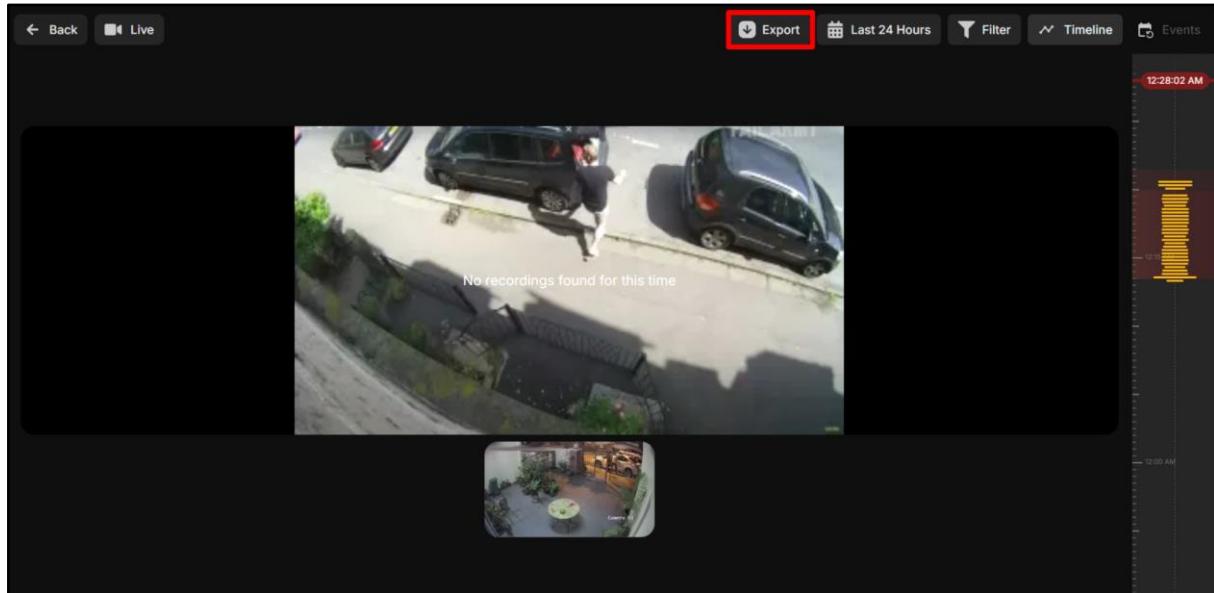


Figure 112. Export camera recording

2.7.5 Download export

Table 43. Download export

ID and Name:	UC-29: Download export		
Created By:	Vo Minh Dat	Date Created:	26/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin initiates the process to download a recording from the Vigision system.		
Description:	The Admin downloads previously recorded video footage from one or more cameras. This includes selecting the desired recordings, specifying from when the recording started, and initiating the download process.		
Pre-conditions:	1. Admin must be signed in. 2. Recordings must be available and stored in the Vigision system.		
Post-conditions:	1. The selected recordings are downloaded to the user's device. 2. The Admin can access and view the downloaded recordings on their device.		

Normal Flow:	<ol style="list-style-type: none"> 1. Admin navigates to the "Live" section in the Vigision app. 2. System displays a list of available recordings. 3. Admin clicks on the recording-screen thumbnail. 4. Admin clicks on button "Export". 5. System prompts user to specify from when the recording started (e.g., Last Hours, Last 4 Hours) and name the recording. 6. Admin clicks on button "Export". 7. System initiates the download process. 8. System displays a notification indicating the download process is started and download folder. 9. Admin receives the exported recording.
Alternative Flows:	-
Exception:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-09, BR-12, BR-13, BR-19, BR-22, BR-23
Other Information:	-
Assumption:	-

Screen layout:

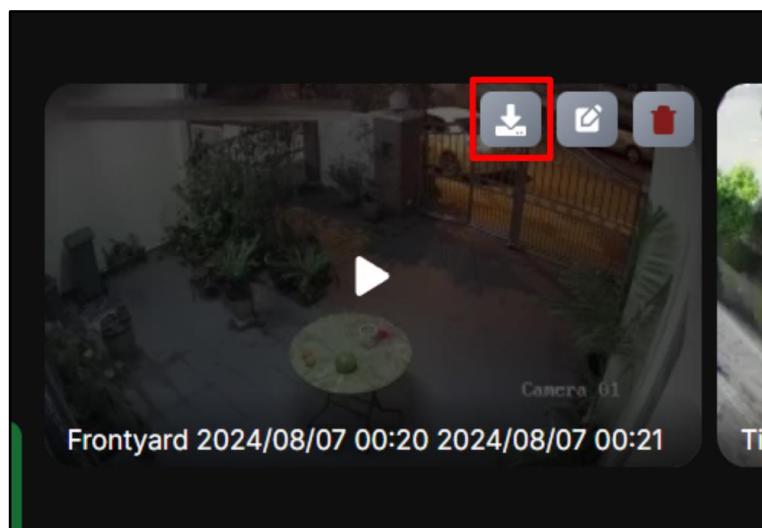


Figure 113. Download export

2.7.6 Rename export

Table 44. Rename export

ID and Name:	UC-30: Rename export		
Created By:	Vo Minh Dat	Date Created:	26/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to rename an export from the exports management section.		
Description:	This use case allows the Admin to rename an existing export file. Renaming exports helps users organize and manage their surveillance data more effectively. The rename process involves selecting the export and entering the new name for the file.		
Pre-conditions:	The Admin must be logged into the system		
Post-conditions:	The selected export file is renamed.		
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the exports management section. 2. The Admin selects an export from the list. 3. The Admin clicks on the "Rename" button. 4. The system prompts the Admin to enter a new name for the export file. 5. The Admin enters the new name and confirms the rename action. 6. The system updates the name of the export file in the database. 7. The system displays a success message indicating that the export file has been renamed. 		
Alternative Flows:	-		
Exception:	-		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	BR-01, BR-09, BR-13, BR-19		
Other Information:	-		
Assumption:	-		

Screen layout:

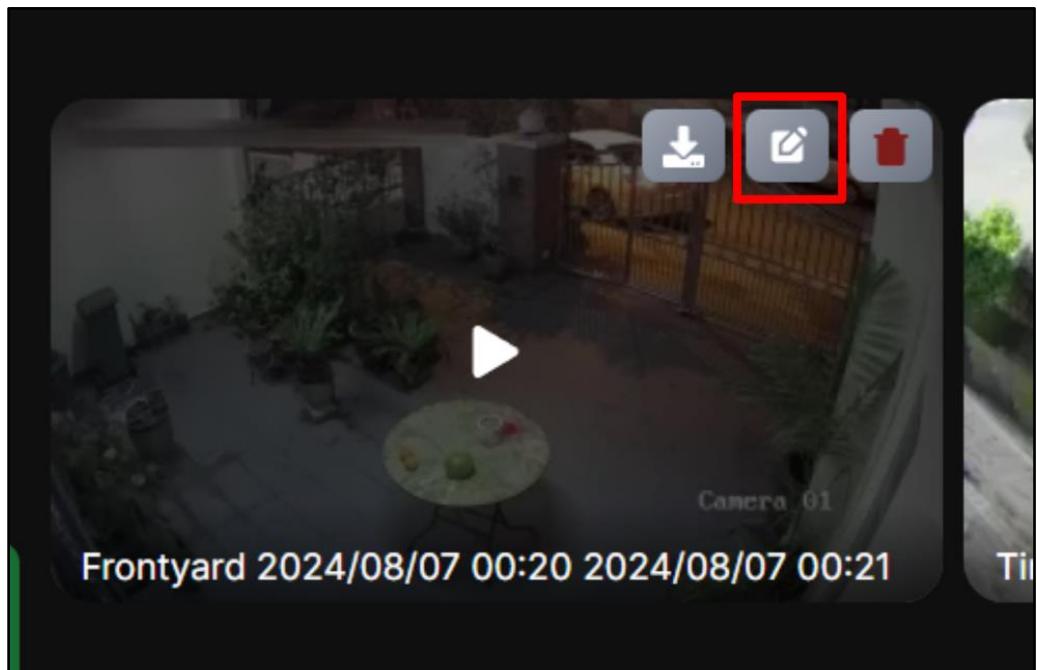


Figure 114. Rename export

2.7.7 Delete export

Table 45. Delete export

ID and Name:	UC-31: Delete export		
Created By:	Vo Minh Dat	Date Created:	26/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to delete an export from the exports management section.		
Description:	This use case allows the Admin to delete an export file that is no longer needed. Deleting unnecessary exports helps keep the export log clean and ensures that only relevant data is retained for future use.		
Pre-conditions:	The Admin must be logged into the system		

Post-conditions:	The selected export file is deleted from the system.
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin navigates to the exports management section. 2. The Admin selects an export from the list. 3. The Admin clicks on the "Delete" button. 4. The system prompts the user to confirm the deletion. 5. The Admin confirms the deletion. 6. The system deletes the export file from the database. 7. The system displays a success message indicating that the export file has been deleted.
Alternative Flows:	-
Exception:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-09, BR-13, BR-19, BR-24, BR-25
Other Information:	-
Assumption:	-

Screen layout:

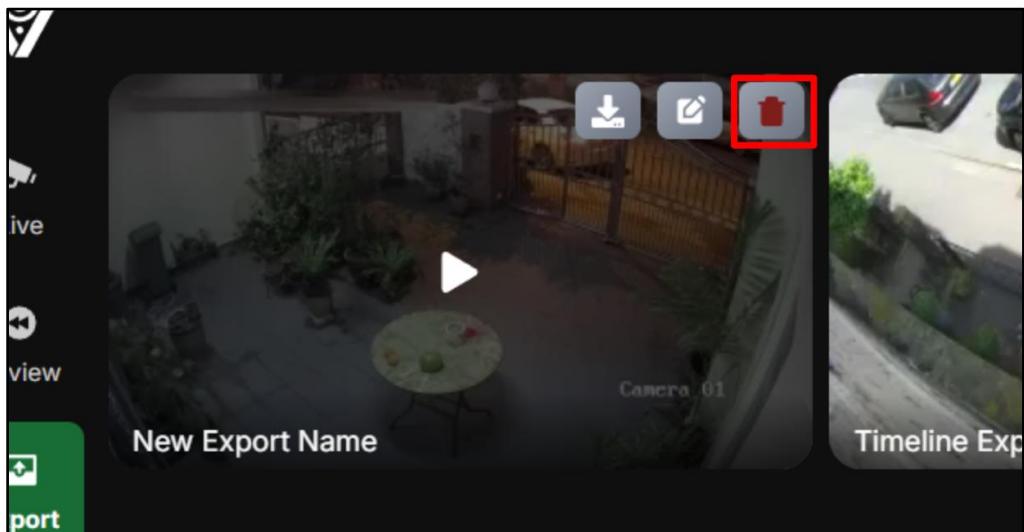


Figure 115. Delete export

2.8 Configuration

2.8.1 Config

Table 46. Config

ID and Name:	UC-32: Config		
Created By:	Vo Minh Dat	Date Created:	26/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	<p>One of the following triggers:</p> <p>View configuration: The Admin initiates the process to view or configure application settings in the Vigision system.</p> <p>Revert configuration: The Admin selects the option to revert configuration changes.</p> <p>Save configuration: The Admin clicks on the button “Save Only” in “Configuration Editor.”</p> <p>Save configuration and restart: The Admin clicks on the button “Save and Restart.”</p>		
Description:	<p>This use case allows the Admin to manage the configuration settings of the Vigision system. It includes the following actions:</p> <ul style="list-style-type: none"> • View Configuration: The Admin views the current configuration settings of the Vigision system, which includes display options, recording settings, and other configurable parameters. • Revert Configuration Change: The Admin reverts the configuration settings to their previous state, undoing any unintended or incorrect adjustments. • Save Configuration: The Admin saves the current configuration settings, ensuring that changes are applied the next time the system starts. • Save Configuration and restart: The Admin saves the current configuration settings and restarts the system to immediately apply the changes. 		
Pre-conditions:	<p>UC-32-01: View configuration 1. Admin must be signed in.</p> <p>UC-32-02: Revert configuration 1. Admin must be signed in.</p>		

	<p>UC-32-03: Save configuration 1. Admin must be signed in.</p> <p>UC-32-04: Save configuration and restart 1. Admin must be signed in. 2. The Admin have just changed the configuration</p>
Post-conditions:	<p>UC-32-01: View configuration The system configuration are shown and Admin can edit whatever they want.</p> <p>UC-32-02: Revert configuration The configuration settings are reverted to their previous state.</p> <p>UC-32-03: Save configuration The new configuration is saved and ready to be applied the next time the system starts.</p> <p>UC-32-04: Save configuration and restart The new configuration is saved, and the system is updated with the new settings after a restart.</p>
Normal Flow:	<p>UC-32-01: View Configuration</p> <ol style="list-style-type: none"> 1. Admin navigates to the "Configuration" section in the Vigision app. 2. Admin selects "Configure App Settings." 3. Admin displays a list of configurable settings. 4. User adjusts the settings as desired (e.g., camera setting, preferences, display options, recording settings). <p>UC-32-02: Revert Configuration Change</p> <ol style="list-style-type: none"> 1. The Admin navigates to the configuration section via the settings menu. 2. The Admin clicks on the "Revert Changes" button. 3. The system prompts the Admin to confirm the revert action. 4. The Admin confirms the revert action. 5. The system reverts the configuration settings to their previous state. 6. The system displays a success message indicating that the configuration changes have been reverted. <p>UC-32-03: Save Configuration</p> <ol style="list-style-type: none"> 1. Admin navigates to the "Configuration" section in the Vigision app. 2. Admin selects "Configuration Editor" button. 3. System displays a text file of configuration, some buttons to manage configurations. 4. The user clicks on the button "Save Only". 5. The configuration is saved and ready to be applied the next time the system starts.

	UC-32-04: Save Configuration and Restart
	<ol style="list-style-type: none"> 1. Admin selects "Save Configurations and Restart." 2. System displays a confirmation prompt to save changes and restart. 3. Admin confirms the save and restart action. 4. System saves the new configuration settings. 5. System initiates a restart. 6. System restarts and loads with the new settings applied.
Alternative Flows:	-
Exception:	-
Priority:	Medium
Frequency of Use:	Low
Business Rules:	BR-01, BR-18, BR-19, BR-28, BR-29
Other Information:	-
Assumption:	-

Screen layout:

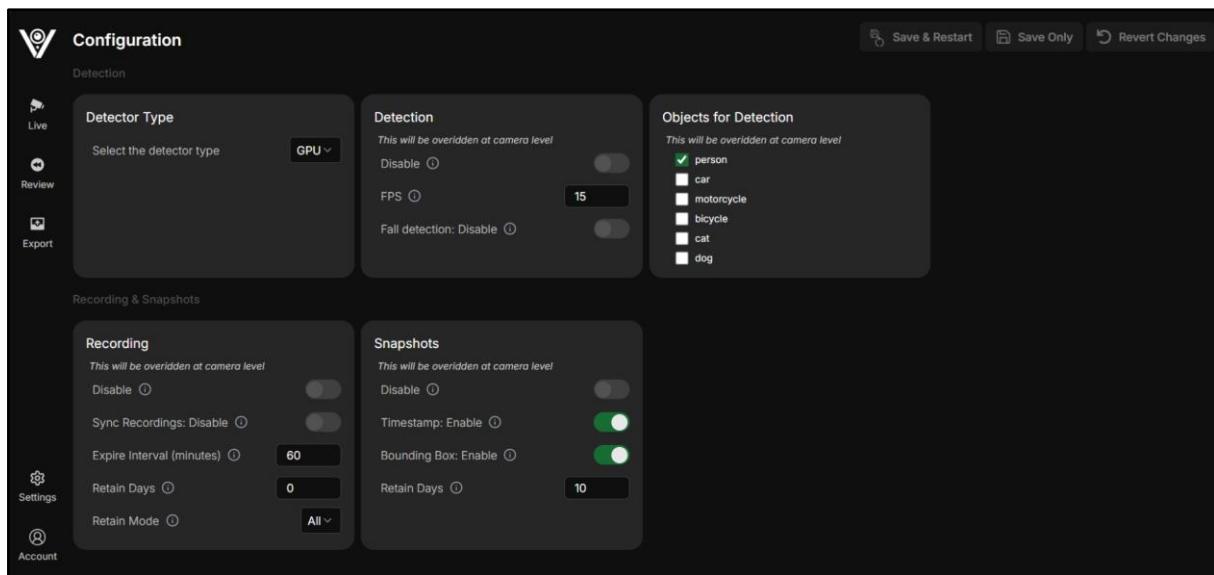


Figure 116. View configuration

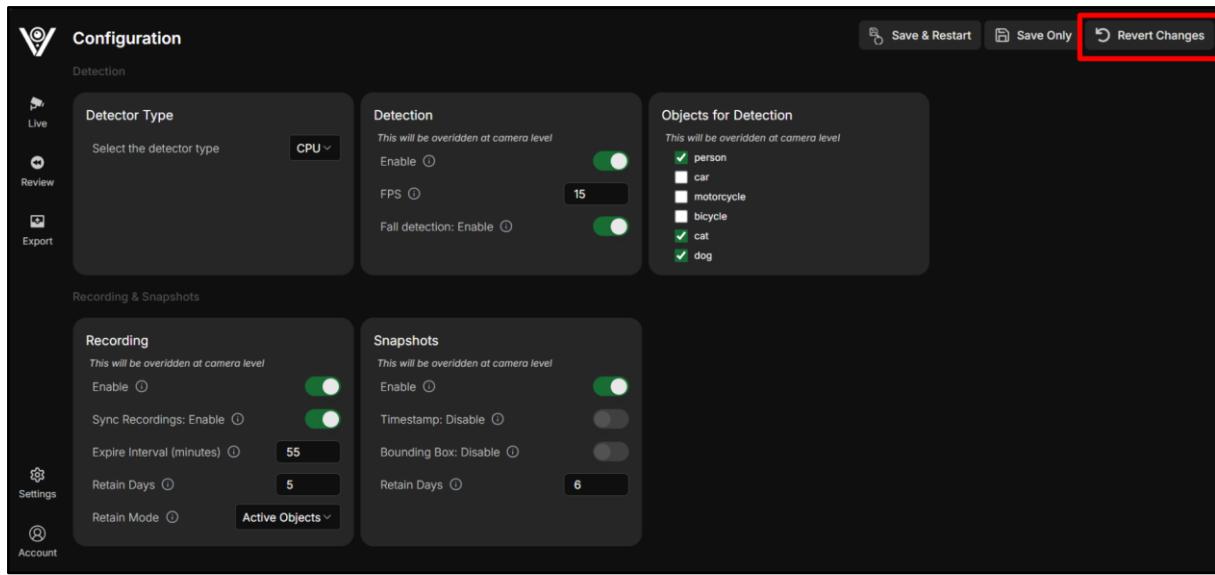


Figure 117. Revert configuration change

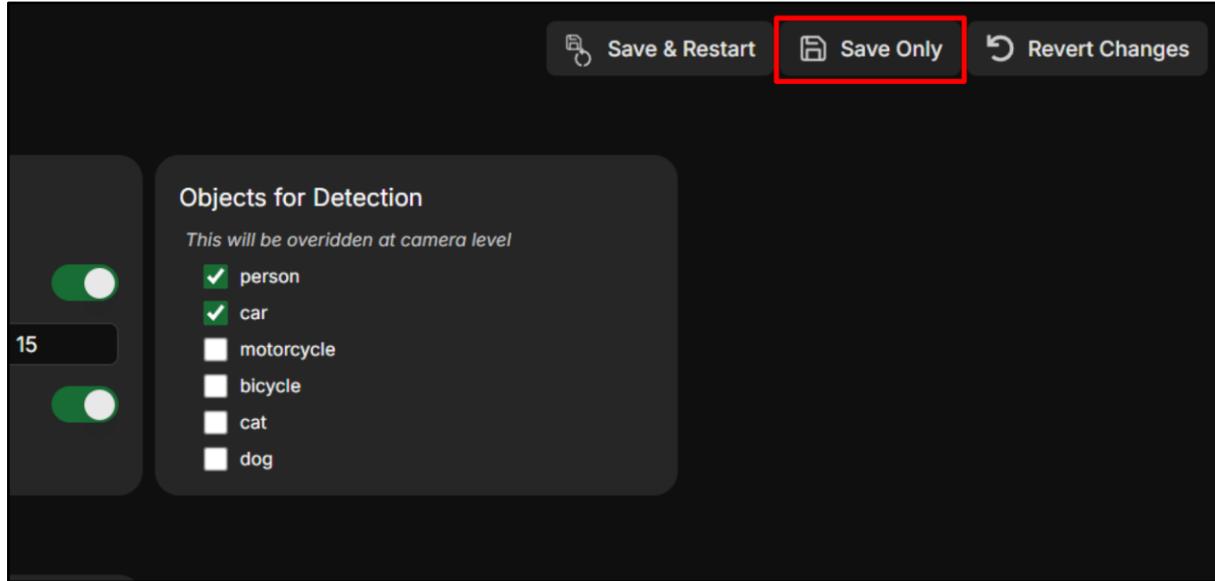


Figure 118. Save configuration

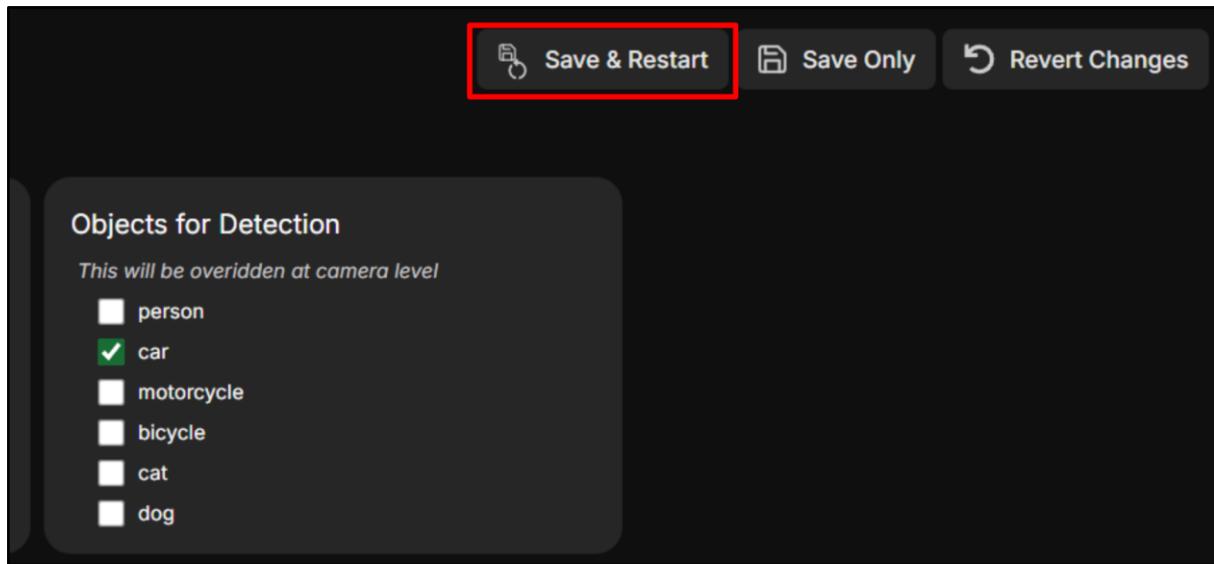


Figure 119. Save configuration and restart

2.9 Statistics Dashboard

2.9.1 View system metrics

Table 47. View system metrics

ID and Name:	UC-33: View system metrics		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin, Member	Secondary Actors:	-
Trigger:	Admin/Member accesses the system metrics section within the Vigision application.		
Description:	This use case allows the Admin/Member to monitor various system performance metrics and storage information to ensure optimal operation of the Vigision surveillance system.		
Pre-conditions:	The actor is logged in.		
Post-conditions:	The Admin/Member has successfully viewed the system metrics and storage information.		
Normal Flow:	1. The Admin/Member navigates to the "System Metrics" section within the Vigision application. 2. The application presents a dashboard or interface displaying various metrics such as CPU usage, memory utilization, network bandwidth, and storage capacity. 3. The Admin/Member selects specific metrics or categories to view detailed information and trends. 4. The application updates the displayed metrics in real-time or with periodic refresh intervals.		

Alternative Flows:	If the Admin/Member encounters issues accessing or viewing system metrics, they may troubleshoot by checking network connectivity, system resource availability, or application status.
Exception:	-
Priority:	High
Frequency of Use:	High
Business Rules:	BR-01, BR-09, BR-14, BR-19, BR-27
Other Information:	-
Assumption:	-

Screen layout:

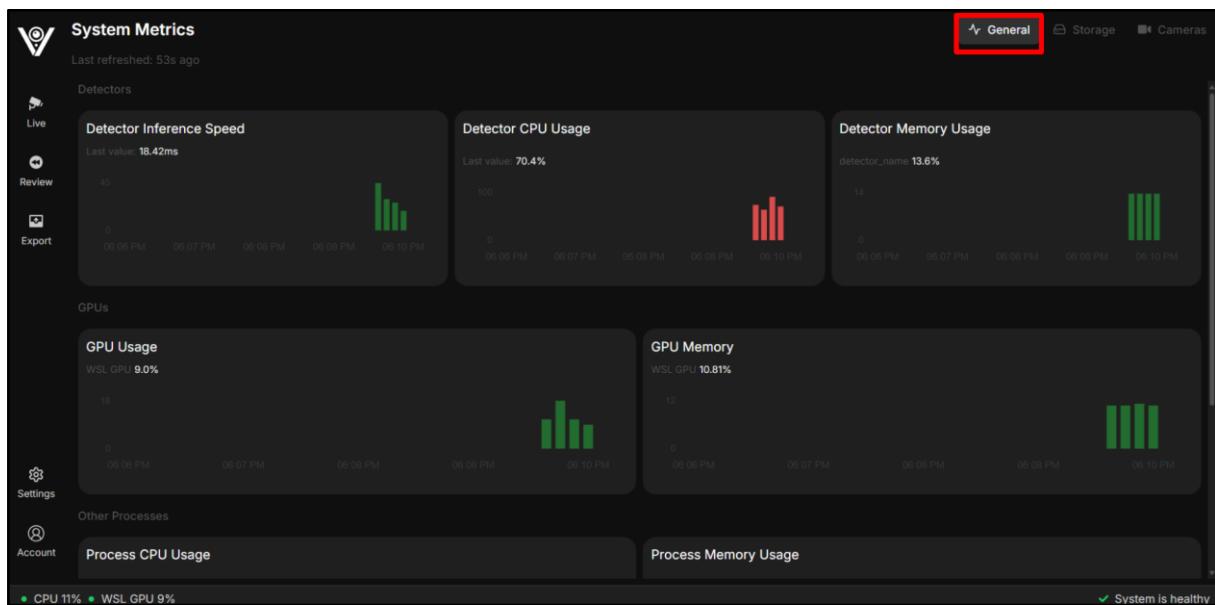


Figure 120. View system metrics

2.10 General

2.10.1 Restart application

Table 48. Restart application

ID and Name:	UC-34: Restart application		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	Admin initiates the restart process through the Vigision application interface.		
Description:	This use case involves restarting the Vigision application to apply changes, updates, or resolve issues without interrupting the overall surveillance system operation.		
Pre-conditions:	The Admin is logged in.		
Post-conditions:	The Vigision application restarts successfully and resumes normal operation.		
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin accesses the “Settings” section within the Vigision application. 2. The Admin selects the option to restart the application through a graphical user interface button. 3. The application prompts the Admin to confirm the restart action. 4. Upon confirmation, the application gracefully shuts down all active processes and services. 5. The application automatically restarts after the shutdown process completes. 		
Alternative Flows:	If there are pending recordings or active surveillance tasks, the application may prompt the Admin to confirm whether to proceed with the restart or delay until tasks are completed.		
Exception:	-		
Priority:	Medium		
Frequency of Use:	Infrequent		
Business Rules:	BR-01, BR-19, BR-28, BR-29, BR-30		
Other Information:	-		
Assumption:	-		

Screen layout:

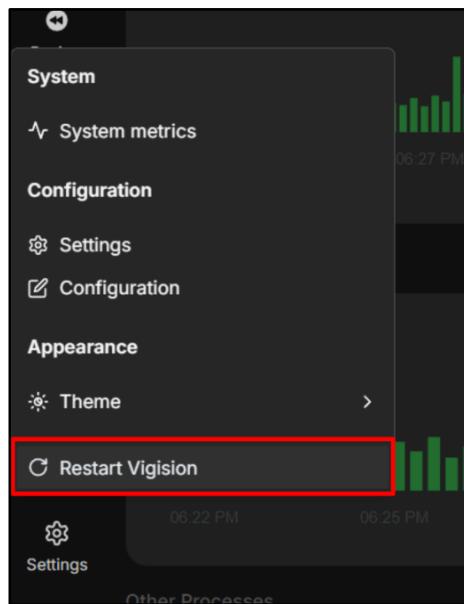


Figure 121. Restart application

2.10.2 Change application theme

Table 49. Change application theme

ID and Name:	UC-35: Change application theme		
Created By:	Dinh The Anh	Date Created:	25/05/2024
Primary Actor:	Admin, Member	Secondary Actors:	-
Trigger:	The Admin/Member interacts with the application's settings interface to toggle between light and dark mode.		
Description:	This use case involves allowing the user to switch between light and dark mode themes in the Vigision application interface for improved visibility and personal preference.		
Pre-conditions:	1. The Admin/Member is logged in. 2. The application must support both light and dark mode themes.		
Post-conditions:	The application interface switches to the selected mode (light or dark), reflecting the actor's preference.		
Normal Flow:	1. The Admin/Member accesses the "Settings" section within the Vigision application. 2. The Admin/Member navigates to the "Dark Mode" option. 3. The Admin/Member selects the desired theme mode (light or dark). 4. The application interface switches to the selected theme mode.		
Alternative Flows:	If the application does not support dark mode, the option to toggle		

	between light and dark mode may be disabled or hidden from the settings interface.
Exception:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-30, BR-31
Other Information:	-
Assumption:	-

Screen layout:

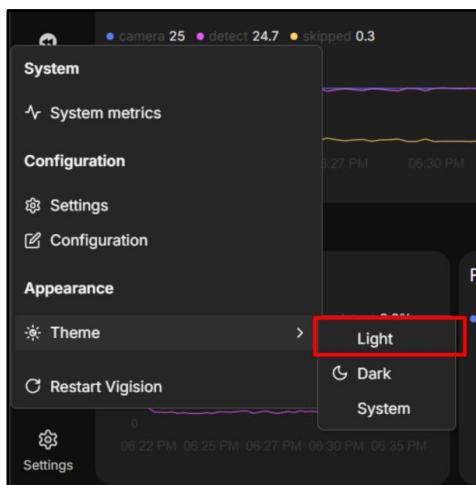


Figure 122. Change application theme

2.10.3 Get shareable link

Table 50. Get shareable link

ID and Name:	UC-06: Get shareable link		
Created By:	Nguyen Le Quang Thinh	Date Created:	25/05/2024
Primary Actor:	Admin	Secondary Actors:	-
Trigger:	The Admin selects the option to get a shareable link from the account settings menu.		
Description:	This use case allows the Admin to generate a shareable link for a camera		

	feed. This link can be shared with other users to provide them with access to the live camera feed without requiring them to log into the system.
Pre-conditions:	The Admin must be logged into the system.
Post-conditions:	A shareable link is generated and displayed to the Admin.
Normal Flow:	<ol style="list-style-type: none"> 1. The Admin clicks on the "Account" button. 2. A menu with account options is displayed. 3. The Admin clicks on the "Get Shareable Link" option. 4. The system generates a shareable link for the camera feed. 5. The system displays the shareable link in a pop-up window. 6. The Admin clicks on the "Copy" button to copy the link to the clipboard. 7. The system displays a success message indicating that the link has been copied. 8. The Admin clicks on the "Done" button to close the pop-up window.
Alternative Flows:	-
Exception:	-
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	BR-01, BR-09, BR-19
Other Information:	-
Assumption:	-

Screen layout:

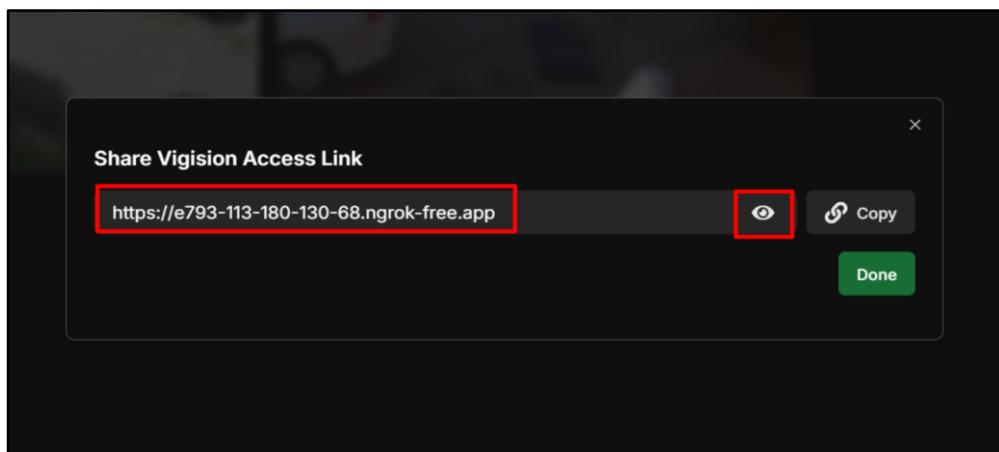


Figure 123. Get shareable link

3. Functional Requirements

3.1 System Functional Overview

3.1.1 Screens Flow

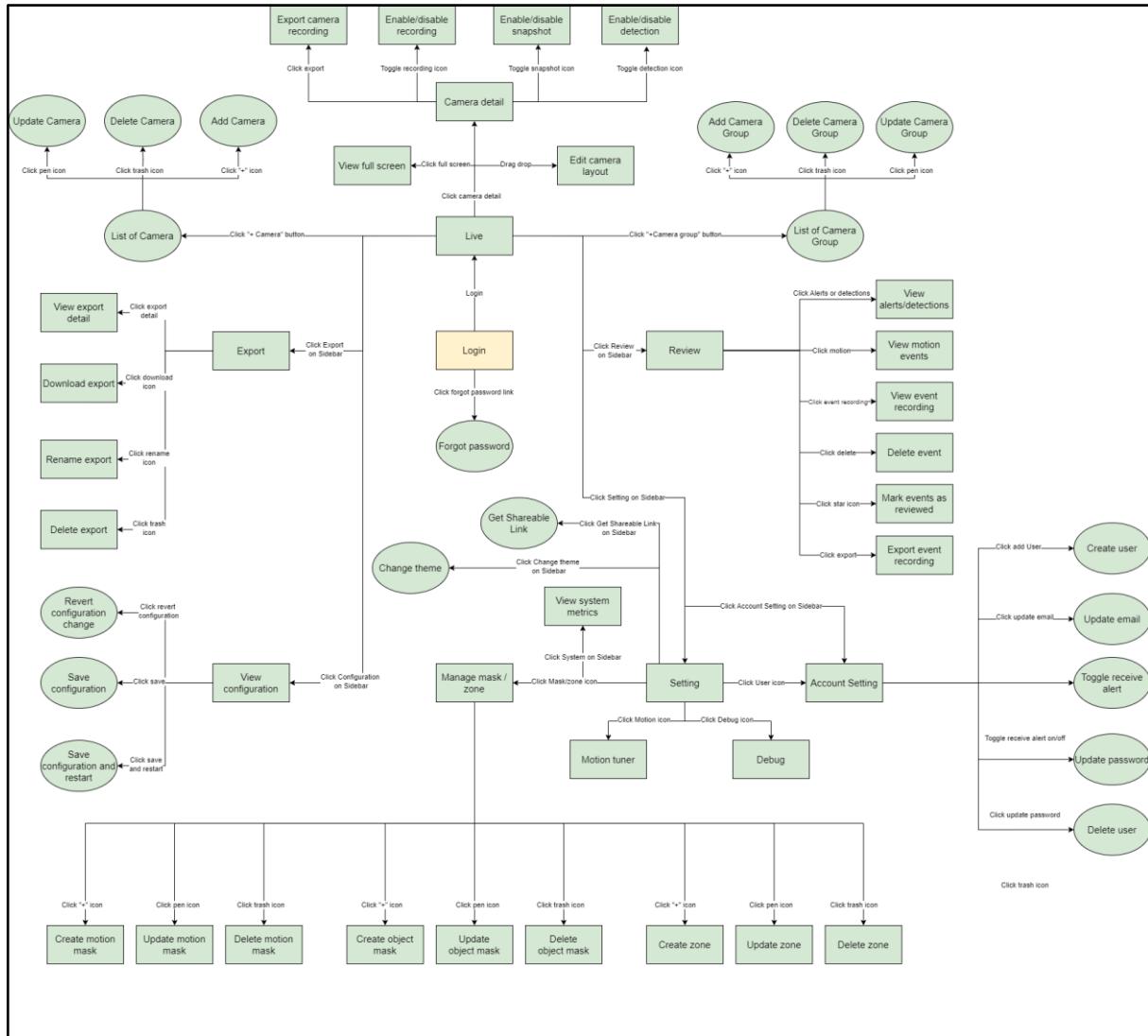


Figure 124. Admin screen flow

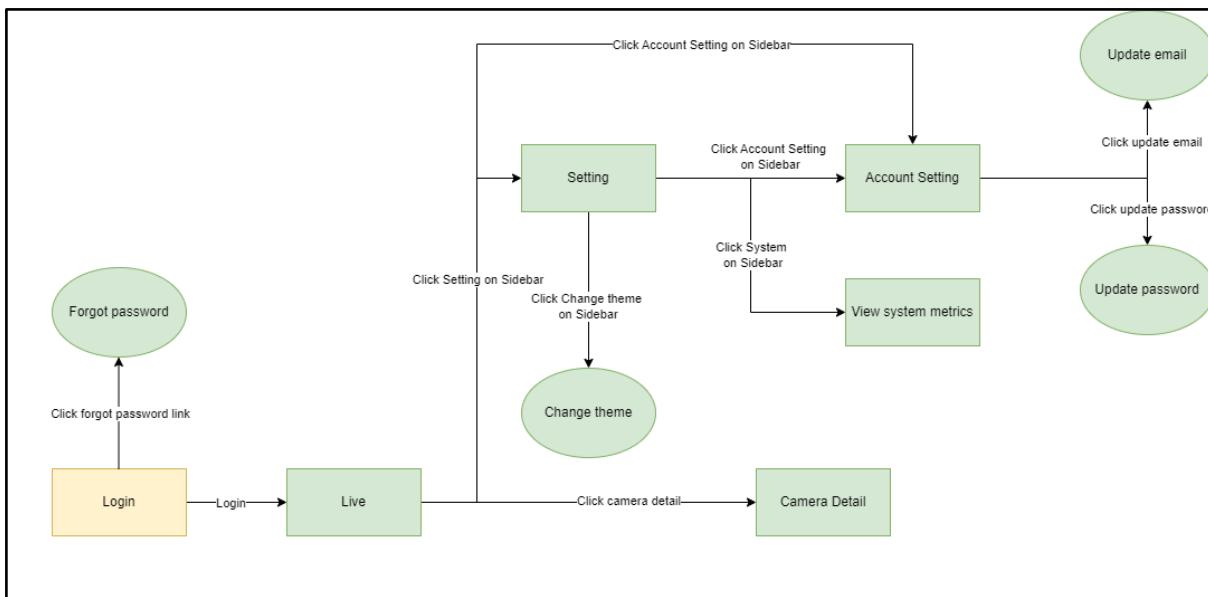


Figure 125. Member screen flow

3.1.2 Screen Descriptions

Table 51. Screen Descriptions

ID	Feature	Screen	Screen description
01	Login	Login	The login screen allows the Admin/Member to enter their credentials (username and password) to access the Vigision system. The screen includes fields for username, password, and a "Login" button.
02	Manage Users	Account Setting	This screen displays a list of all users with options to add, update, and delete users. It includes buttons for "Create User," "Update Email," "Update Password," "Delete User," and "Toggle Receive Alert.". A Member can only update email, update password on this screen.
03	Get Shareable Link	Setting	This screen generates a shareable link for camera feeds. It includes an input field to select the camera and a "Generate Link" button.
04	View Live Camera	Live	The live dashboard displays real-time video feeds from all configured cameras. It includes options to view

			camera details, enable/disable detection, recording, snapshots, and switch to full screen.
05	Manage Cameras	Live	This screen lists all cameras with options to add, update, and delete cameras. It includes buttons for "Add Camera," "Update Camera," and "Delete Camera."
06	Manage Camera Groups	List of Camera Group	This screen lists all camera groups with options to add, update, and delete camera groups. It includes buttons for "Add Camera Group," "Update Camera Group," and "Delete Camera Group."
07	Edit Camera Group Layout	Edit camera layout	This screen allows the Admin to edit the layout of camera groups by dragging and dropping camera feeds into different positions. It includes a "Save Layout" button.
08	View in Full Screen	View full screen	This screen displays a single camera feed in full screen mode for detailed monitoring. It includes an option to exit full screen.
09	View Detail of Live Camera	Camera detail	This screen provides detailed information and live feed of a selected camera. It includes options to enable/disable detection, recording, snapshots, and export recordings.
10	Enable/Disable Detection	Enable/Disable Detection	This screen allows the Admin to toggle detection features for a specific camera. It includes a switch to enable or disable detection.
11	Enable/Disable Recording	Enable/Disable Recording	This screen allows the Admin to toggle recording features for a specific camera. It includes a switch to enable or disable recording.

12	Enable/Disable Snapshots	Enable/Disable Snapshots	This screen allows the Admin to toggle snapshot features for a specific camera. It includes a switch to enable or disable snapshots.
13	View Defined Masks and Zones	Manage mask / zone	This screen displays the motion masks and zones defined for a specific camera. It includes options to create, update, and delete masks and zones.
14	Manage Motion Masks	Manage mask / zone	This screen allows the Admin to create, update, and delete motion masks for a specific camera. It includes a list of existing masks and buttons for "Add Mask," "Edit Mask," and "Delete Mask."
15	Manage Zones	Manage mask / zone	This screen allows the Admin to create, update, and delete zones for a specific camera. It includes a list of existing zones and buttons for "Add Zone," "Edit Zone," and "Delete Zone."
16	Manage Object Masks	Manage mask / zone	This screen allows the Admin to create, update, and delete object masks for a specific camera. It includes a list of existing masks and buttons for "Add Object Mask," "Edit Object Mask," and "Delete Object Mask."
17	Fine-tune Motion Detection	Motion Tuner	This screen allows the Admin to adjust the sensitivity and other parameters of motion detection for a specific camera. It includes sliders and input fields for fine-tuning settings.
18	Debug	Debug	This screen provides debugging tools and options to analyze and troubleshoot camera performance. It includes options to display bounding boxes, motion masks, and other diagnostic information.

19	View Alerts/Detections	View Alerts/Detections	This screen displays a list of alerts and detections recorded by the system. It includes details such as date, time, camera, and type of alert/detection.
20	View Motion Events	View Motion Events	This screen displays a list of motion events recorded by the system. It includes details such as date, time, camera, and duration of motion.
21	View Event Recording	View Event Recording	This screen allows the Admin to view the recorded video of a specific event. It includes playback controls and options to export the recording.
22	Mark Events as Reviewed	Mark Events as Reviewed	This screen allows the Admin to mark one or multiple events as reviewed. It includes checkboxes to select events and a "Mark as Reviewed" button.
23	Delete Events	Delete Events	This screen allows the Admin to delete one or multiple events. It includes checkboxes to select events and a "Delete" button.
24	View Exports	View Exports	This screen displays a list of all exports available in the system. It includes details such as date, time, camera, and type of export.
25	View Export Detail	View Export Detail	This screen provides detailed information about a specific export. It includes metadata such as date, time, camera, and duration of the recording.
26	Export Event Recording	Export Event Recording	This screen allows the Admin to export the recording of a specific event. It includes options to select the event, configure export settings, and save the recording.
27	Export Camera Recording	Export Camera Recording	This screen allows the Admin to export the recording of a specific camera. It includes options to select the camera, configure export settings, and save the recording.

28	Download Export	Download Export	This screen allows the Admin to download an export file to their local device. It includes options to select the export and save the file to the desired location.
29	Rename Export	Rename Export	This screen allows the Admin to rename an existing export file. It includes an input field to enter the new name and a "Rename" button.
30	Delete Export	Delete Export	This screen allows the Admin to delete an export file that is no longer needed. It includes options to select the export and a "Delete" button.
31	View configuration	View configuration	This screen encompasses all actions related to viewing, reverting, saving, and restarting the configuration settings of the Vigision system. It includes buttons for "View Configuration," "Revert Changes," "Save Configuration," and "Save & Restart."
32	View System Metrics	View System Metrics	This screen displays various system metrics, including performance statistics, resource usage, and camera status. It helps users monitor the health and performance of the Vigision system.
33	Restart Application	Restart Application	This screen allows the Admin to restart the Vigision application. It includes a confirmation message and a "Restart" button.
34	Change Application Theme	Change Theme	This screen allows the user to change the theme of the Vigision application. It includes a list of available themes and options to select and apply a new theme.
35	Forgot Password	Forgot Password	The forgot password screen allows the Admin to reset their password. It includes a field to enter the registered email address and a "Submit" button. A link to reset the password is sent to the entered email address.

3.1.3 Screen Authorization

Table 52. Screen Authorization

Screen	Admin	Member
Login	X	X
Account Setting	X	X
Setting	X	
Live	X	X
Camera detail	X	X
View Masks and Zones	X	
Manage Motion Masks	X	
Manage Zones	X	
Manage Object Masks	X	
Motion Tuner	X	
Debug	X	
View Alerts/Detections	X	
View Motion Events	X	
View Event Recording	X	
Mark Events as Reviewed	X	
Delete Events	X	
View Exports	X	
View Export Detail	X	
Export Event Recording	X	
Export Camera Recording	X	
Download Export	X	

Rename Export	X	
Delete Export	X	
View configuration	X	
View System Metrics	X	X
Change Theme	X	X
Forgot Password	X	X

3.1.4 Non-Screen Functions

Table 53. Non-Screen Functions

#	Feature	System Function	Description
1	Real-time AI Computer Vision Features	Motion Detection	Detects motion in the camera using AI model and detect automatically, filtering significant movements to reduce false alerts.
2	Real-time AI Computer Vision Features	Human Detection	Identifies human presence in the camera using deep learning models automatically, enhancing security by distinguishing humans from other objects.
3	Real-time AI Computer Vision Features	Fall Detection	Recognizes instances of human falls in the camera automatically, providing immediate alerts to ensure prompt response in emergencies.
4	Recording Features	Auto-delete recordings, snapshots	Automatically deletes old recordings and snapshots based on predefined settings to manage storage space efficiently.
5	Recording Features	Recording, snapshot saving	Manages the recording of video and saving of snapshots, ensuring that critical surveillance data is captured and stored.
6	General features	Restart application	Restarts the Vigision application, applying any new configurations or updates and resolving any operational issues.
7	Config	Load and validate config	Loads and validates configuration settings when the application starts, ensuring that all settings are correctly applied and functional.

3.1.5 Entity Relationship Diagram

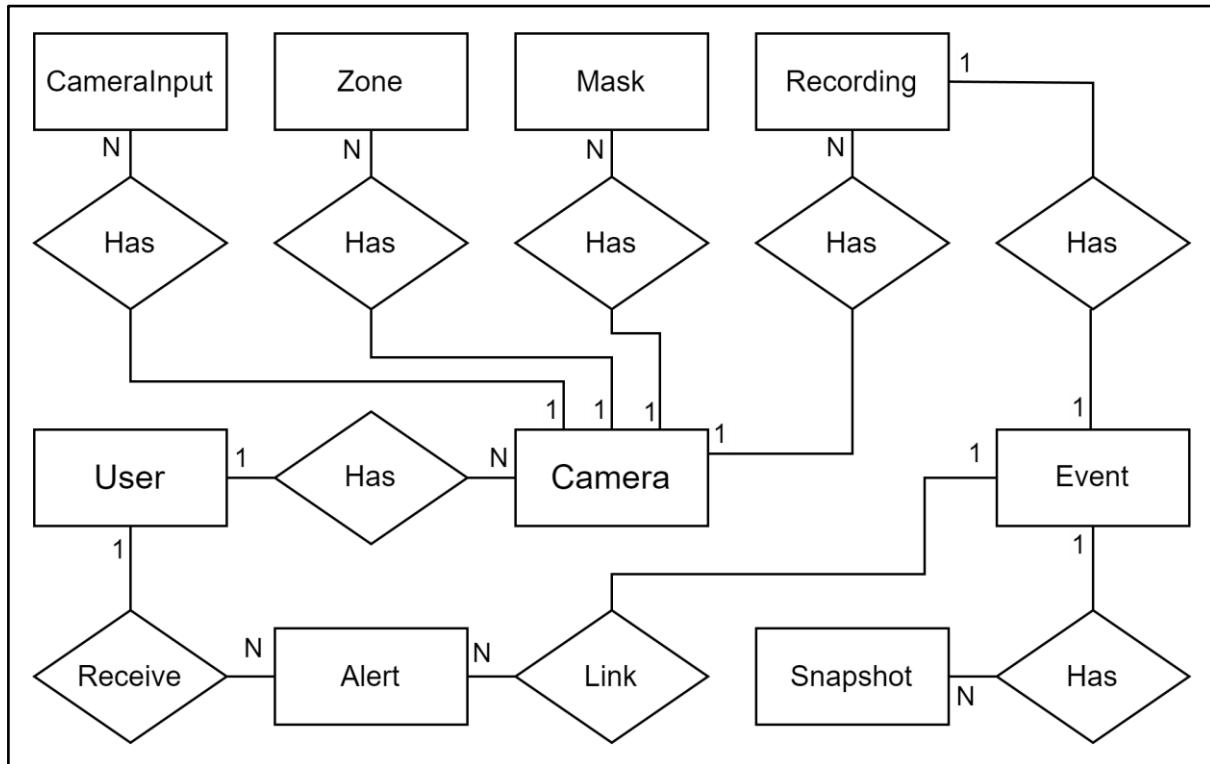


Figure 126. Entity relationship diagram

Entities List

Table 54. Entities List

#	Entity	Description
1	User	Represents an individual who interacts with the Vigision system
2	Camera	Represents a camera connected to the system, capturing video footage.
3	CameralInput	Represents the stream of data received from a connected camera.
4	Zone	Represents a predefined area within a camera's field of view, determining whether or not an object is within a particular area.
5	Mask	Represents an area within a camera's field of view that is excluded from detection or analysis.
6	Recording	Represents a saved video footage captured by a camera.
7	Event	Represents a detected incident or activity captured by the system, such as a fall, object detection, or motion.
8	Snapshot	Represents a still image captured from a camera at a specific moment, associated with an event.
9	Alert	Represents a message sent to a user via Google Mail.
10	OTP	Represents otp with corresponding email

Entities Description

3.1.5.1 User

Table 55. Entities Description - User

#	Attribute	Description
1	ID (PK)	A unique identifier for each User record in the database.
2	Username	The unique name chosen by the user to log into the system.

3	Email (FK)	The user's email address, used for account verification, password recovery, and receiving notifications.
4	Password	The user's chosen password, securely stored and hashed for security purposes.
5	CreateAt	Timestamp indicating when the user account was created.
6	UpdateAt	Timestamp indicating when the user account information was last updated.

3.1.5.2 Camera

Table 56. Entities Description - Camera

#	Attribute	Description
1	ID (PK)	A unique identifier for each Camera record.
2	UserID (FK)	Reference to which user it belongs to.
	Name	Set the name of the stream that is used for live view.
3	Enabled	Enable/Disable the camera. If disabled: config is used but no live stream, no capture, events/recording are still viewable.
4	UIOrder	Sort order of the camera in the UI.
5	Status	Status of the camera: active or inactive.
6	IsDetectEnabled	Is detection mode enabled or not.
7	DetectWidth	Width of the frame for the input with the detect role. This is optional, by default Vigision tries to automatically detect resolution.
8	DetectHeight	Height of the frame for the input with the detect role. This is optional, by default Vigision tries to automatically detect resolution.
9	FPS	Desired fps for your camera for the input with the detect role. Recommended value of 5.
10	MinInitialized	Number of consecutive detection hits required for an object to be initialized in the tracker.
11	MaxDisappeared	Number of frames without a detection before Vigision considers an object to be gone.
12	IsSnapshotEnabled	Enable writing image snapshot to local.
13	SnapshotTimestamp	Print a timestamp on the snapshots.
14	SnapshotBoundingBox	Draw a bounding box on the snapshots.
15	SnapshotHeight	Height to resize the snapshot to, default is the original size.
16	SnapshotRetainDays	Number of days snapshot will be deleted.
17	IsRecordingEnabled	Is recording mode enabled or not.
18	recording_retain_days	Number of days to retain recordings regardless of events.
19	RecordingMode	Mode for retention. Available options are: - all: save all recording segments regardless of activity. - motion: save all recordings segments with any detected motion. - active_objects: save all recording segments with active/moving objects.
20	IsEventRecordingEnabled	Is event recording mode enabled or not.
21	EventRecordingPreCapture	Number of seconds before the event to include.
22	EventRecordingPostCapture	Number of seconds after the event to include.
23	EventRecordingObjects	List of objects to save recordings for.

24	EventRecordingRetainDays	Number of retention days.
25	EventRecordingMode	Mode for retention: - all: save all recording segments for events regardless of activity. - motion: save all recordings segments for events with any detected motion. - active_objects: save all recording segments for event with active/moving objects.
26	IncludeTimestamp	Timestamp will be overlayed on the camera stream.

3.1.5.3 CameraInput

Table 57. Entities Description - CameraInput

#	Attribute	Description
1	ID (PK)	A unique identifier for each CameraInput record in the database.
2	CameraID (FK)	Reference to which camera it belongs to.
3	Path	The path to the RTSP stream.
4	Roles	List of roles for this stream.

3.1.5.4 Zone

Table 58. Entities Description - Zone

#	Attribute	Description
1	ID (PK)	A unique identifier for each Zone record in the database.
2	CameraID (FK)	Reference to which camera it belongs to.
3	Name	Name of the zone. This must be different than any camera names, but can match with another zone on another camera.
4	Coordinates	List of x,y coordinates to define the polygon of the zone.
5	Inertia	Number of consecutive frames required for object to be considered present in the zone.
6	Objects	List of objects that can trigger this zone (default: all tracked objects).

3.1.5.5 Mask

Table 59. Entities Description - Mask

#	Attribute	Description
1	ID (PK)	A unique identifier for each Mask record in the database.
2	CameraID (FK)	Reference to which camera it belongs to.
3	Name	Name of the mask. This must be different than any camera names, but can match with another mask on another camera.
4	Coordinates	List of x,y coordinates to define the polygon of the mask.
5	MaskType	Type of the mask: motion or object.

3.1.5.6 Recording

Table 60. Entities Description - Recording

#	Attribute	Description
1	ID (PK)	A unique identifier for each Recording record in the database.
2	CameraID (FK)	Reference to which camera it belongs to.
3	Type	Type of the recording: continuous or event.
4	SavePath	Local path where the recording is saved.

3.1.5.7 Event

Table 61. Entities Description - Event

#	Attribute	Description
1	ID (PK)	A unique identifier for each Event record in the database.
2	RecordingID (FK)	Reference to which Recording it belongs to.
4	Type	The type of event detected, such as "Fall Detection", "Object Detection", or "Motion Detection".
5	StartTimestamp	Timestamp indicating the start time of the event.
6	EndTimestamp	Timestamp indicating the end time of the event.
7	KeyMoments	A list of timestamps within the event recording that highlight important moments.
8	IsStarred	A boolean flag indicating whether the user has marked this event as important for easier retrieval and review.

3.1.5.8 Snapshot

Table 62. Entities Description - Snapshot

#	Attribute	Description
1	ID (PK)	A unique identifier for each Recording record in the database.
2	EventID (FK)	Reference to which Event it belongs to.
4	SavePath	Local path where the snapshot is saved.

3.1.5.9 Alert

Table 63. Entities Description - Alert

#	Attribute	Description
1	ID (PK)	A unique identifier for each Alert record in the database.
2	UserID (FK)	Reference to which User it belongs to.
3	EventID (FK)	Reference to which Event it belongs to.
4	Type	The type of alert.
5	Message	The text message content of the alert.
6	CreatedAt	Timestamp indicating when the alert was generated and sent to the user.

4. Non-Functional Requirements

4.1 External Interfaces

4.2.1 User interfaces

- UI-1: The web interface should be responsive and adapt to different screen sizes, ensuring a seamless user experience across desktops, laptops, and mobile devices.
- UI-2: The interface should be organized logically, with clear menus, buttons, and icons to guide users through the system.
- UI-3: Users should be able to view live streams from all connected cameras simultaneously or individually.
- UI-4: The web interface should be compatible with major web browsers, including Chrome, Firefox, Safari, and Edge.

4.2.2 Software Interfaces

- SI-1: Interfaces with various IP cameras using standard protocols (RTSP) for streaming and control.
- SI-2: Vigision will be packaged as a Docker container for easy deployment and self-hosting on local hardware devices.

- SI-3: Interfaces for configuring and managing AI models.
- SI-4: Interfaces with the database for storing user data, configurations, and other data.
- SI-5: Interfaces for sending alerts and notifications via SMS.

4.2.2 Communication Interfaces

- CI-1: Require a stable internet connection for remote access and live streaming.
- CI-2: Utilize WebSockets for real-time data streaming and updates, providing a seamless user experience.
- CI-3: Provide RESTful endpoints for external integrations, enabling extensibility and automation.

4.2 Quality Attributes

4.2.1 Usability

- The user interface should be intuitive and easy to navigate for both novice and experienced users.
- All information presented to the user should be clear, concise, and easy to understand.

4.2.2 Reliability

- The system should be designed for high availability, ensuring minimal downtime and continuous operation.
- All data collected and processed by Vigision should be accurate and reliable.
- The system should be resilient to failures and able to recover gracefully from unexpected events.

4.2.3 Performance

- AI algorithms should process video streams in real-time to provide timely alerts and analysis.
- Video streaming should have minimal latency to ensure a smooth and responsive user experience.
- The system should be optimized for efficient resource utilization, minimizing system load and power consumption.

4.2.4 Security

- All data transmitted and stored by Vigision should be encrypted to protect user privacy and sensitive information.
- Secure authentication mechanisms should be implemented to control user access and prevent unauthorized access.

4.2.5 Supportability

- Detailed documentation should be provided for users and developers.
- Users should have access to technical support channels for assistance with installation, configuration, and troubleshooting.
- Regular software updates should be provided to improve functionality, address bugs, and enhance security.

4.2.6 Design Constraints

- The system should be designed to scale to accommodate a growing number of users, cameras, and data.
- The system should be cost-effective to deploy and maintain.

IV. Software Design Description

1 System Design

1.1 System Architecture

The system architecture is designed to handle surveillance data efficiently, featuring local cameras streaming video to a Flask-based backend, which processes and stores data in an SQLite local database. The backend communicates with a React and TypeScript-based web application via WebSocket and API, providing real-time updates to clients. Users interact with the system through a web application deployed in Docker for scalability, and receive notifications via integrated Gmail. This architecture ensures seamless data flow, real-time monitoring, and efficient user notification management.

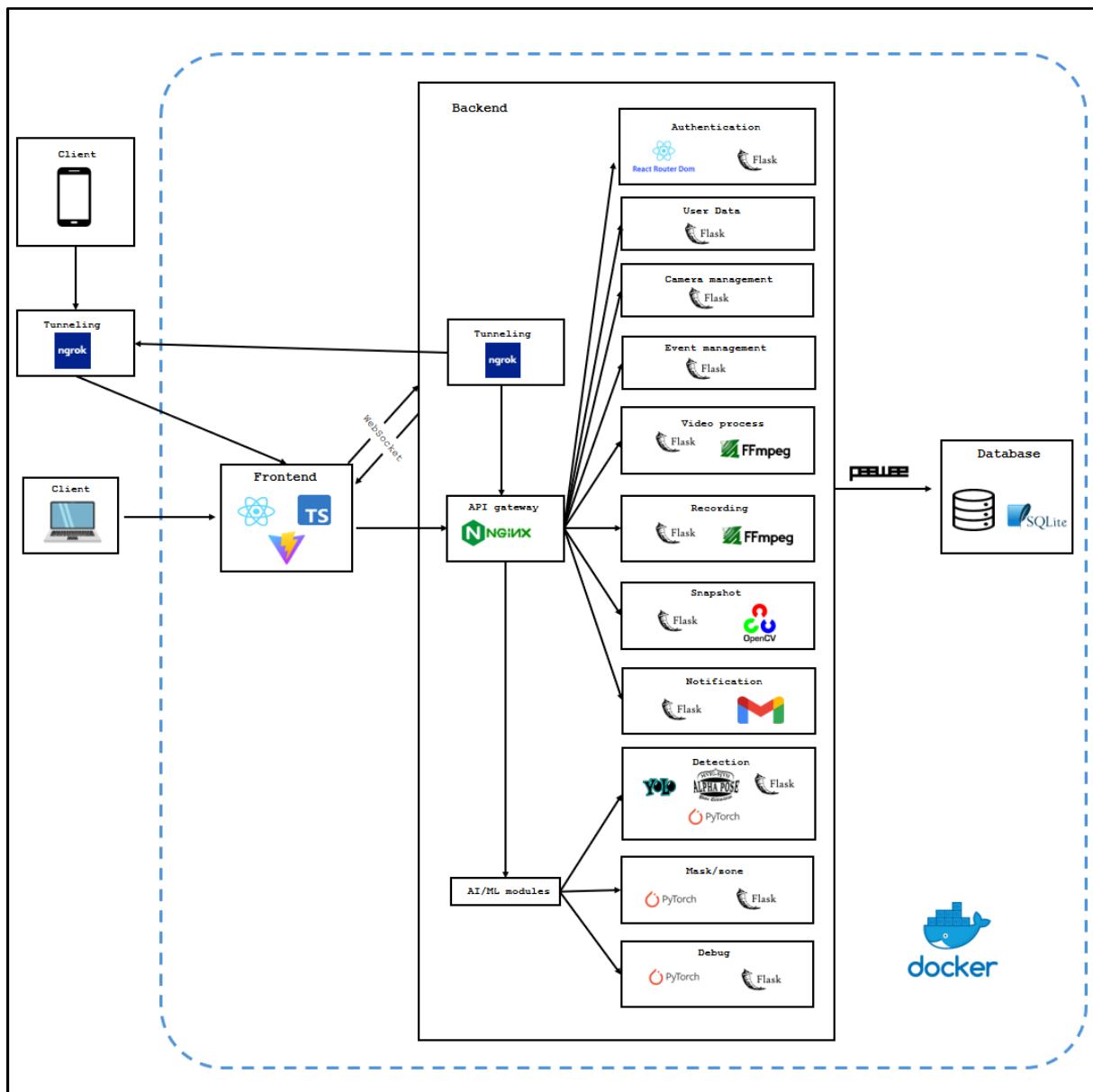


Figure 127. System architecture

1.2 Package Diagram

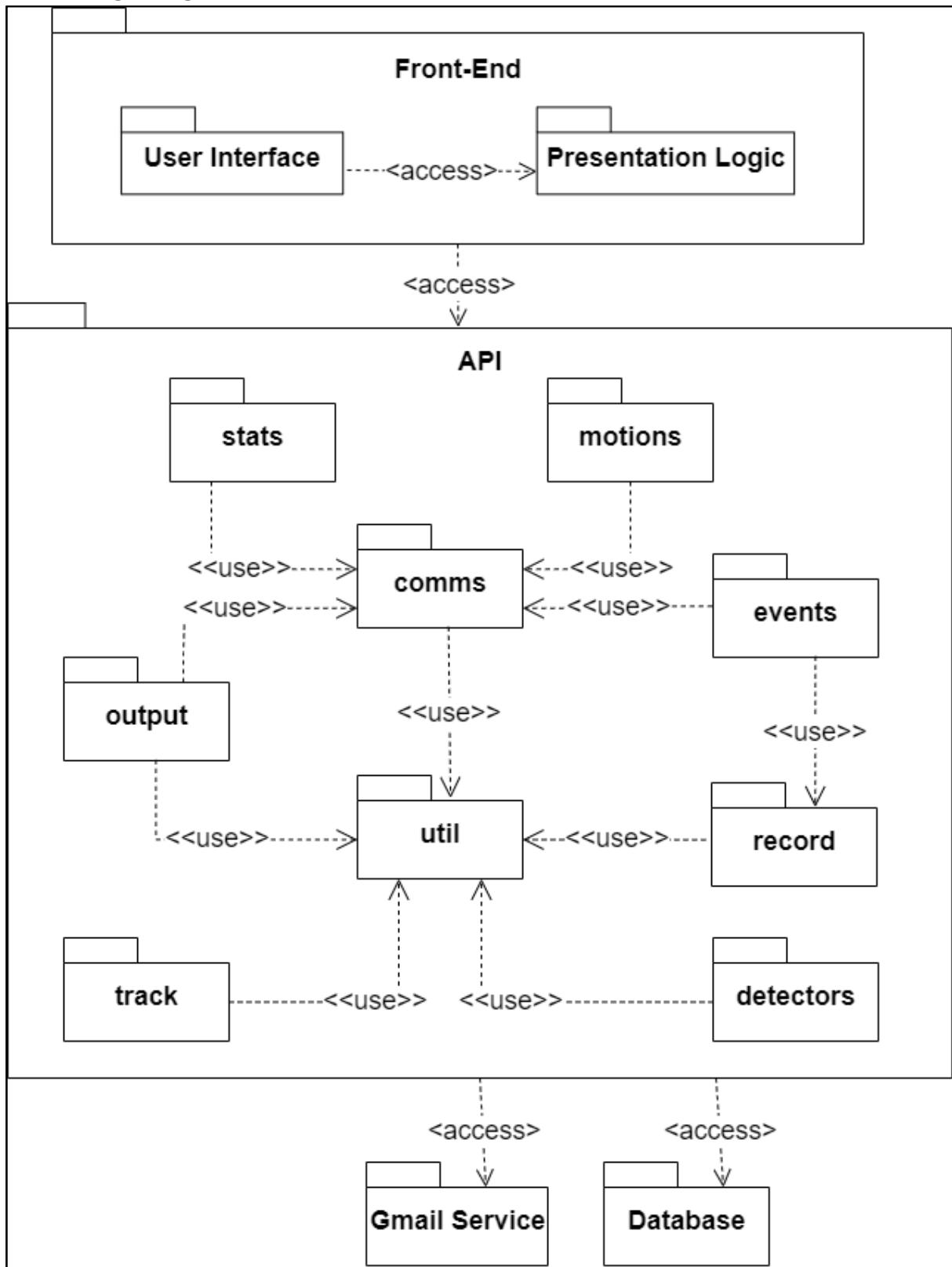


Figure 128. Package diagram

Package Descriptions

Table 64. Package Descriptions

No	Package	Description
01	Front-end	Covers both the operational logic and the user interface elements that ensure the system works as intended.
02	User Interface	All the visual aspects and interactive elements of the system.
03	Presentation Logic	Programming logic responsible for how the user interface functions.
04	API	Handles communication between the front-end and other system components through requests and responses.
05	comms	Facilitates inter-process communication through various mechanisms, including ZeroMQ and WebSockets. It includes classes for publishing and subscribing to configuration changes, handling video detections, and managing event updates. It also supports proxying messages and inter-process communication with both request-response and pub-sub models. The package ensures efficient and flexible communication between different components, enabling real-time updates and interactions across processes.
06	output	Responsible for handling the output of camera streams, generating low-resolution preview segments, and broadcasting video feeds via WebSocket for real-time viewing. It includes classes and functions for converting video frames, managing WebSocket connections, and recording preview segments.
07	detectors	Provides an abstraction layer for handling various object detection models and their configurations. It includes classes and methods for initializing and managing detection models, pre-processing input data, and post-processing detection outputs.
08	track	Responsible for tracking objects detected by the system over time. This involves maintaining the state of detected objects, assigning unique IDs to them, and updating their status based on new detections.
09	events	Responsible for managing and processing events, particularly focusing on cleaning up old events based on retention policies, handling external events created by users, and processing internal event updates.
10	motions	Detects and processes motion events within video frames. It includes classes and methods for initializing motion detectors, improving contrast, masking frames, and identifying motion through contour analysis.
11	stats	Responsible for emitting and tracking various statistics related to the application. This includes system performance metrics, camera metrics, GPU/CPU usage, and other relevant data points.
12	record	Manages the recording and cleanup of video segments based on a configured retention policy. It includes functionality to handle expired recordings, export

		recordings, maintain recording segments in cache, and perform necessary database synchronization and cleanup.
13	util	Contains various utility functions and classes designed to assist with different tasks such as configuration management, event handling, image processing, object detection, and system monitoring.
14	Gmail Service	Facilitates integration with Google services, allowing users to log into the system using their Gmail accounts.
15	Database	Stores and manages structured data crucial for operations. It ensures data integrity, scalability, and efficient retrieval, supporting application functionality.

1.3 Machine Learning Model Architecture

1.3.1 Methodology

1.3.1.1 Pipeline overview

The proposed computer vision pipeline for efficient skeleton-based human fall detection integrates several techniques to achieve accurate and lightweight performance. The pipeline integrates multiple state-of-the-art techniques for motion detection, object detection, pose estimation, and fall detection. This system is optimized for deployment on various hardware configurations, offering flexibility in terms of processing power and accuracy. Figure 129 below provides a visual summary of this process.

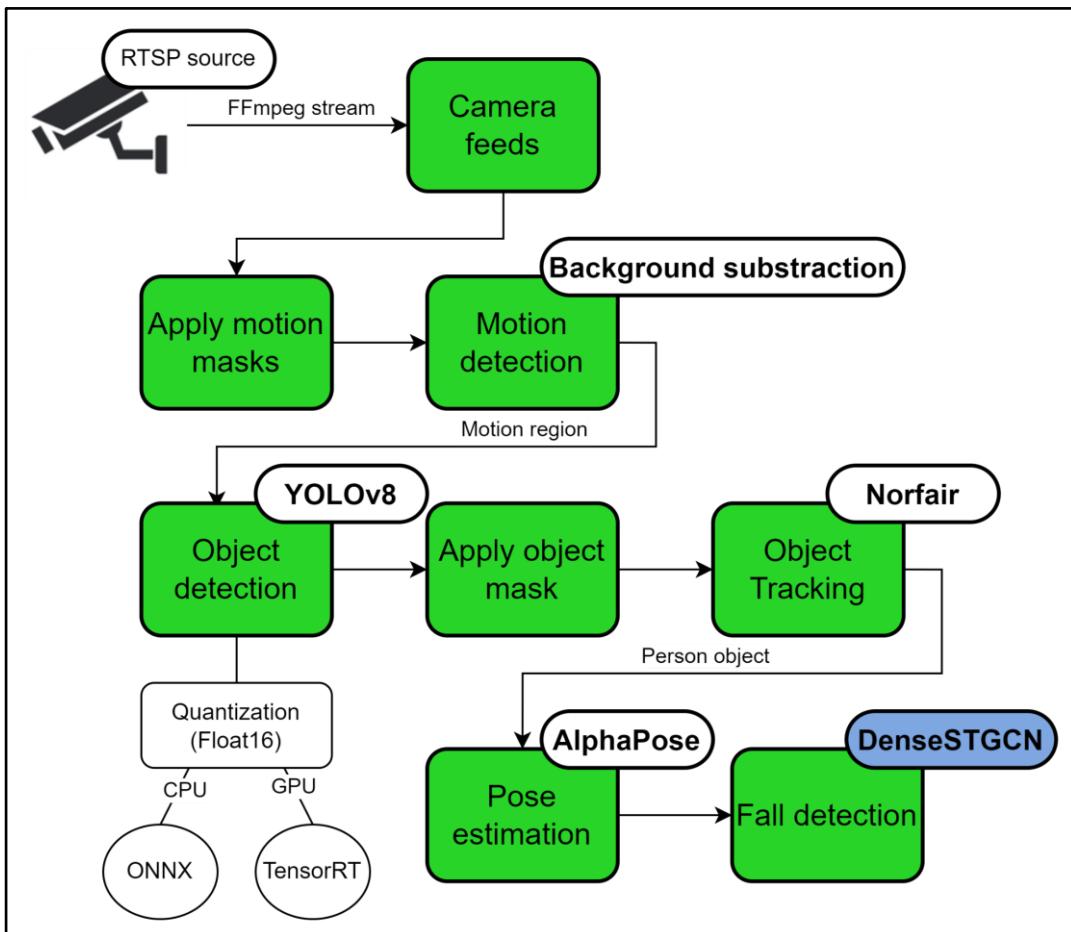


Figure 129. Overview of the computer vision pipeline

1.3.1.1.1 Camera Feeds and Stream Handling

The pipeline begins with real-time video input from an RTSP source, such as a surveillance camera. The video stream is processed using FFmpeg to handle the data and extract frames for analysis.

1.3.1.1.2 Motion Detection

To identify regions of interest within the video frames, background subtraction is performed, isolating moving objects from the static background. The motion detection component further refines this by focusing on the moving regions, which are then used to apply motion masks. These masks help in reducing the processing load by limiting the object detection to areas with significant motion.

1.3.1.1.3 Object Detection and Masking

YOLOv8 [31],[32], a state-of-the-art object detection model, is employed to detect objects within the motion regions. This model is optimized using quantization (Float16) techniques, with deployment options on both CPU and GPU through ONNX and TensorRT. Once the objects are detected, object masks are applied to identify and isolate the detected entities, focusing on people, which is crucial for the next steps in the pipeline.

1.3.1.1.4 Object Tracking

The Norfair tracking algorithm is used to maintain the identity of the detected objects across frames. This step is essential for ensuring that the same person is consistently tracked, even if they move around within the camera's field of view.

1.3.1.1.5 Pose Estimation

Once the human figures are detected, we utilize AlphaPose [16], a state-of-the-art pose estimation model, to extract skeletal keypoints from the detected human regions. AlphaPose is a top-down pose estimation approach that first detects individual people in an image and then estimates the pose of each person separately. It uses a deep neural network to estimate the positions of important body key joints like the shoulders, elbows, wrists, hips, knees, and ankles. The output of this stage is a sequence of skeletal representations, where each skeleton consists of a set of keypoints representing the body joints of the detected person. In our implementation, we utilize 14 keypoints provided by AlphaPose, as shown in Figure 130 (a).

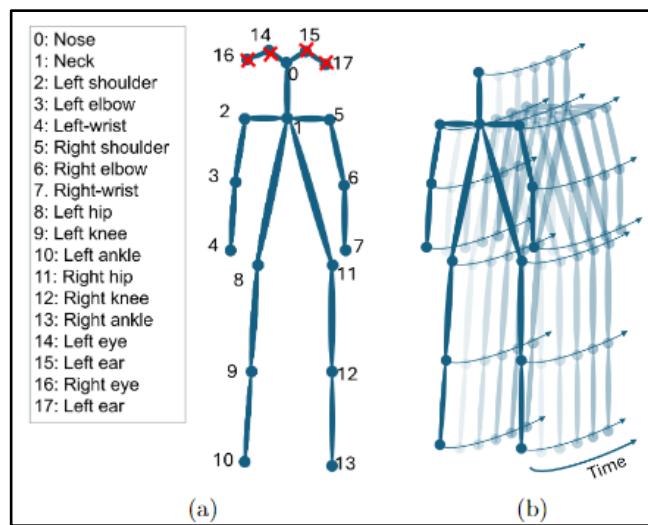


Figure 130. Skeleton representation with joint indices (a) and corresponding Spatio-Temporal Graph construction (b).

1.3.1.1.6 Skeleton-based Human Fall Detection

The final stage of the pipeline involves classifying the extracted skeletal sequences into fall or non-fall events. We propose a novel approach, DenseSTGCN, for this classification task. DenseSTGCN is an enhancement of the traditional STGCN [34], [35]. Inspired by DenseNet [36], our DenseSTGCN incorporates feature reusability, allowing the model to learn mappings with fewer parameters and avoid redundant computations. The DenseSTGCN processes the sequence of skeletal keypoints provided by AlphaPose, analyzing spatial and temporal features to classify the actions. This approach is particularly effective for fall detection, as it captures both the posture and movement patterns over time. The input to DenseSTGCN is a sequence of human skeletons represented as a spatial-temporal graph, as depicted in Figure 130 (b).

1.3.1.2 Dense Spatial-Temporal Graph Convolution Network

This section introduces the DenseSTGCN, our novel approach for skeleton-based action recognition. The DenseSTGCN architecture, illustrated in Figure 131 network architecture, is designed based on the traditional STGCN architecture to effectively capture both spatial and temporal dependencies within skeleton sequences architecture while incorporating feature reusability inspired by DenseNet. The network takes as input a sequence of human skeleton data represented as a spatial-temporal graph, it normalizes the data and processes this data through a series of densely connected STGCN layers. After that, global average pooling is applied to the feature maps. This reduces the spatial dimensions of the features while preserving the most important information. Finally, the output layer is a fully connected layer with a Sigmoid activation function that takes the features extracted from the final spatial-temporal graph convolution layer and outputs a probability distribution over the fall/not-fall classes.

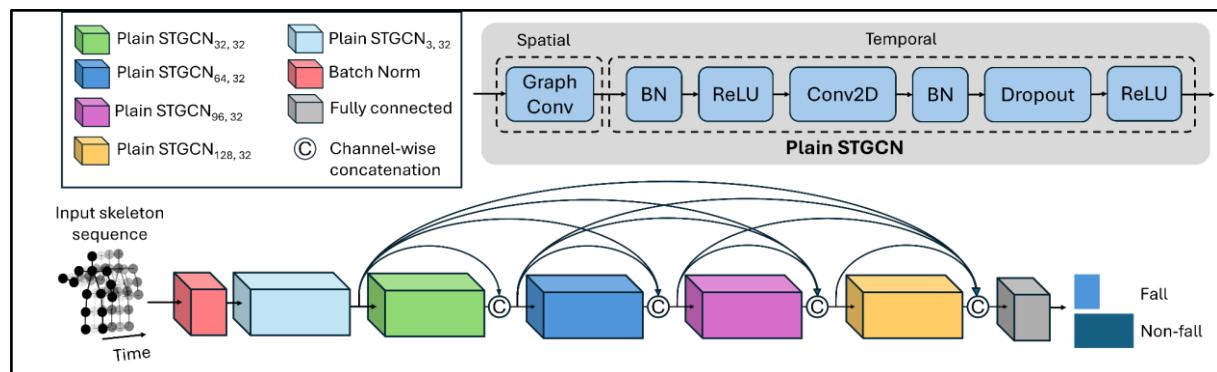


Figure 131. Network architecture of Dense Spatial-Temporal Graph Convolution Network (DenseSTGCN) for Skeleton-Based Human Fall Detection. Note: 'BN' stands for Batch Normalization, the subscript under each Plain STGCN are the input and output channels..

1.3.1.2.1 Input layer

The input to the DenseSTGCN is a sequence of human skeleton data, represented as a spatial-temporal graph. Each frame in the sequence is represented as a graph, where nodes correspond to the joints of the human skeleton and edges represent the spatial relationships between them. The temporal dimension is captured by stacking the graphs across consecutive frames. The input to the network is a sequence of T such graphs. The temporal dimension is captured by stacking the feature vectors of all nodes in each frame, resulting in a tensor of the size of size (T, N, C), where T is the number of frames, N is the number of joints, and C is the number of features per joint.

1.3.1.2.2 Plain STGCN layer

The core building layer of the DenseSTGCN is the Plain STGCN layer, which is a simplified version of the original STGCN layer without residual connection. This layer is used both in the head of the network for increasing the input channel to 32 and in the densely connected STGCN layers block. The Plain STGCN layer consists of two main components: a Spatial Convolutional Layer and a Temporal Convolutional Layer. These layers are designed to effectively extract both spatial and temporal information from the skeleton sequence. In Figure 131, the input to the first Plain STGCN layer is a tensor of shape $(T, N, 3)$, where T is the number of frames, N is the number of joints (14 in this case), and 3 represents the (x, y, score) coordinates for each joint. The output of this layer is a tensor of shape $(T, N, 32)$, where 32 is the number of output channels. Subsequent Plain STGCN layers in the densely connected block take as input the channel-wise concatenated feature maps from all preceding layers, resulting in an increasing number of input channels. The number of output channels for these layers remains 32.

- Spatial Convolutional Layer (GCL): GCL aims to capture the spatial relationships between joints in the skeleton. Each GCL takes as input a feature matrix X and an adjacency matrix A . First, the input tensor X undergoes a 2D convolution with a weight tensor W_g and biases b_g , resulting in an intermediate tensor X' . This intermediate tensor is reshaped to separate the multiple channels introduced by the kernel size, yielding a shape that incorporates the kernel dimension K and the output channels. Subsequently, the graph convolution aggregates information from neighbouring nodes based on the adjacency matrices A , which capture different types of relationships between nodes. This aggregation is performed by summing the product of each slice of X' (corresponding to a specific kernel dimension) with its respective adjacency matrix slice A_k . This process is represented by the Equation 1 below:

$$Y_{GCL} = \sum_{k=1}^K ((Conv2D(X, W_g)_k + b_g) \cdot A_k) \quad (1)$$

where Y_{GCL} is the output tensor that integrates spatial features from the graph structure and $Conv2D$ denotes the convolution operation.

- Temporal Convolutional Layer (TCL): serves to enhance the temporal representation of the input tensor Y_{GCL} , which has undergone previous graph convolutional processing. The TCL applies a convolutional operation with learned weights W_t and biases b_t over the temporal dimension, followed by batch normalization, ReLU activation, and dropout regularization. This sequential transformation aims to capture temporal dependencies within the data while controlling overfitting. The Equation 2 below succinctly encapsulates this process:

$$Y_{TCL} = \sigma(BN(Conv2D(Y_{GCL}, W_t) + b_t) \odot M) \quad (2)$$

where Y_{TCL} represents the output tensor, sigma denotes the activation function, BN signifies batch normalization, $Conv2D$ denotes the convolution operation, and M is a mask generated by dropout. This holistic approach ensures that the TCL operation effectively extracts meaningful temporal features while promoting model generalization and robustness. The number of input and output channels remains the same through TCL because the temporal

convolution operates along the time dimension while preserving the spatial and channel dimensions.

1.3.1.2.5 Densely connected STGCN layers

Traditional STGCN architectures employ residual connections between STGCN layers. These connections allow the network to learn "residual" information, which is the difference between the input and output of a layer. This helps to alleviate the vanishing gradient problem, which can occur in deep networks and enables the network to learn more effectively. However, the use of residual connections can increase the complexity of the model and require more computational resources. Drawing inspiration from the DenseNet architecture, the DenseSTGCN architecture utilizes densely connected STGCN layers instead of relying on residual connections to effectively capture and integrate features across multiple layers, thereby facilitating a comprehensive understanding of the spatio-temporal dynamics present in the data. DenseSTGCN consists of a series of Plain STGCN layer, denoted as PlainSTGNC_i for layer *i*. The input is an initial feature tensor X^0 , along with an adjacency matrix A and a set of edge importance weights W_e . The processing within the block follows an iterative procedure, where each PlainSTGCN refines and augments the feature set produced by the previous layer. The operation performed by each layer can be described as the following Equation 3:

$$X^{i+1} = \text{PlainSTGCN}(X^i, A \cdot W_e[i]) \quad (3)$$

Here, X^i represents the feature tensor input to the *i*th layer, $A \cdot W_e[i]$ represents the modulated adjacency matrix where the edge importance weights are applied, and X^{i+1} is the output feature tensor of the *i*th layer. This operation ensures that each layer can adaptively focus on the most significant edges in the graph, guided by the learned edge importance weights.

The process continues for a predefined number n of layers. After passing through all the layers, the output feature sets from each layer, including the initial input feature set, are concatenated along the channel dimension. This concatenation allows the network to preserve and utilize features at multiple levels of abstraction, providing a richer and more diverse representation of the spatio-temporal data. The final output can be expressed as the Equation 4 below:

$$X_{out} = concat(X^0, X^1, \dots, X^n) \quad (4)$$

This dense connectivity pattern ensures that each layer not only contributes to the final output but also retains its intermediate representations, thereby facilitating gradient flow and mitigating the vanishing gradient problem. The resulting concatenated feature tensor X_{out} embodies a comprehensive representation that captures both the temporal dynamics and spatial dependencies in the input data and allows the network to learn more complex features by combining information from multiple stages. The dense connections also promote a richer flow of information throughout the network, allowing features learned in earlier layers to contribute to the learning process in later layers. Furthermore, the dense connections reduce the number of parameters required for the network because features are reused, eliminating the need for redundant learning.

1.3.2 Experiments and evaluation

1.3.2.1 Dataset Preparation

The UR Fall Detection (URFD) dataset [37] serves as the backbone for evaluating the proposed DenseSTGCN for efficient skeleton-based human fall detection.

1.3.2.1.1 Overview

URFD is a benchmark dataset specifically curated for evaluating fall detection algorithms. It has 70 sequences (includes 30 falls sequences and 40 activities of daily living sequences), it's specifically curated for evaluating fall detection algorithms. This dataset consists of videos depicting different scenarios, including both fall and non-fall activities. Each video is annotated with corresponding human skeleton sequences extracted using advanced pose estimation techniques. This dataset includes both depth and RGB images captured from multiple camera angles. The fall events of this dataset are recorded using two Microsoft Kinect cameras, while the activities of daily living (ADL) event are recorded only using one camera. This visualization underscores the dataset's variety and complexity, essential for developing robust fall detection models. For our training data, we are only using fall RGB images shown as Figure 132.

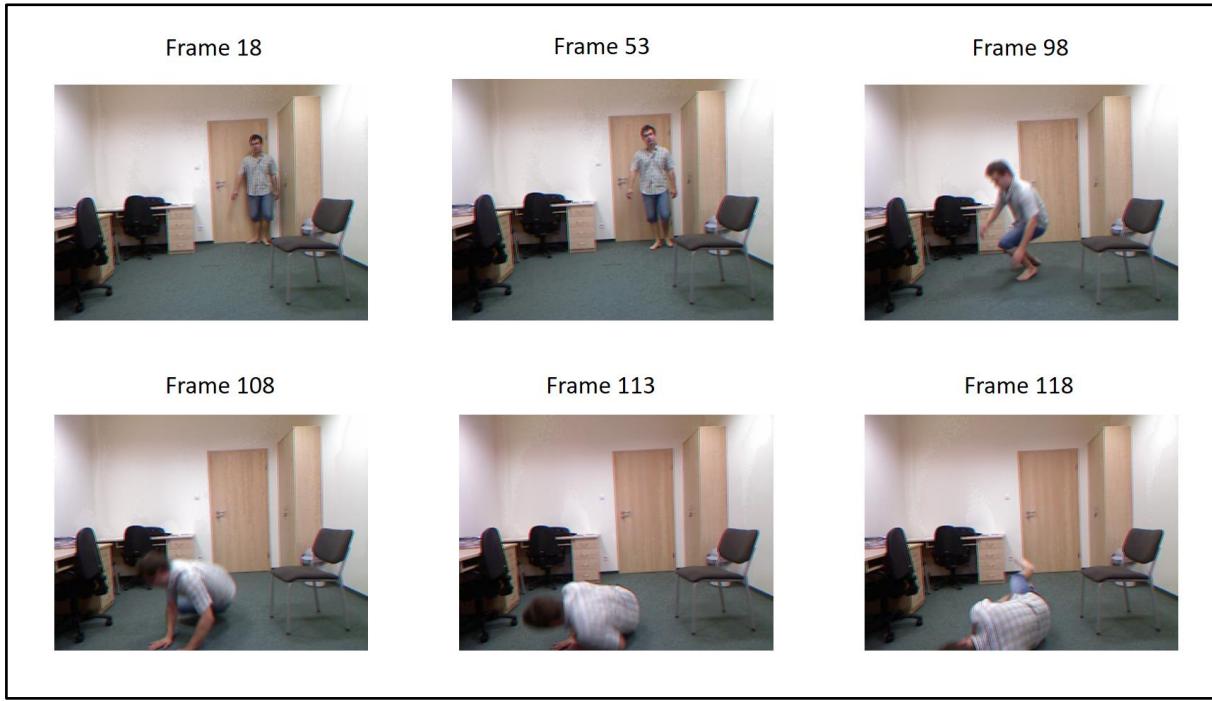


Figure 132. Example of fall RGB images from the URFD dataset.

1.3.2.1.2 Data Preprocessing

In the context of our research, the videos from URFD dataset are processed to extract skeleton sequences. This involves utilizing a top-down approach employing a lightweight human detector (TinyYOLOv3) and a lightweight pose estimator (AlphaPose) to extract human poses accurately from the input videos. For the purpose of this study, sequences of 6 consecutive frames are used for each skeleton. URFD facilitates binary classification for evaluating fall detection algorithms, categorizing activities into 'Fall' and 'Non-fall' instances. Our pose estimation successfully extracted 2941 out of 2995 frames, with 2046 'Non-fall' instances and 895 'Fall' instances, as shown in Figure 133 (a). For the purpose of this study, sequences of 6 consecutive frames are used for each skeleton, and a label smoothing technique is applied to the labels to improve the model's generalization capabilities and prevent overfitting. The data are divided into training, validation, and test sets to ensure the model's performance is thoroughly evaluated. The training set contains 1337 samples, the validation set is used for tuning model parameters and preventing overfitting, it includes 167 samples. And finally, the test set utilized for assessing the model's generalization capability, it comprises 168 samples, as shown in Figure 132 (b).

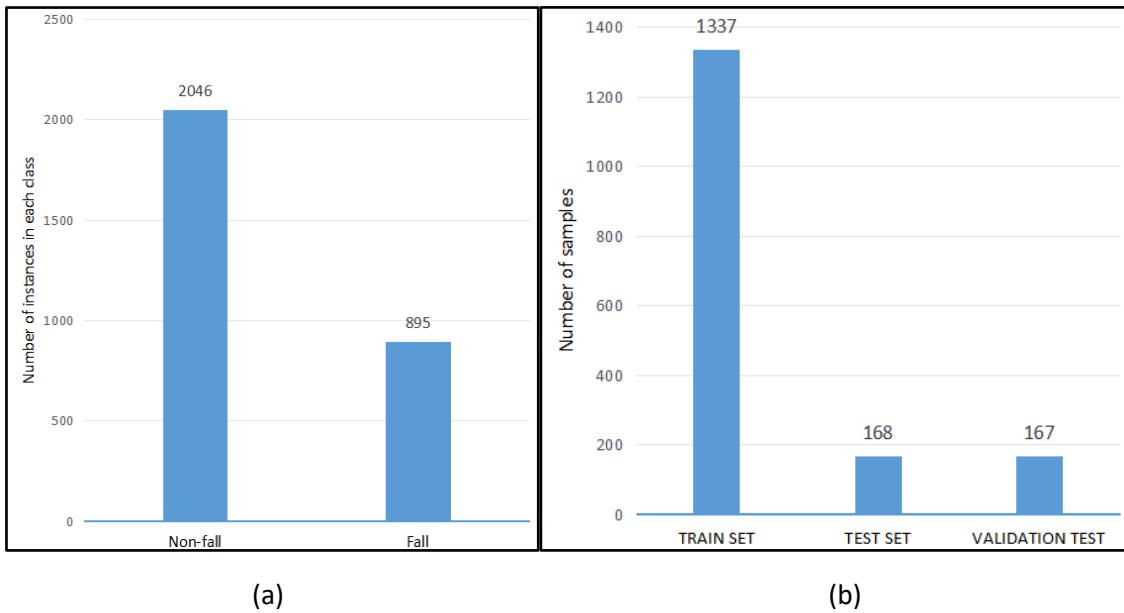


Figure 133. : Class distribution in URFD dataset (a) and Data partition for fall detection model training (b).

1.3.2.2 Evaluation Metrics

To thoroughly evaluate our DenseSTGCN models' effectiveness in binary classification, we use a variety of relevant assessment metrics. These metrics give us a comprehensive understanding of model performance, including accuracy, precision, recall, and the F1-score.

- Accuracy (acc): Accuracy is a metric that calculates the proportion of correct predictions both positive and negative out of the total number of predictions made. The Equation 5 below indicates the calculation for this metric:

$$acc = \frac{TP + TN}{FN + FP + TN + TP} \quad (5)$$

where FN , FP , TN , and TP stand for false negatives and false positives, true negatives and true positives, respectively.

- Precision (p): Precision measures the proportion of true positive predictions out of all positive predictions made by the model. This metric is calculated as the Equation 6 below:

$$p = \frac{TP}{TP + FP} \quad (6)$$

where TP and FP stand for true positives and false positives, respectively.

- Recall (r): Recall, also known as sensitivity, calculates the proportion of correctly predicted positive observations to the actual positive observations in the data. The calculation of this metric is provided as the Equation 7 below:

$$r = \frac{TP}{TP + FN} \quad (7)$$

where TP and FN stand for true positives and false negatives, respectively.

- F1-score (f1): The F1-score is a metric that combines precision and recall by calculating their harmonic mean. It serves as a balanced measure of a model's performance, taking into

account both the accuracy of positive predictions and the ability to capture all positive instances. It is defined in the following Equation 8:

$$f1 = \frac{r \times p}{r + p} \times 2 \quad (8)$$

where r and p stand for recall and precision, respectively.

In addition to these classification metrics, we also assess how computationally efficient the models are using additional metrics:

- Estimated Total Size: This metric represents the total size of the model, considering the memory footprint of all model parameters.
- FLOPs (Floating Point Operations Per Second): FLOPs is a metric used to measure the computational complexity of the model. It shows how many floating-point operations are necessary for the model to make predictions.
- Average Inference Time: This metric quantifies the average time taken by the model to process a single input and generate a prediction, providing insights into its real-time performance capabilities on both GPU and CPU.
- FPS (Frames Per Second): This metric, FPS (frames per second), measures the number of frames the model can process in one second. This indicates how well the model can perform in real-time on both CPU and GPU.

1.3.2.3 Implementation Details

For all models, we used the cross-entropy loss function and employed the Adadelta optimizer with an initial learning rate of 0.0001 for training process. The batch size was set to 32, and the models were trained for 80 epochs. To mitigate overfitting and enhance generalization, a dropout rate of 0.5 is employed at each Temporal Convolutional Layer. Additionally, the input skeleton sequences were normalized to have zero mean and unit variance, ensuring consistent scaling across different datasets. For the DenseSTGCN models, we experimented with different numbers of densely connected STGCN layers, namely 4 (DenseSTGCN₄), 12 (DenseSTGCN₁₂), and 18 layers (DenseSTGCN₁₈). This allowed us to investigate the impact of network depth on performance and efficiency.

1.3.2.1 Classification results on URFD dataset

To assess the efficacy of our proposed DenseSTGCN models, we conducted a comprehensive evaluation on the URFD dataset to compare their performance against the baseline STGCN model. Table 1 presents a comprehensive comparison of the proposed DenseSTGCN models (DenseSTGCN₄, DenseSTGCN₁₂, and DenseSTGCN₁₈) with the baseline STGCN model on the URFD dataset. This evaluation focused on classification performance metrics, including accuracy, precision, recall, and F1-score, and computational efficiency, including the number of parameters, estimated total size, FLOPs, and average inference time on both GPU and CPU.

1.3.2.2 Classification Performance

As demonstrated in Table 65, our proposed DenseSTGCN₁₈ achieves 99.40% accuracy, significantly outperforming the baseline STGCN, which achieves 97.40%. Specifically, DenseSTGCN₁₈ exhibits a precision of 99.41% and a recall of 99.40%, underscoring its robustness in fall detection tasks. The accuracy of DenseSTGCN₁₂ is 98.21%, which is slightly lower than DenseSTGCN₁₈ but still higher than the baseline STGCN. DenseSTGCN₄ has the lowest accuracy among the DenseSTGCN models at 94.64%, but it still maintains an acceptable level of accuracy in fall detection tasks. The confusion matrices in Figure 134 provide a detailed breakdown of the classification performance for each DenseSTGCN model and the baseline STGCN. The STGCN model exhibits a high true positive rate for both fall and

non-fall classes, indicating its ability to correctly classify most instances. However, there are a few misclassifications, particularly for the fall class, suggesting potential areas for improvement. DenseSTGCN₄ shows a good true positive rate for the non-fall class but a slightly lower true positive rate for the fall class compared to STGCN. DenseSTGCN₁₂ demonstrates a very high true positive rate for both fall and non-fall classes, indicating strong discriminative power. The number of misclassifications is minimal, suggesting that the increased depth of this model contributes to improved performance. DenseSTGCN₁₈ achieves near-perfect classification, with almost all instances correctly classified. The confusion matrix shows very few misclassifications, highlighting the good performance of DenseSTGCN₁₈ in accurately distinguishing falls from non-falls.

Table 65. Comprehensive comparison of classification performance for DenseSTGCN models versus baseline STGCN on the URFD dataset. Note: The highest results are in bold, and the second-highest results are in bold and italics.

Backbone	Accuracy	Precision	Recall	F1-score
STGCN (Baseline)	0.973958	0.971757	0.970238	0.970115
DenseSTGCN ₄	0.946429	0.951156	0.946429	0.945955
DenseSTGCN ₁₂	0.982143	0.982701	0.982143	0.982102
DenseSTGCN ₁₈	0.994048	0.994111	0.994048	0.994044

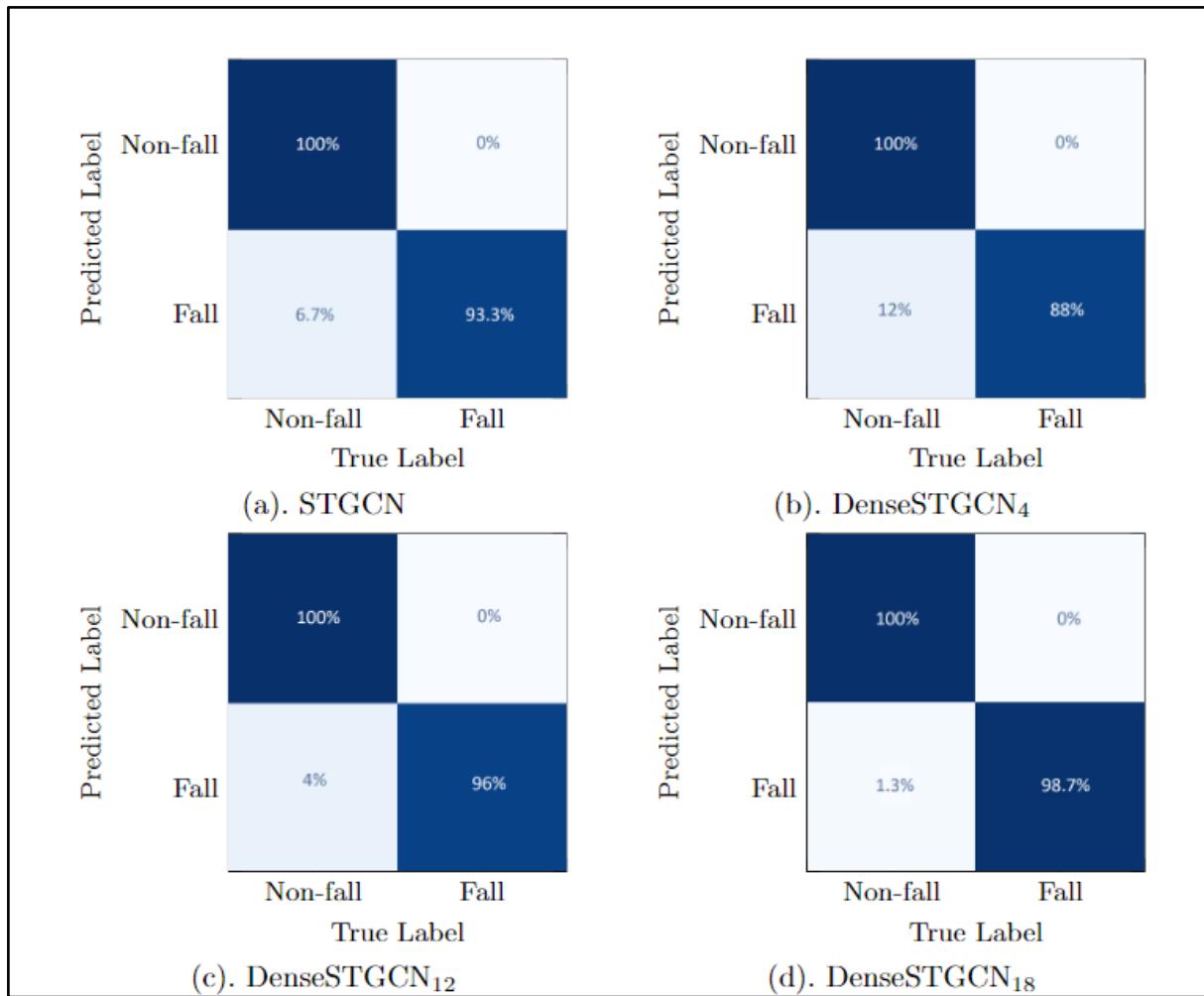


Figure 134. Confusion matrices for fall and non-fall classification of DenseSTGCN models and baseline STGCN on the URFD dataset.

1.3.2.3 Computational Efficiency

In the context of computational efficiency, Table 66 provides a comprehensive comparison of our DenseSTGCN models against the baseline STGCN model. The metrics evaluated include the number of parameters, estimated total size, FLOPs (Floating Point Operations per Second), and average inference time on both GPU and CPU. The DenseSTGCN models exhibit a notable decrease in parameter count in contrast to the baseline STGCN model. For instance, DenseSTGCN₄ has only 81,714 parameters, while the baseline STGCN has 3,070,990 parameters. This reduction translates to a smaller model size, with DenseSTGCN₄ having an estimated total size of 21.06 MB, significantly less than the 118.79 MB of the baseline STGCN. Even the more complex DenseSTGCN₁₈ model maintains a relatively compact size of 81.37 MB. Additionally, the computational complexity, as indicated by the FLOPs metric, shows that DenseSTGCN models are more efficient. The baseline STGCN model has a FLOPs value of 3.417923 GFLOPs, whereas DenseSTGCN₄ requires only 0.210696 GFLOPs. Even DenseSTGCN₁₈, which is the most complex among the DenseSTGCN variants, requires significantly fewer FLOPs (1.896638 GFLOPs) compared to the baseline STGCN. Moreover, inference time is crucial for real-time applications. The DenseSTGCN models exhibit substantial improvements in inference time on both GPU and CPU. DenseSTGCN₄ achieves the fastest inference times, with an average of 0.189 milliseconds on GPU and 0.411 milliseconds on CPU. DenseSTGCN₁₈, while having higher computational demands, still maintains reasonable inference times of 0.602 milliseconds on GPU and 1.910 milliseconds on CPU.

Table 66. Comprehensive comparison of computational efficiency for DenseSTGCN models versus baseline STGCN on the URFD dataset. Note: The best results are in bold, and the second best results are in bold and italics.

Backbone	Parameters	Estimated Total Size (MB)	FLOPs (G)	Average Inference Time on GPU (millisecond)	Average Inference Time on CPU (millisecond)
STGCN (Baseline)	3,070,990	118.79	3.417923	0.450	1.450
DenseSTGCN ₄	81,714	21.06	0.210696	0.189	0.411
DenseSTGCN ₁₂	371,602	55.23	0.975910	0.497	1.157
DenseSTGCN ₁₈	718,042	81.37	1.896638	0.602	1.910

1.3.2.4 Trade-off Between Accuracy and Computational Efficiency

In addition to accuracy, DenseSTGCN models offer notable advantages in parameter efficiency, as shown in Figure 135 (a). For instance, DenseSTGCN₄, with only 81,714 parameters, achieves 94.64% accuracy, while the baseline STGCN requires 3,070,990 parameters to achieve 97.4% accuracy. This reduction in parameters translates to a substantial decrease in memory footprint as indicated in Figure 135 (b), DenseSTGCN₄ has the smallest estimated total size (21.06 MB), making it the most lightweight model. DenseSTGCN₁₂ and DenseSTGCN₁₈ have larger sizes but are still considerably smaller than STGCN. The reduced number of parameters and smaller model sizes in DenseSTGCN models lead to lower computational complexity, as indicated by the FLOPs metric as Figure 135 (c). The reduced complexity of DenseSTGCN models translates to faster inference times on both GPU and CPU. As depicted Figure 135 (d) and (e), DenseSTGCN₄ has the fastest inference time on both GPU (0.189 milliseconds) and CPU (0.411 milliseconds), followed by DenseSTGCN₁₂ and DenseSTGCN₁₈. Although DenseSTGCN₁₈ has higher computational demands, with 1.897 GFLOPs, it maintains reasonable inference times of 0.602 milliseconds on GPU and 1.910 milliseconds on CPU. These findings indicate that relying solely on FLOPs is inadequate for accurately predicting inference time, due to the considerable variations in GPU computational efficiencies among models. DenseSTGCN models, particularly DenseSTGCN₁₈, utilize GPU computing resources more effectively, achieving a balance between performance and computational efficiency.

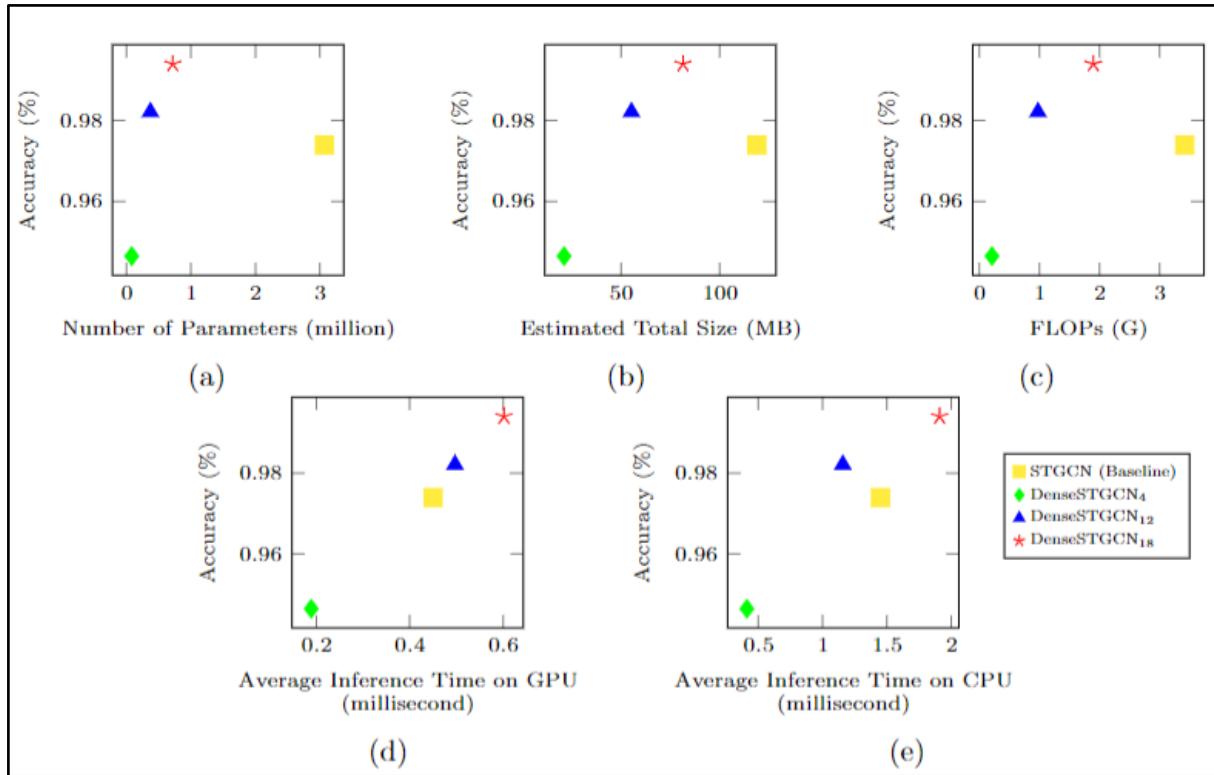


Figure 135. Comparison of different DenseSTGCN models with STGCN. (a) Accuracy vs. Number of Parameters. (b) Accuracy vs. Estimated Total Size. (c) Accuracy vs. FLOPs. (d) Accuracy vs. Average Inference Time on GPU. (e) Accuracy vs. Average Inference Time on CPU.

1.3.2.5 Real-time performance discussion

Table 67 presents the Frames Per Second (FPS) obtained during the process of using TinyYOLOv3 for human detection, AlphaPose for pose estimation, and different fall detection methods. The results show that DenseSTGCN₄ achieves the highest efficiency with 17.25 FPS on GPU and 5.27 FPS on CPU, indicating it is well-suited for real-time applications because of its reduced computational complexity. The baseline STGCN, while slightly less efficient with 15.55 FPS on GPU and 5.10 FPS on CPU, still performs competitively. DenseSTGCN₁₂ and DenseSTGCN₁₈ demonstrate a trade-off between performance and complexity, with DenseSTGCN₁₂ slightly below the baseline at 15.53 FPS on GPU and 5.23 FPS on CPU, and DenseSTGCN₁₈ showing the lowest FPS of 14.78 on GPU and 5.14 on CPU due to its higher computational demands. Thus, while DenseSTGCN₄ is ideal for applications requiring quick responses, DenseSTGCN₁₈ might be preferred for scenarios prioritizing accuracy over speed. In addition, to demonstrate the model's capability to accurately identify fall events in various scenarios, Figure 136 illustrates the fall detection outcomes of the DenseSTGCN₁₈ model on several images from the URFD dataset. The model successfully identifies and highlights the skeletal structure of the person in each image, accurately classifying fall events (shown in red) and non-fall events (shown in green).

Table 67. Average FPS on GPU and CPU for different fall detection methods using the same workflow of TinyYOLOv3 and AlphaPose for extracting the skeleton data. Note: The highest results are in bold, and the second-highest results are in bold and italics.

Fall detection method	FPS on GPU	FPS on CPU
STGCN (Baseline)	15.55	5.10
DenseSTGCN ₄	17.25	5.27
DenseSTGCN ₁₂	15.53	5.23
DenseSTGCN ₁₈	14.78	5.14

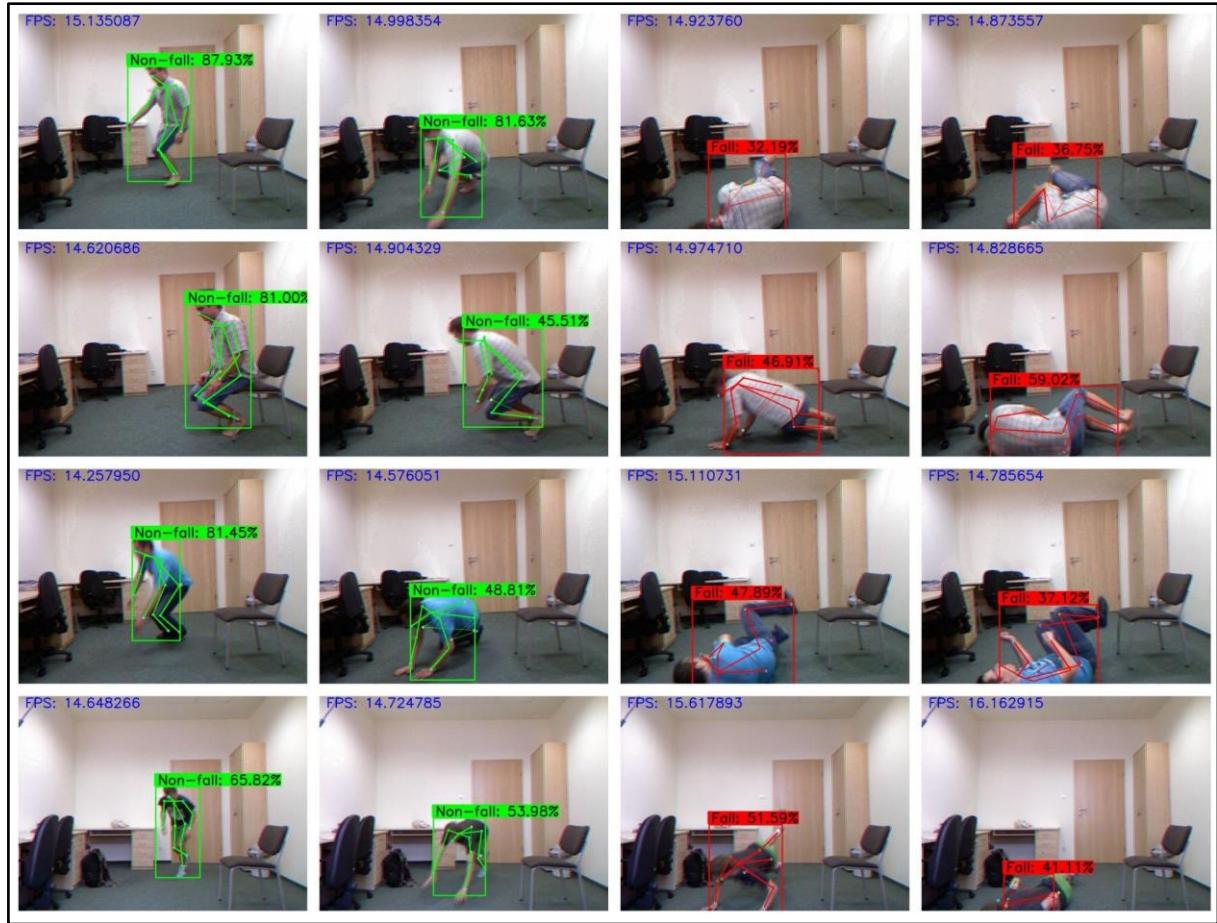


Figure 136. Fall detection results of the DenseSTGCN18 on some of the URFD dataset images. Images from each row in the figure belong to the same sequence of consecutive frames of a fall event.

1.3.2.6 Comparison with State-of-the-art methods

In this section, we compare the performance of our proposed DenseSTGCN models with state-of-the-art methods on the URFD dataset. Table 68 presents a comprehensive comparison of various techniques, including optical flow, MHI, LSTM, and GCNs. Our DenseSTGCN models, particularly DenseSTGCN₁₈, demonstrates superior performance compared to existing methods. DenseSTGCN₁₈ achieves a remarkable accuracy of 99.40%, surpassing all other approaches. Notably, it outperforms the baseline ST-GCN by 2%, highlighting the effectiveness of incorporating feature reusability from

DenseNet. Moreover, DenseSTGCN₁₈ exhibits high precision (99.41%) and recall (99.40%), indicating its ability to accurately identify both fall and non-fall events. In comparison to other GCN-based methods, DenseSTGCN₁₈ also demonstrates superior accuracy. While Abdo et al. [40] achieved 98.20% accuracy using ST-GCN with OpenPose, our DenseSTGCN₁₈ further improves upon this result. Similarly, Zheng et al. [26] reported 97.28% accuracy using ST-GCN with AlphaPose, which is also lower than our DenseSTGCN₁₈'s performance.

Table 68. Comparison with state-of-the-art fall detection methods on URFD dataset. Note: The highest and second-highest results are in bold and bold italics respectively

Method	Technique	Accuracy	Precision	Recall	F1-score
Núñez-Marcos et al. (2017) [38]	Optical flow	0.950000	-	1.000000	-
Cai et al. (2019) [24]	CC-MHI	0.923400	-	-	-
Lin et al. (2021) [25]	LSTM, OpenPose	0.982000	-	1.000000	-
Romaissa et al. (2021) [39]	Bidirectional LSTM	0.960000	1.000000	0.866650	-
Abdo et al. (2023) [40]	ST-GCN, AlphaPose	0.982000	0.976000	0.994000	0.974000
Zheng et al. (2022) [26]	ST-GCN, AlphaPose	0.972800	0.971500	0.974300	-
DenseSTGCN ₄	DenseSTGCN, AlphaPose	0.946429	0.951156	0.946429	0.945955
DenseSTGCN ₁₂	DenseSTGCN, AlphaPose	0.982143	0.982701	0.982143	0.982102
DenseSTGCN ₁₈	DenseSTGCN, AlphaPose	0.994048	0.994111	0.994048	0.994044

2 Background Processing Design

2.1 Recording Management Processing

2.1.1 Class Diagram

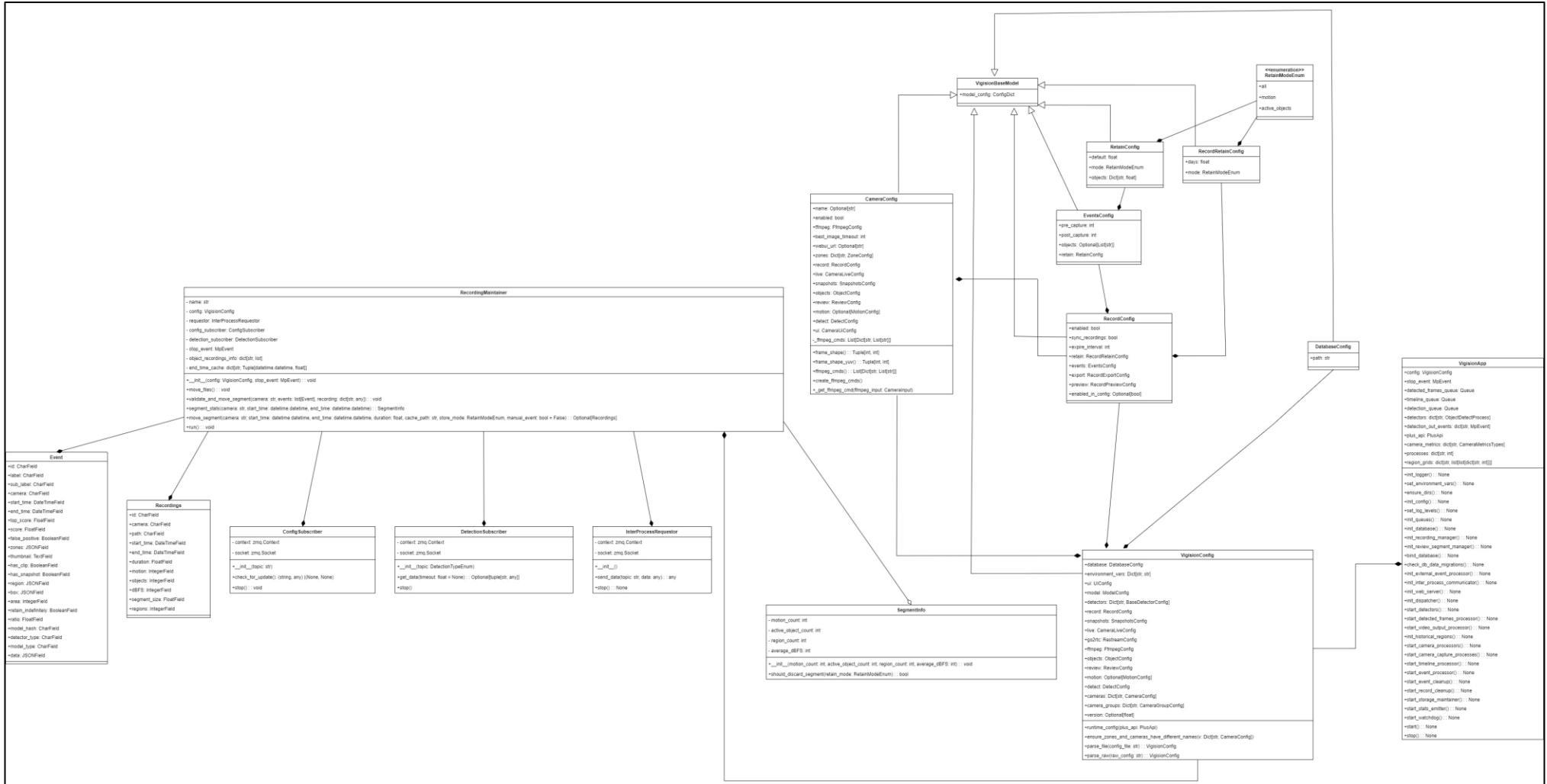
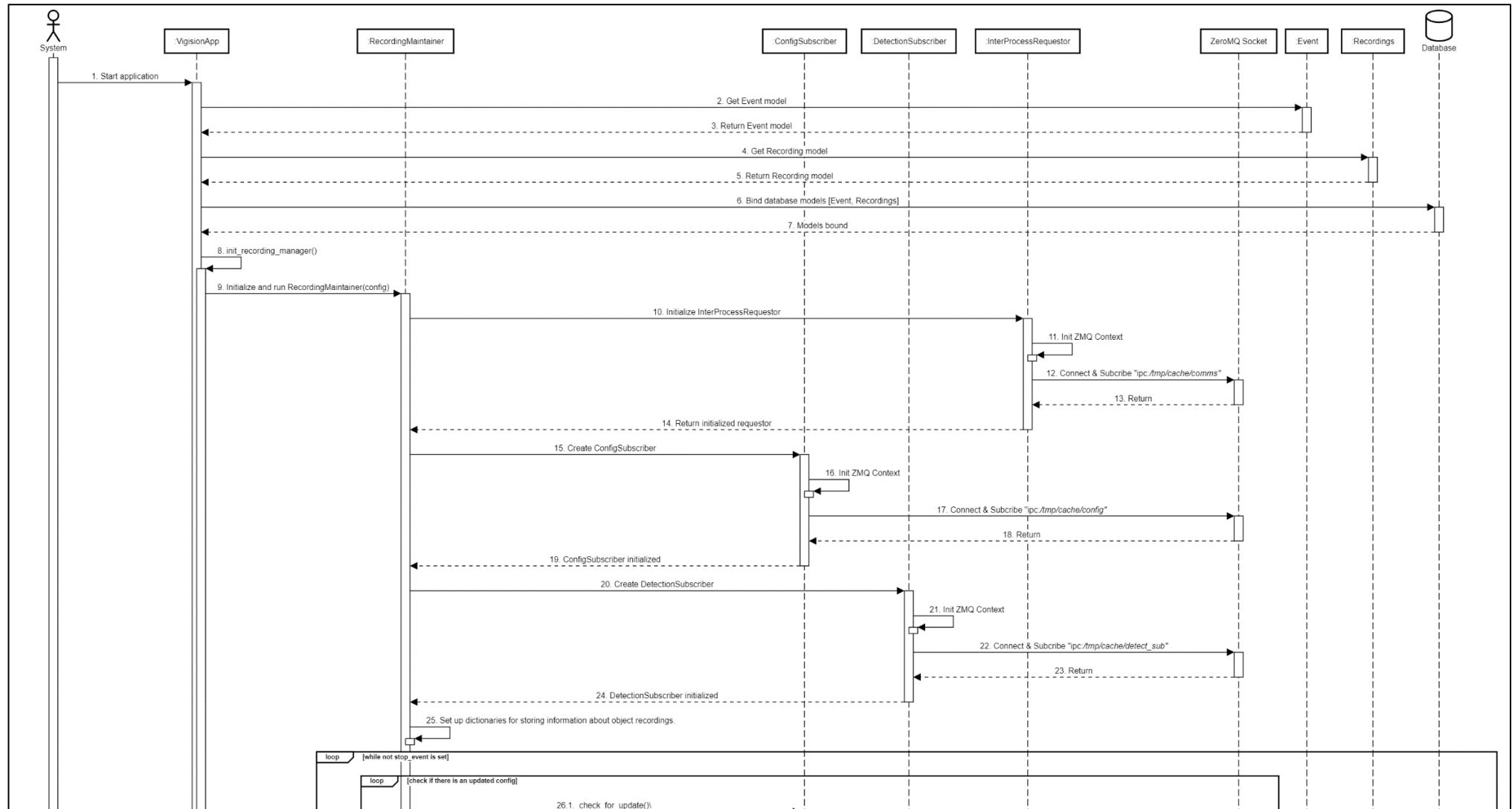
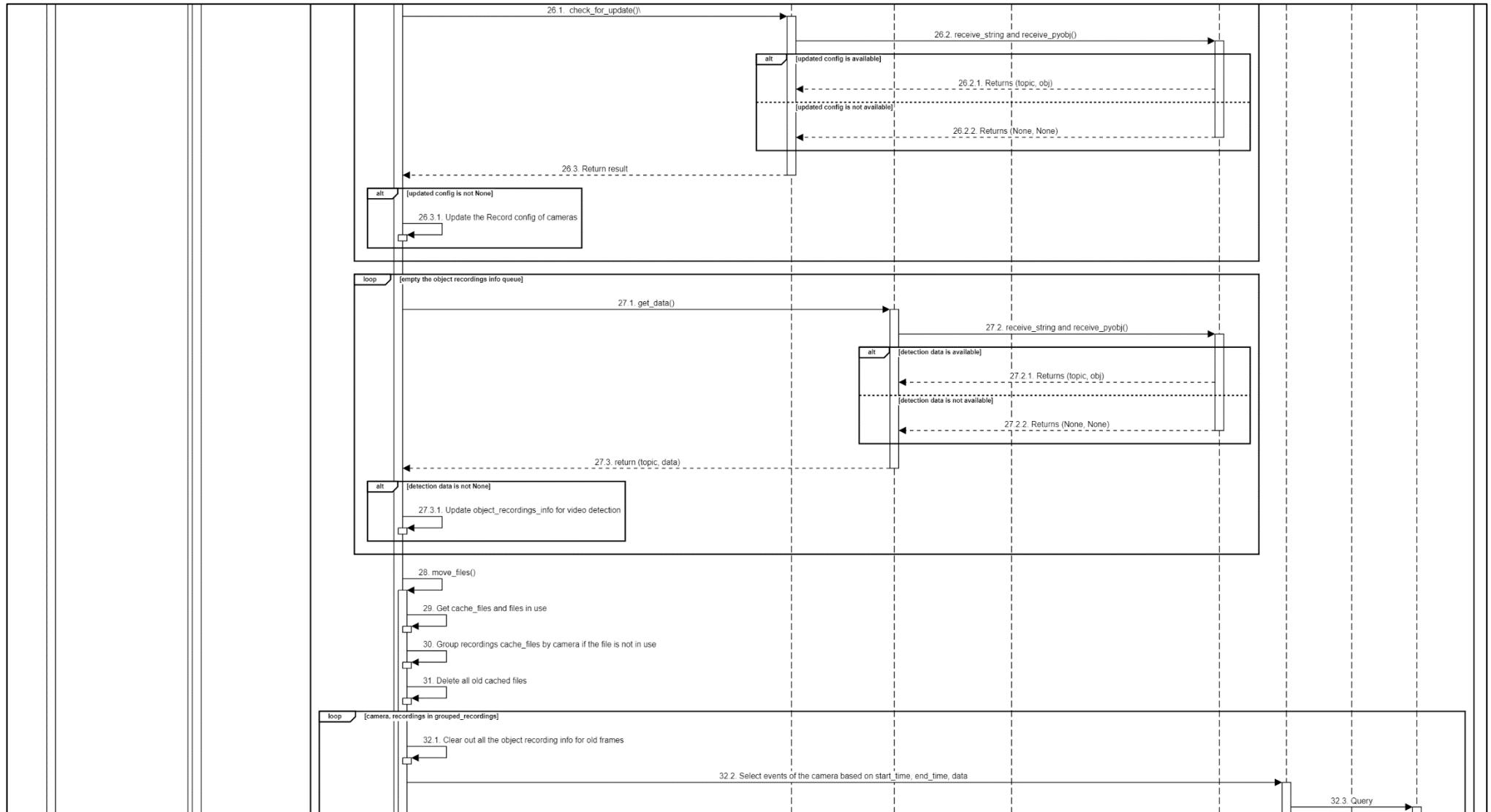
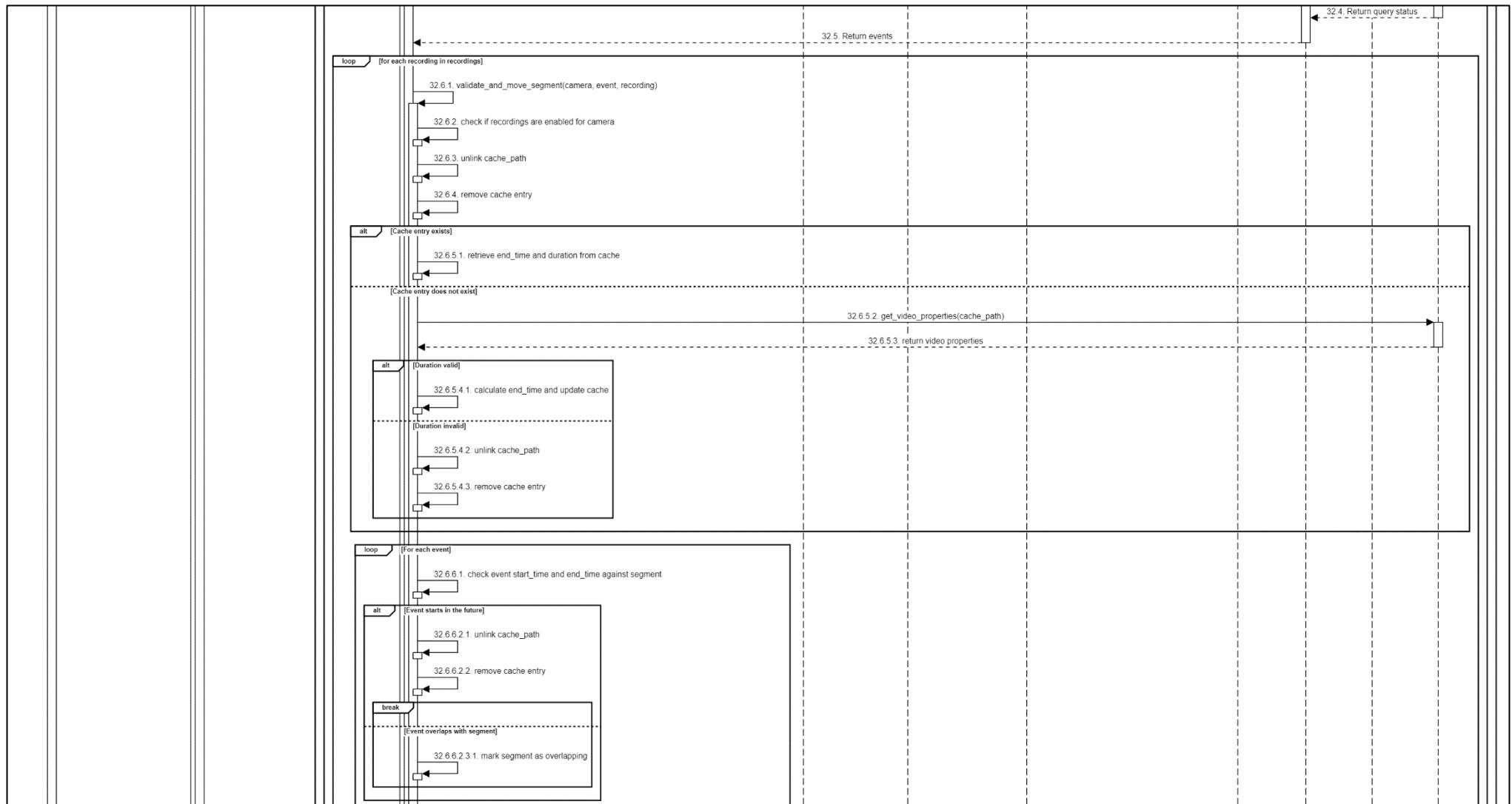


Figure 137. Class diagram - Recording Management Processing

2.1.2 Sequence Diagram







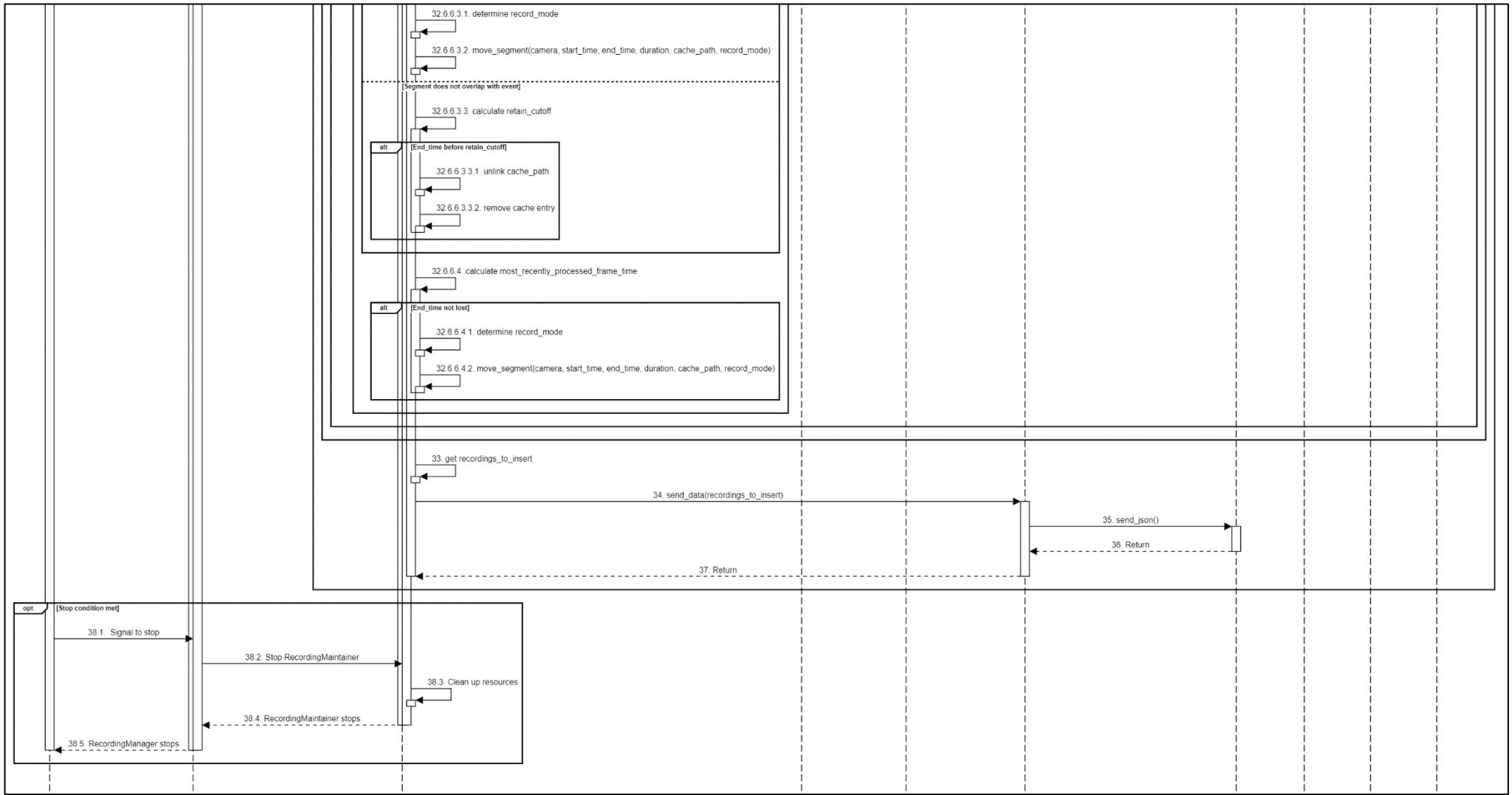


Figure 138. Sequence diagram - Recording Management Processing

2.2 Review Segment Management Processing

2.2.1 Class Diagram

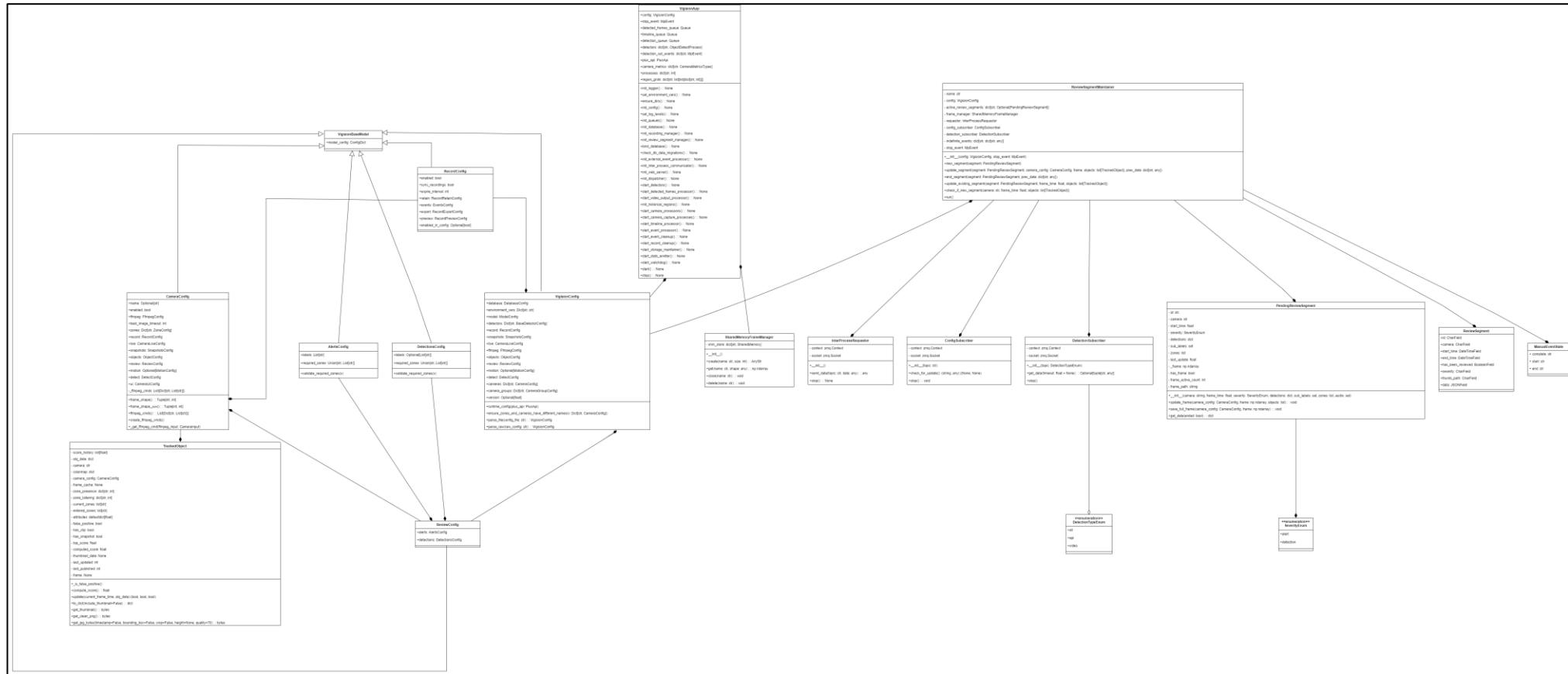
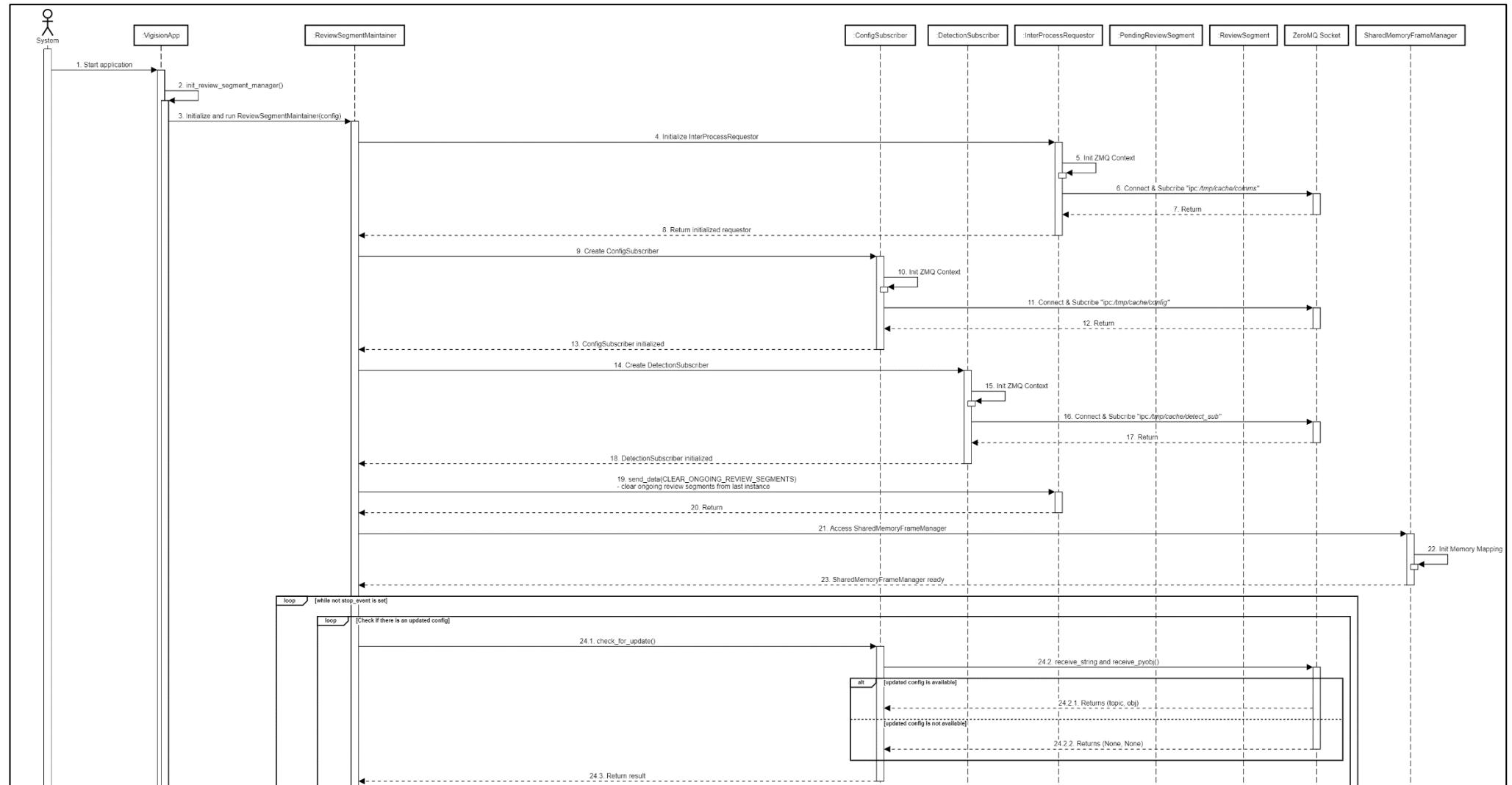
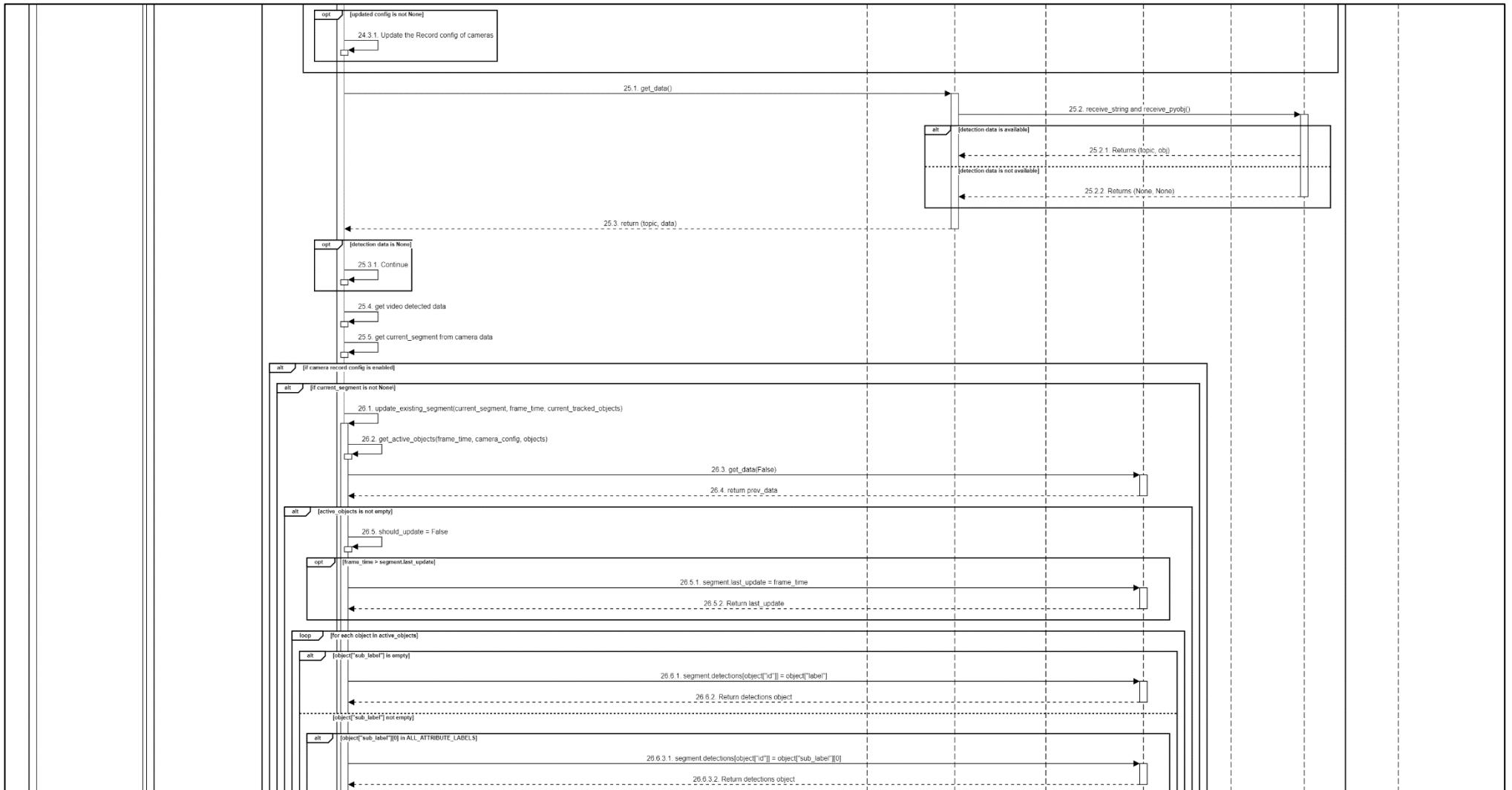
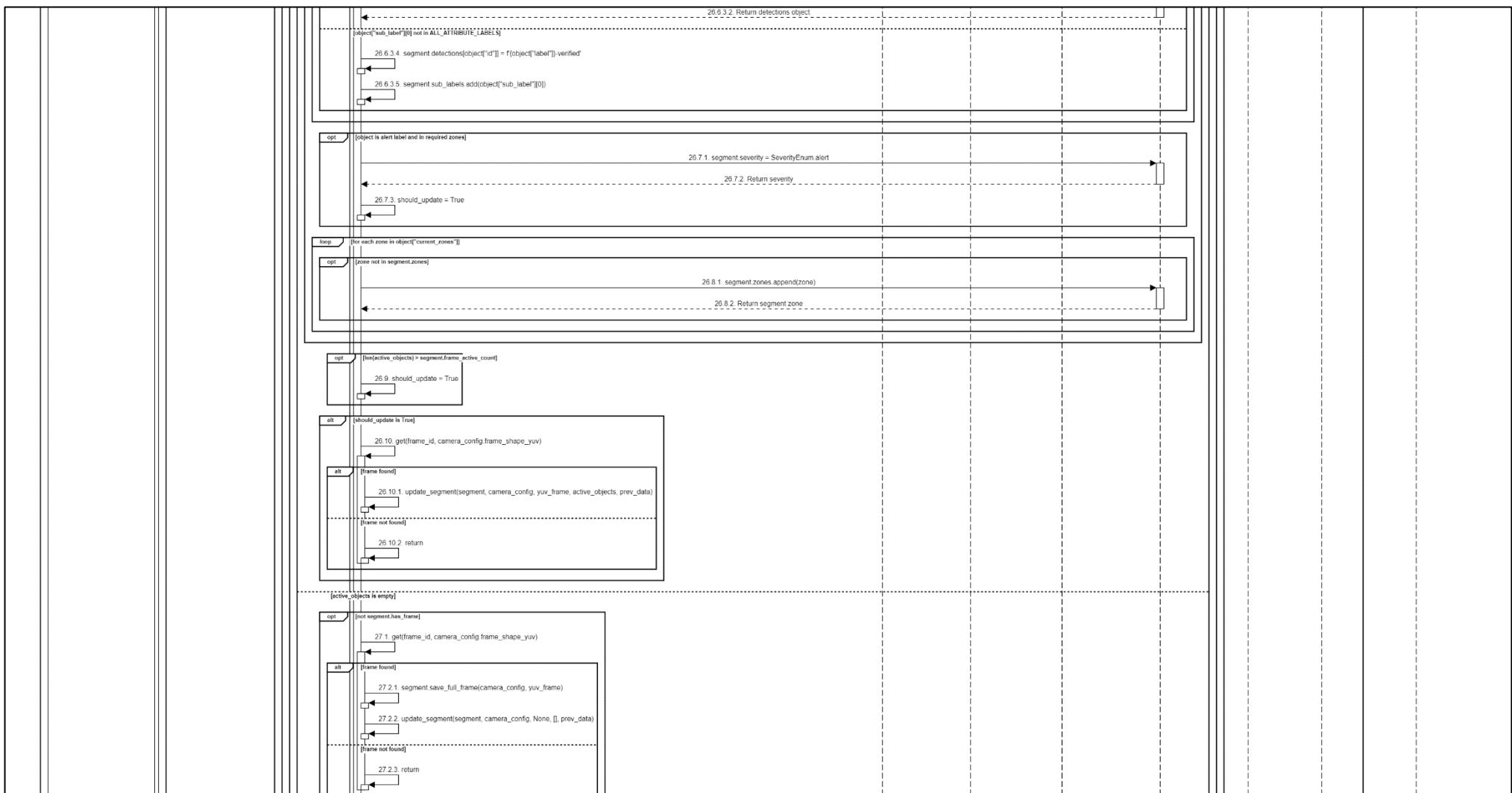


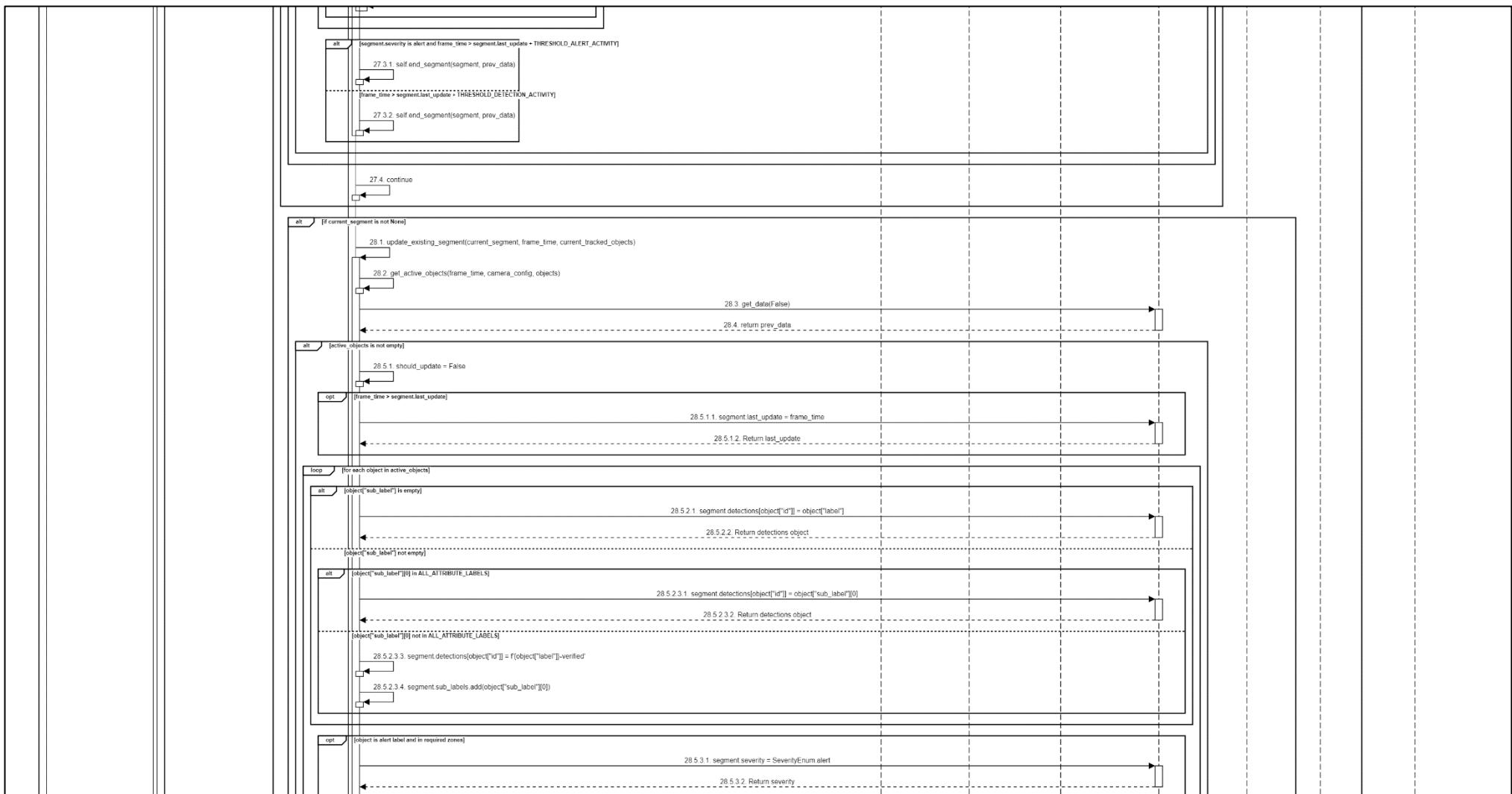
Figure 139. Class diagram - Review Segment Management Processing

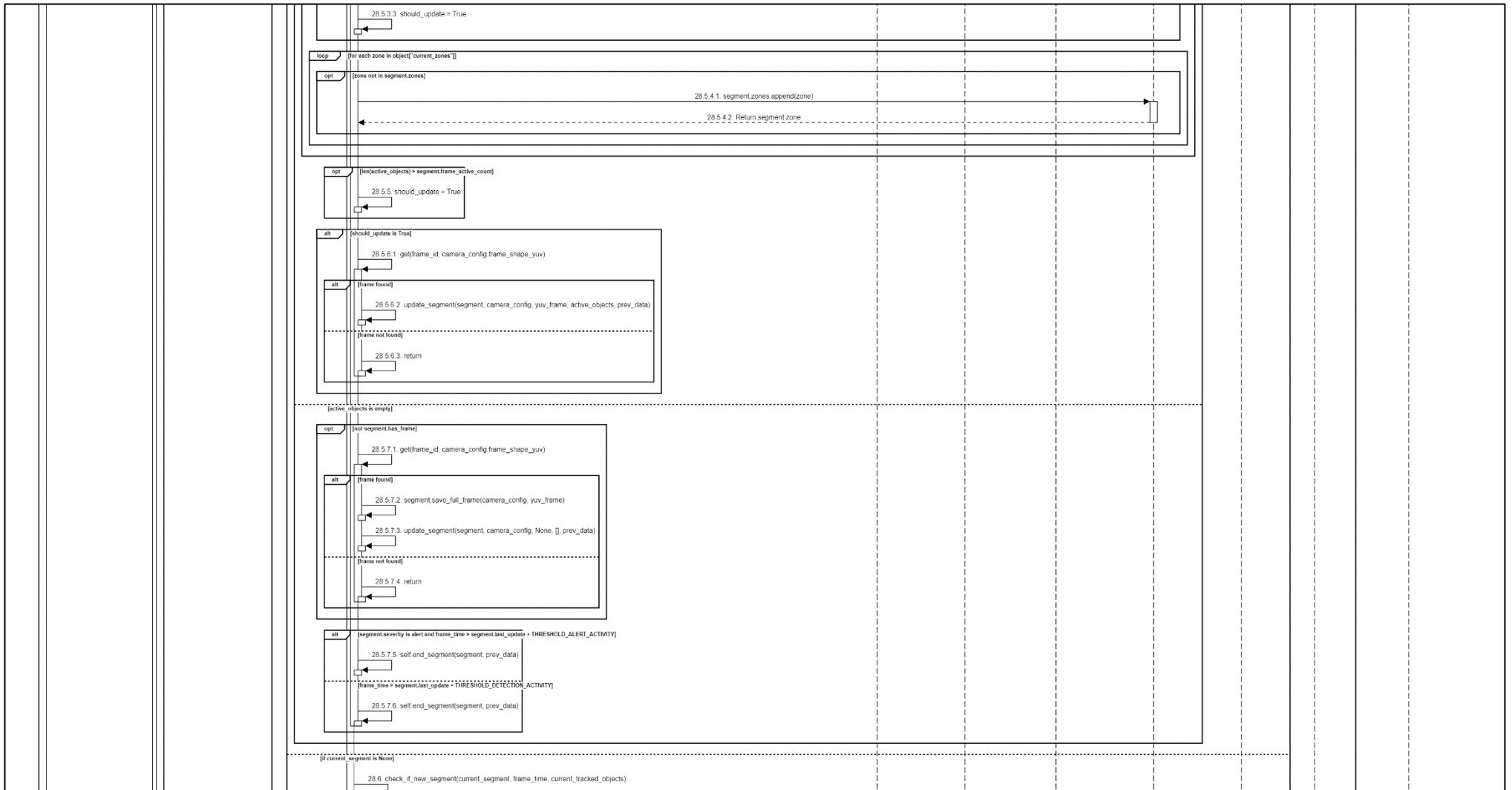
2.2.2 Sequence Diagram

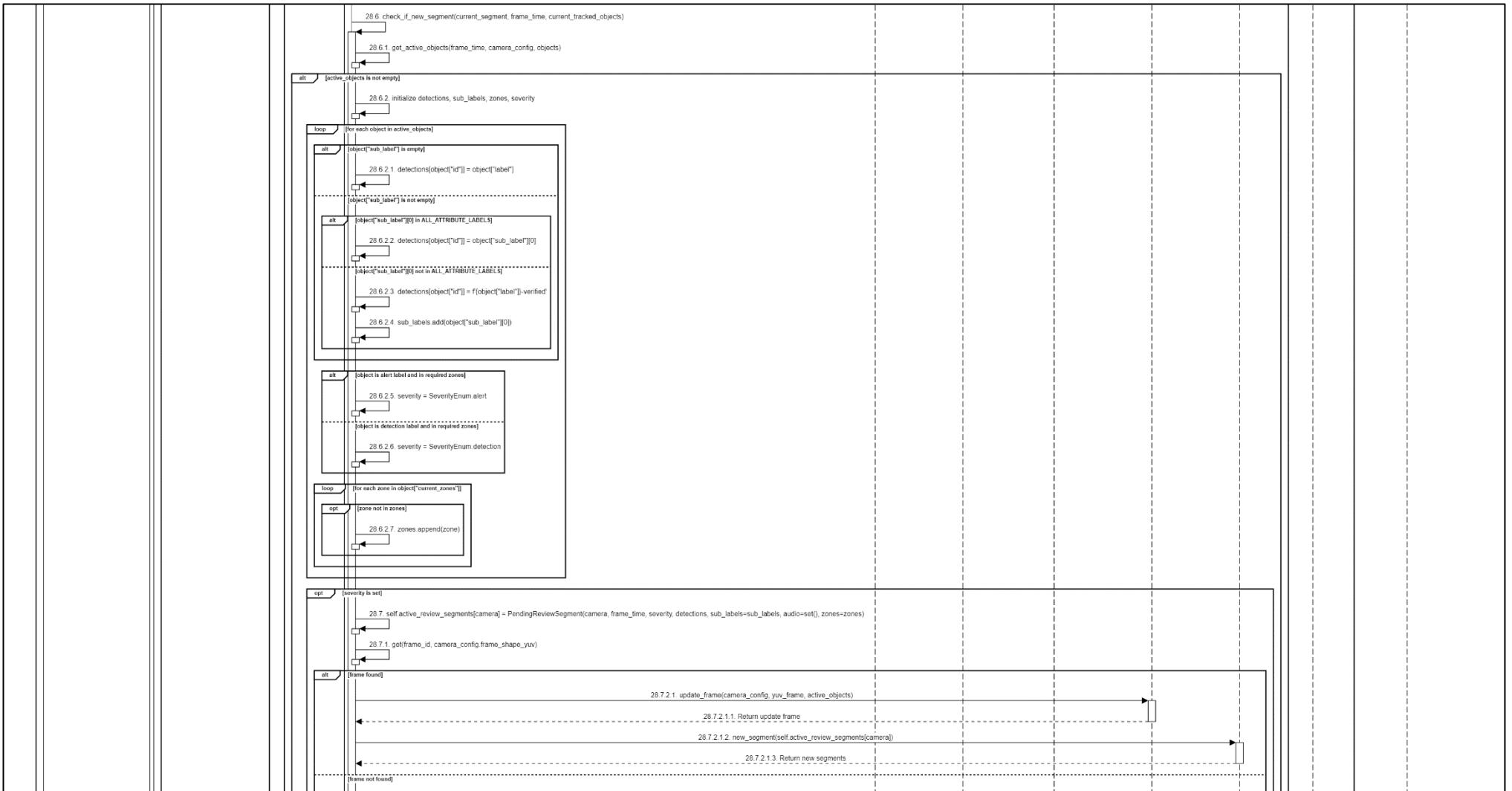












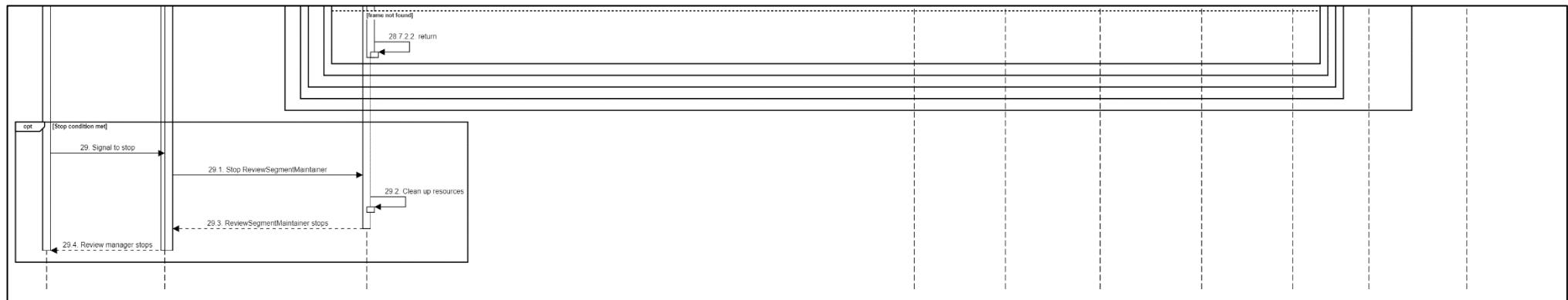


Figure 140. Sequence diagram - Review Segment Management Processing

2.3 Object Detection Processing

2.3.1 Class Diagram

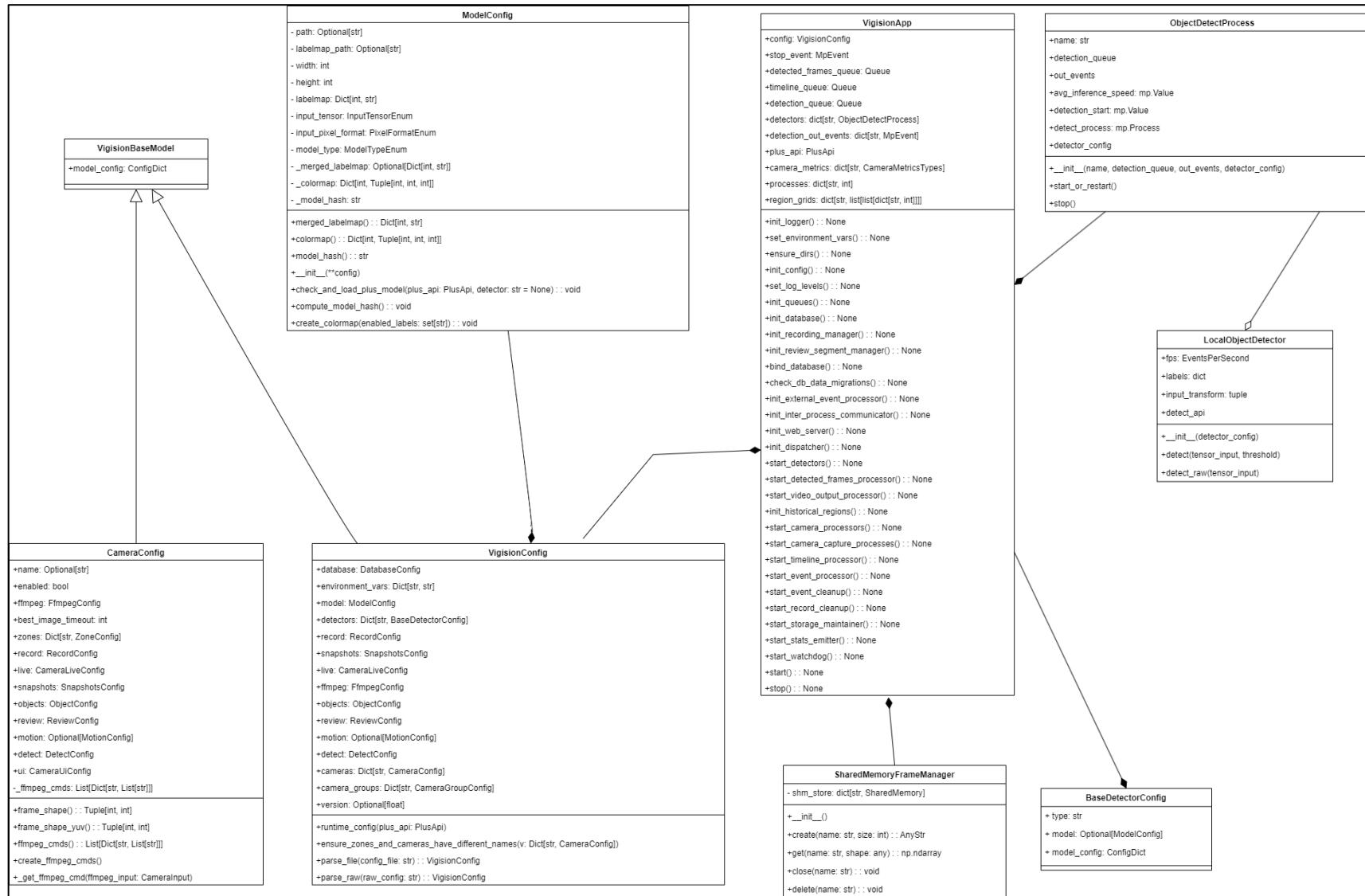
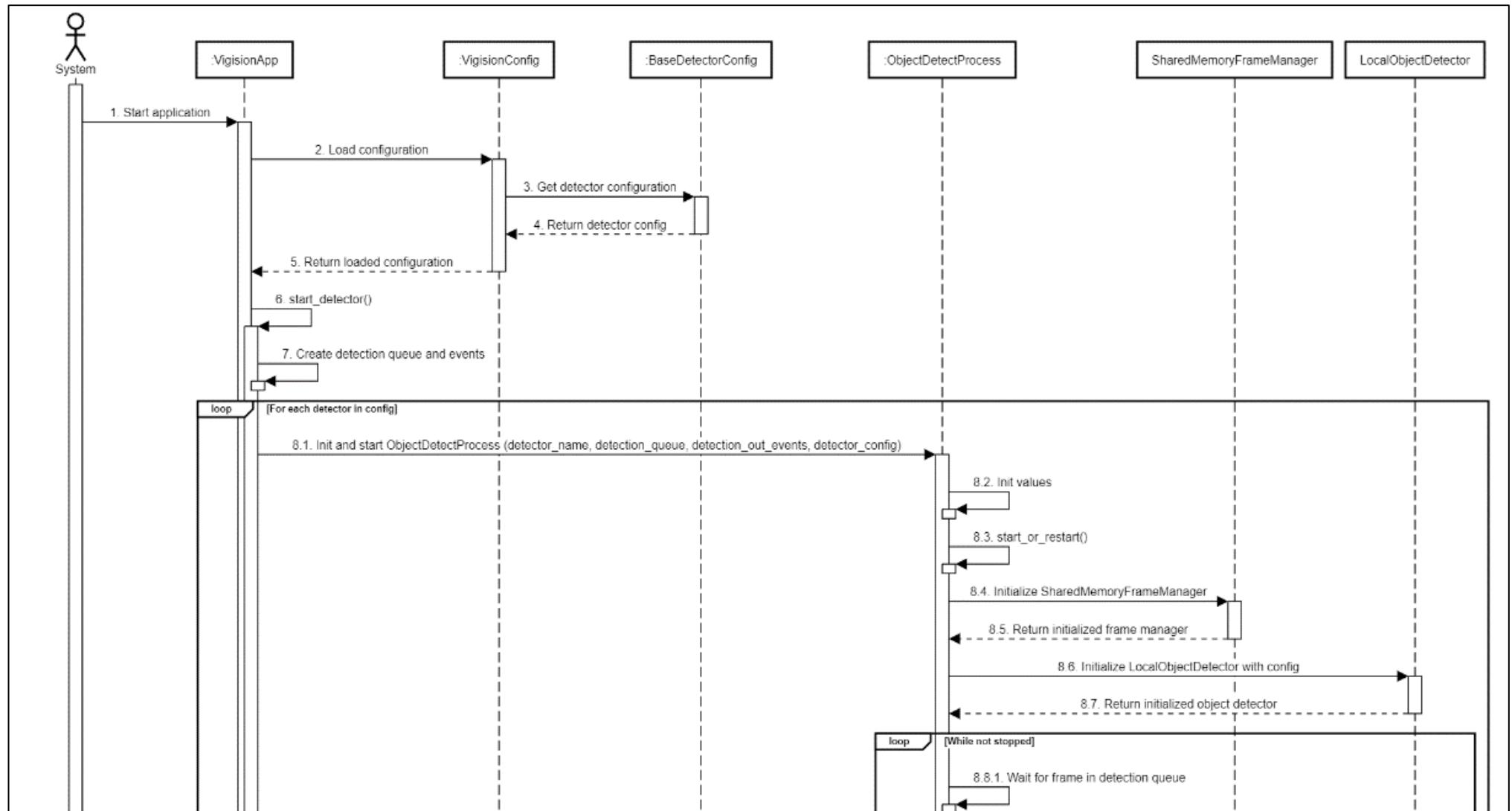


Figure 141. Class diagram - Object Detection Processing

2.3.2 Sequence Diagram



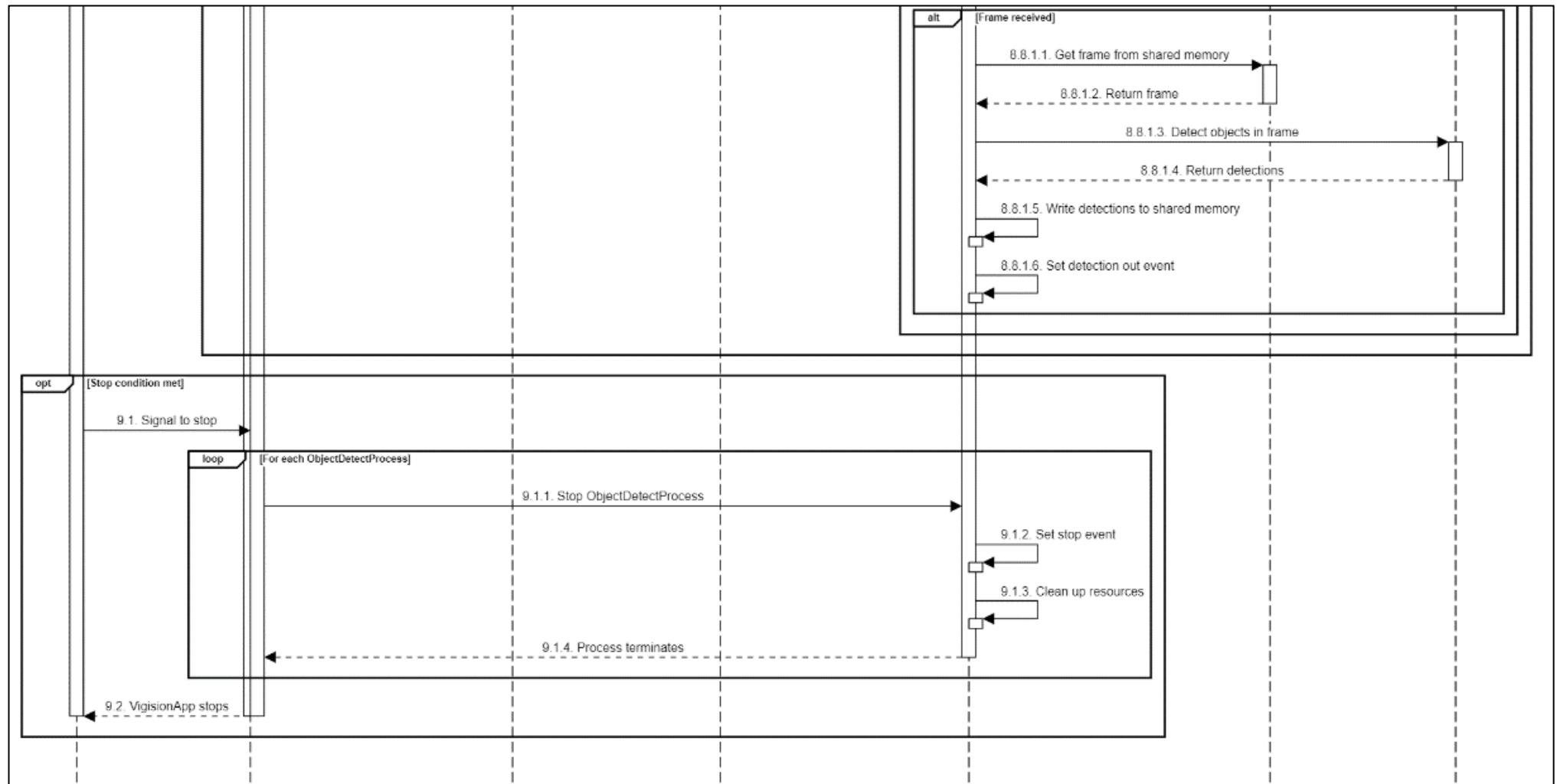


Figure 142. Sequence diagram - Object Detection Processing

2.4 Detected Frame Processing

2.4.1 Class Diagram

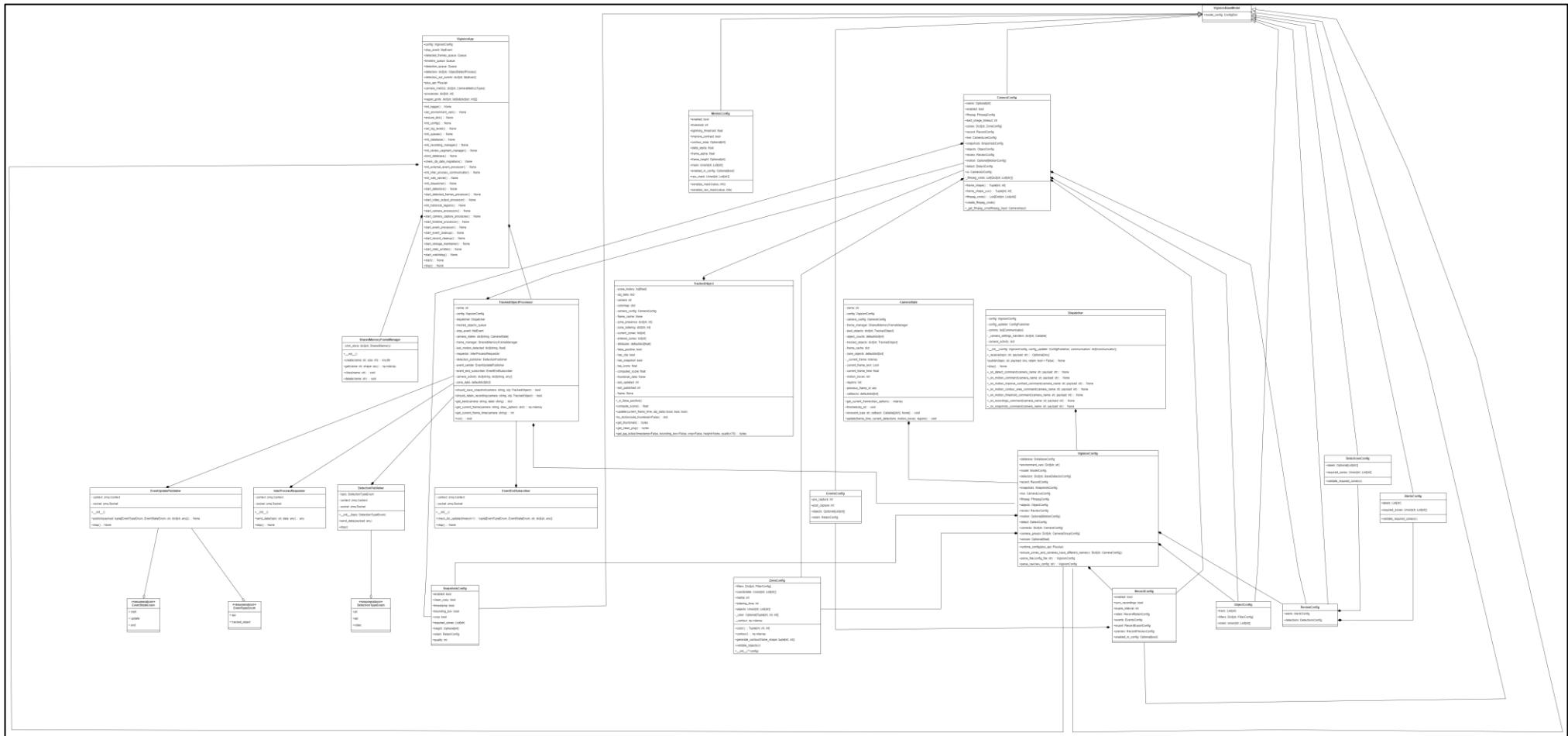
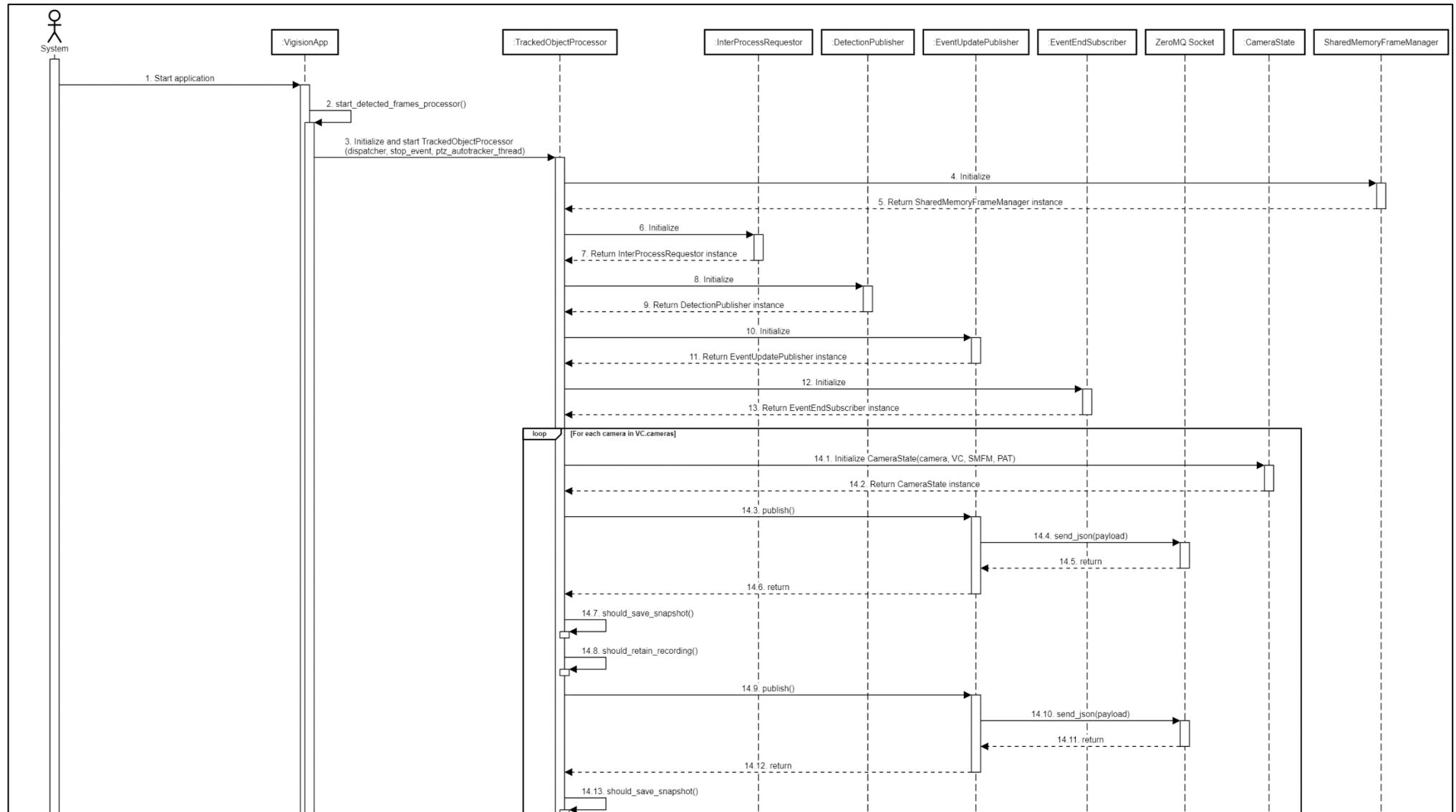


Figure 143. Class diagram - Detected Frame Processing

2.4.2 Sequence Diagram



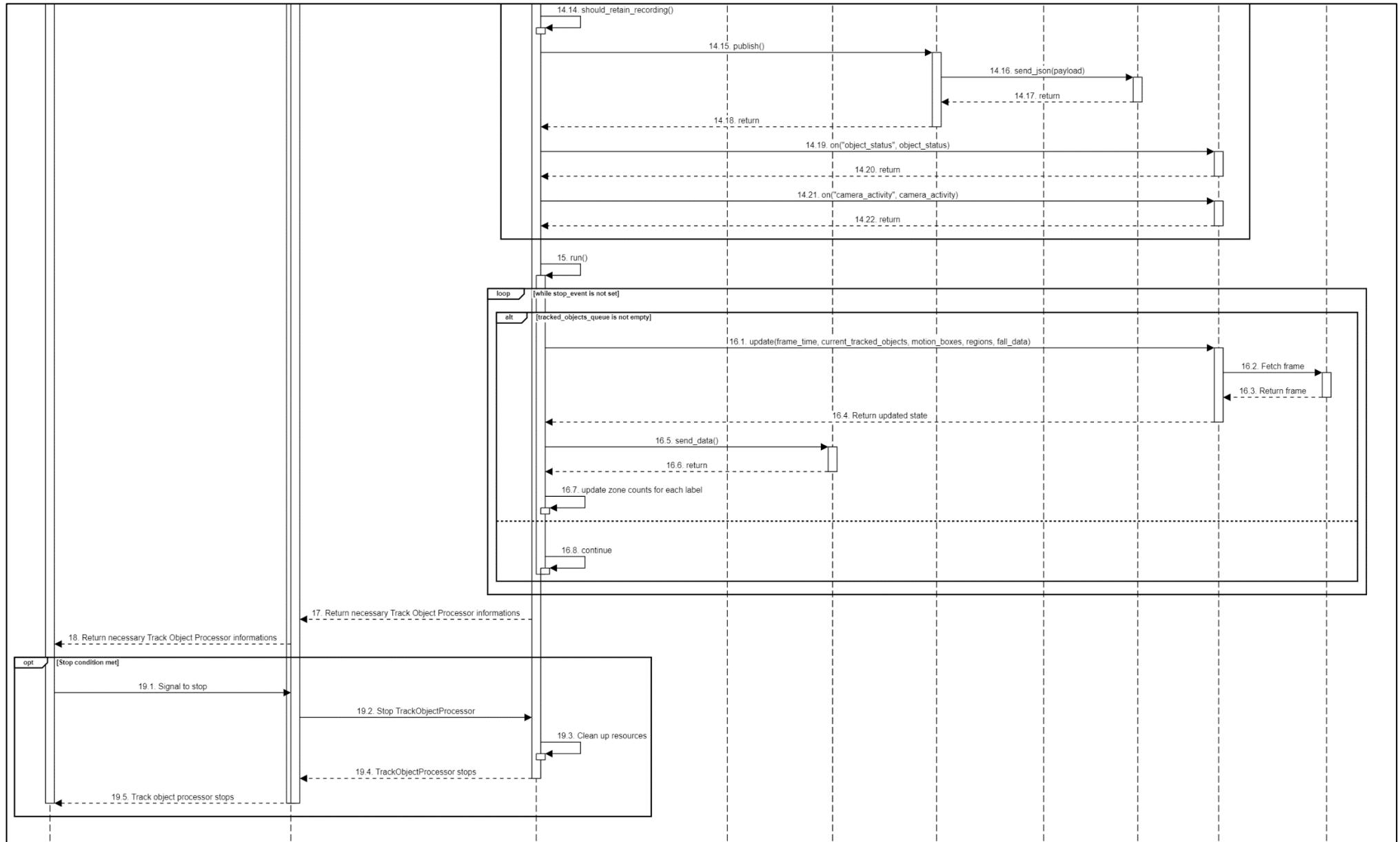


Figure 144. Sequence diagram - Detected Frame Processing

2.5 Video Output Processing

2.5.1 Class Diagram

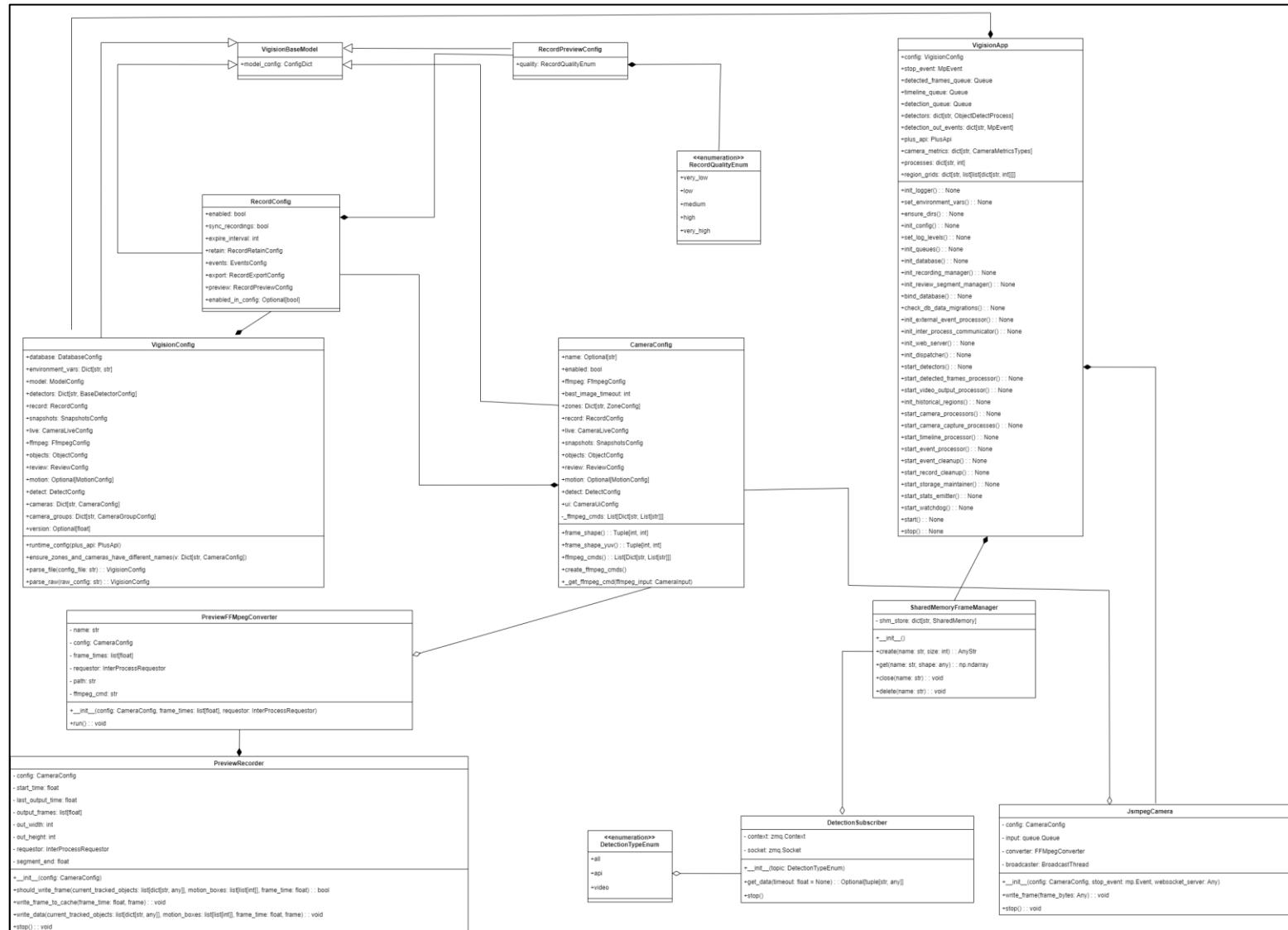
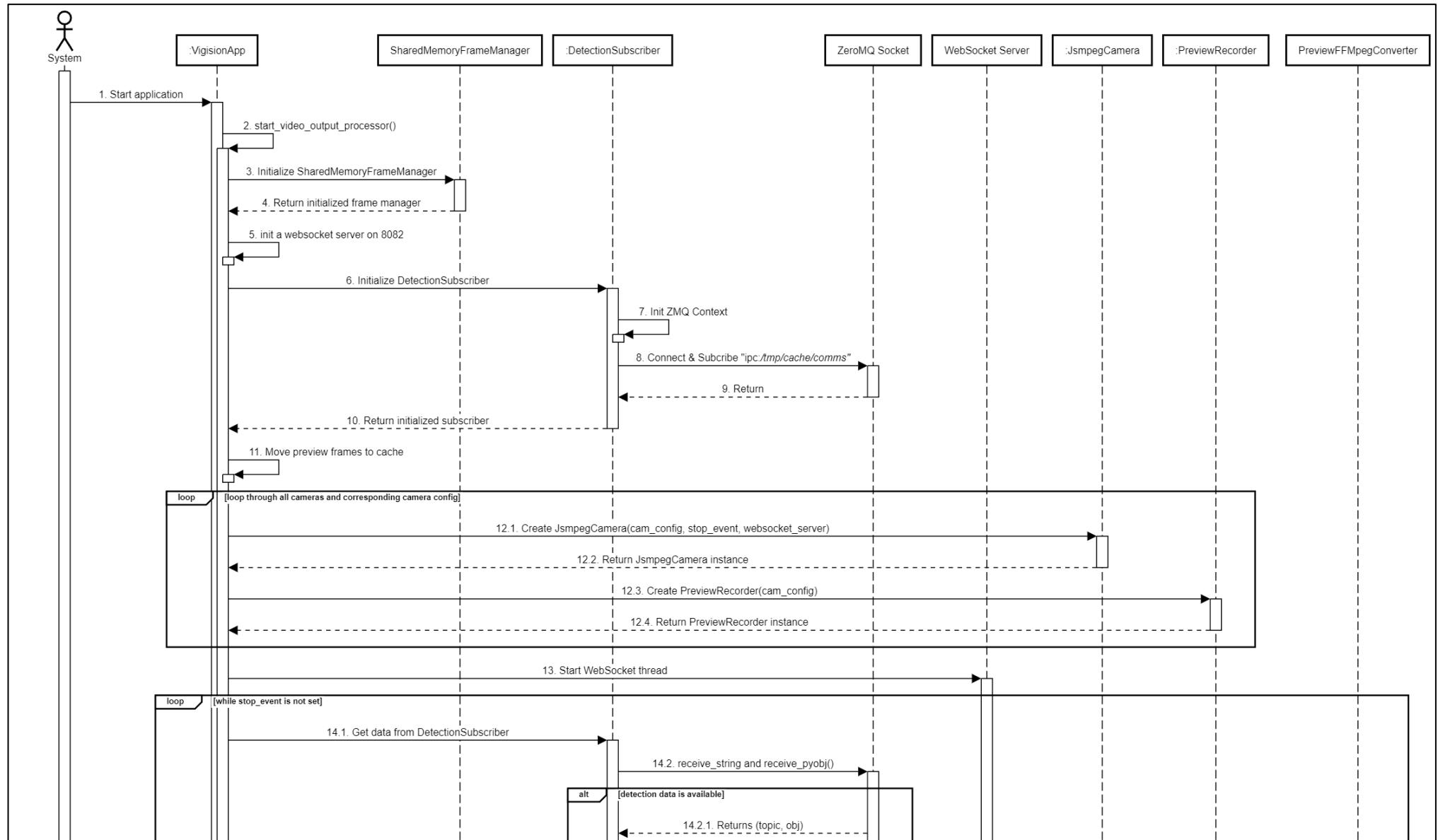


Figure 145. Class diagram - Video Output Processing

2.5.2 Sequence Diagram



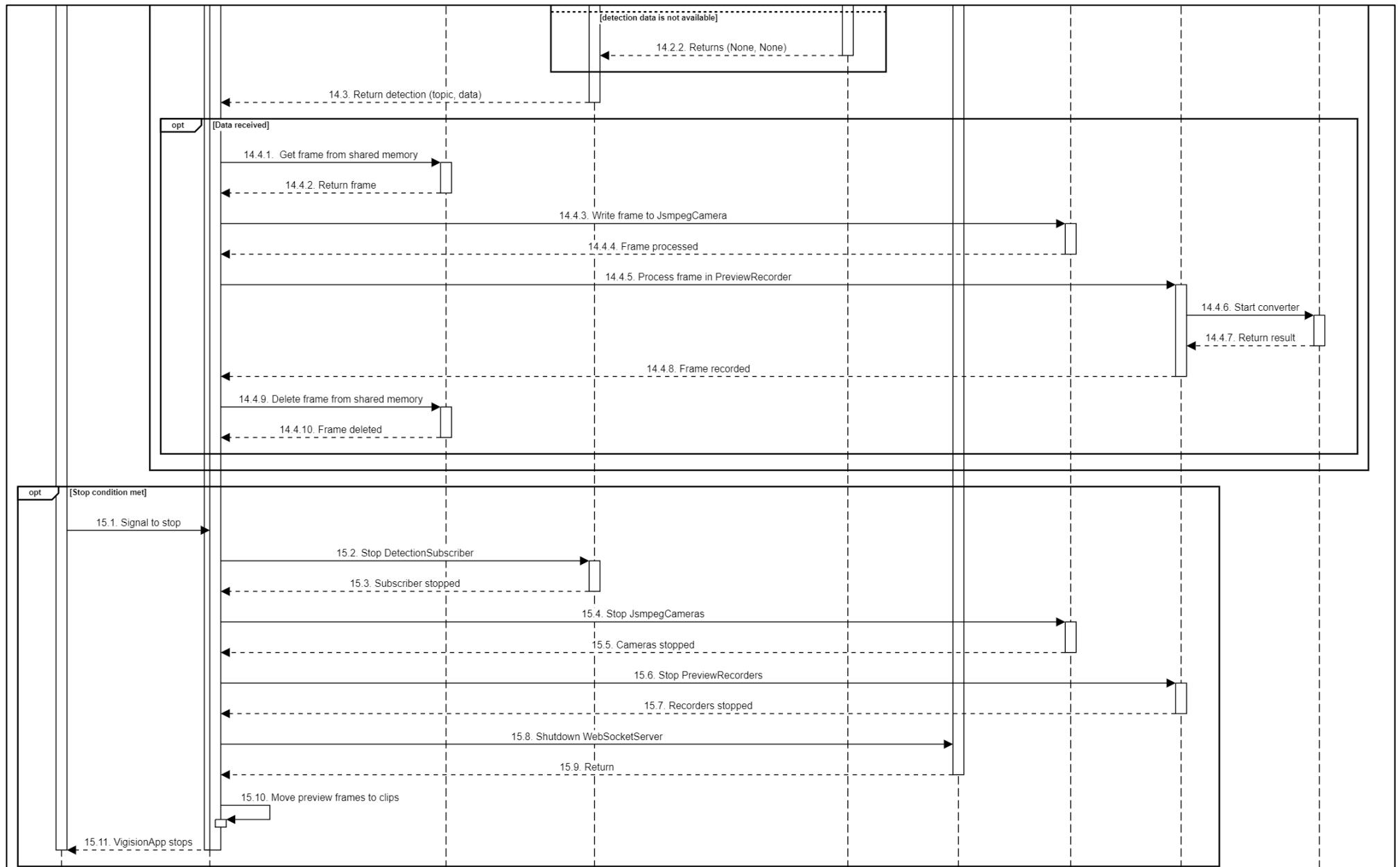


Figure 146. Class diagram - Video Output Processing

2.6 Camera frames processing

2.6.1 Class Diagram

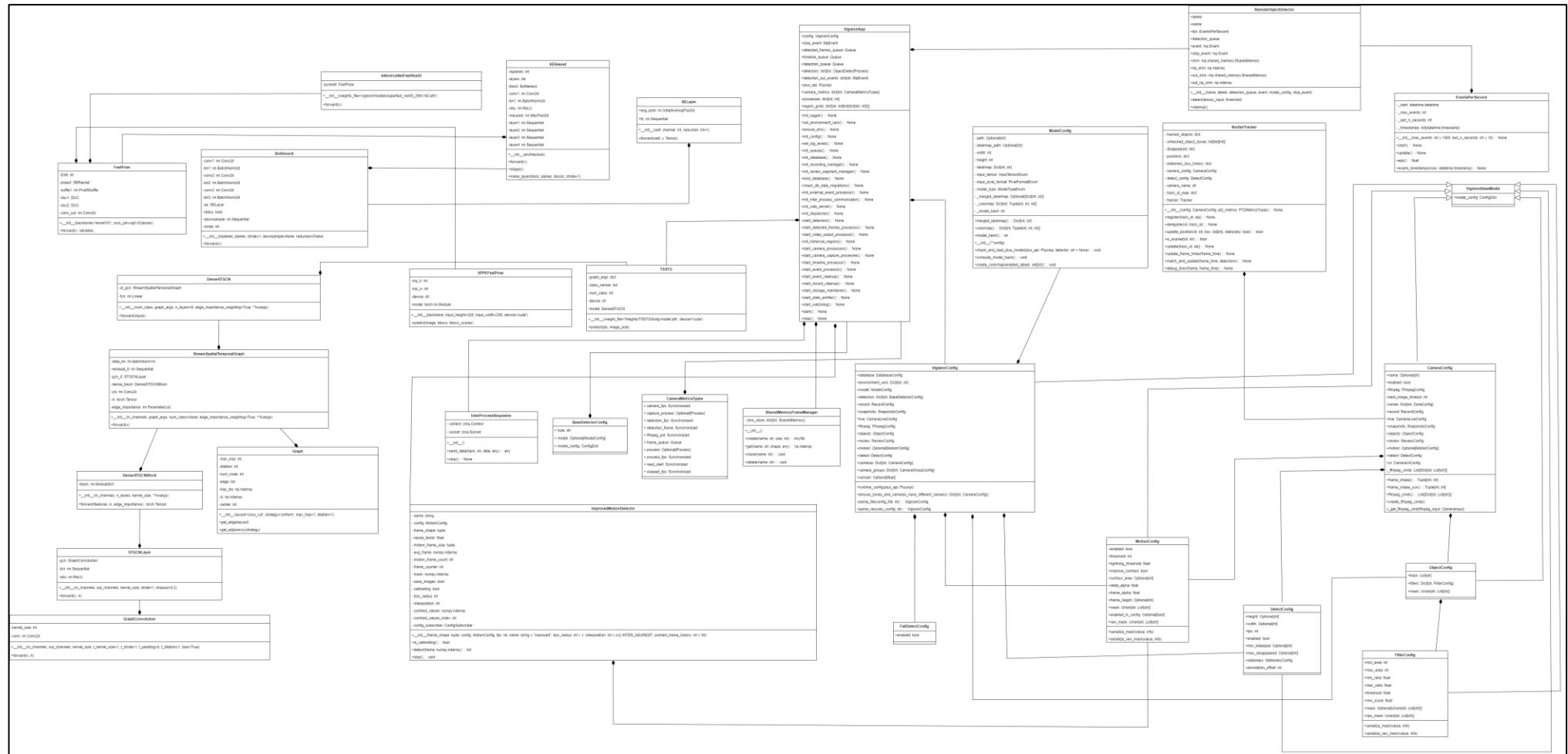
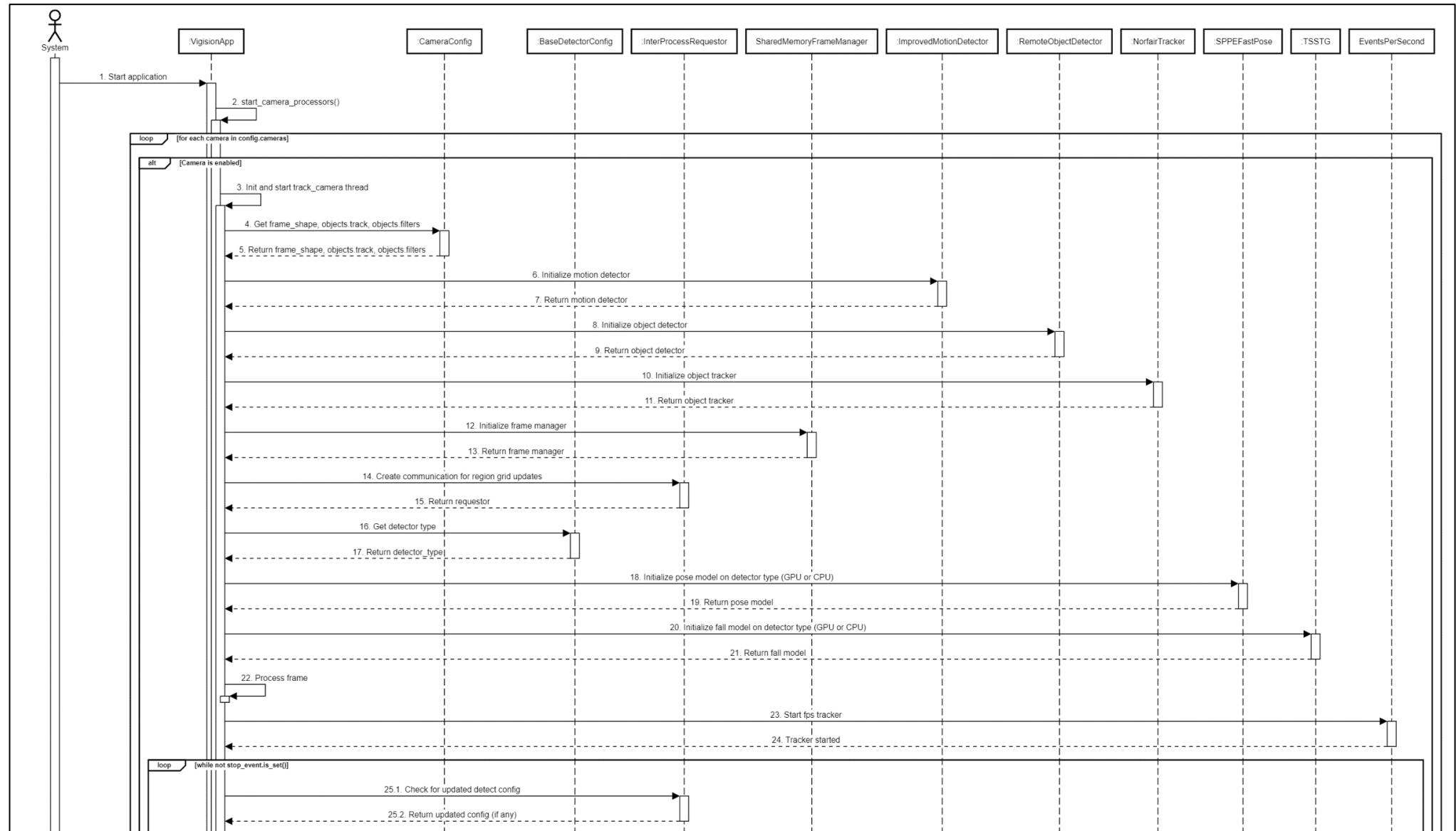
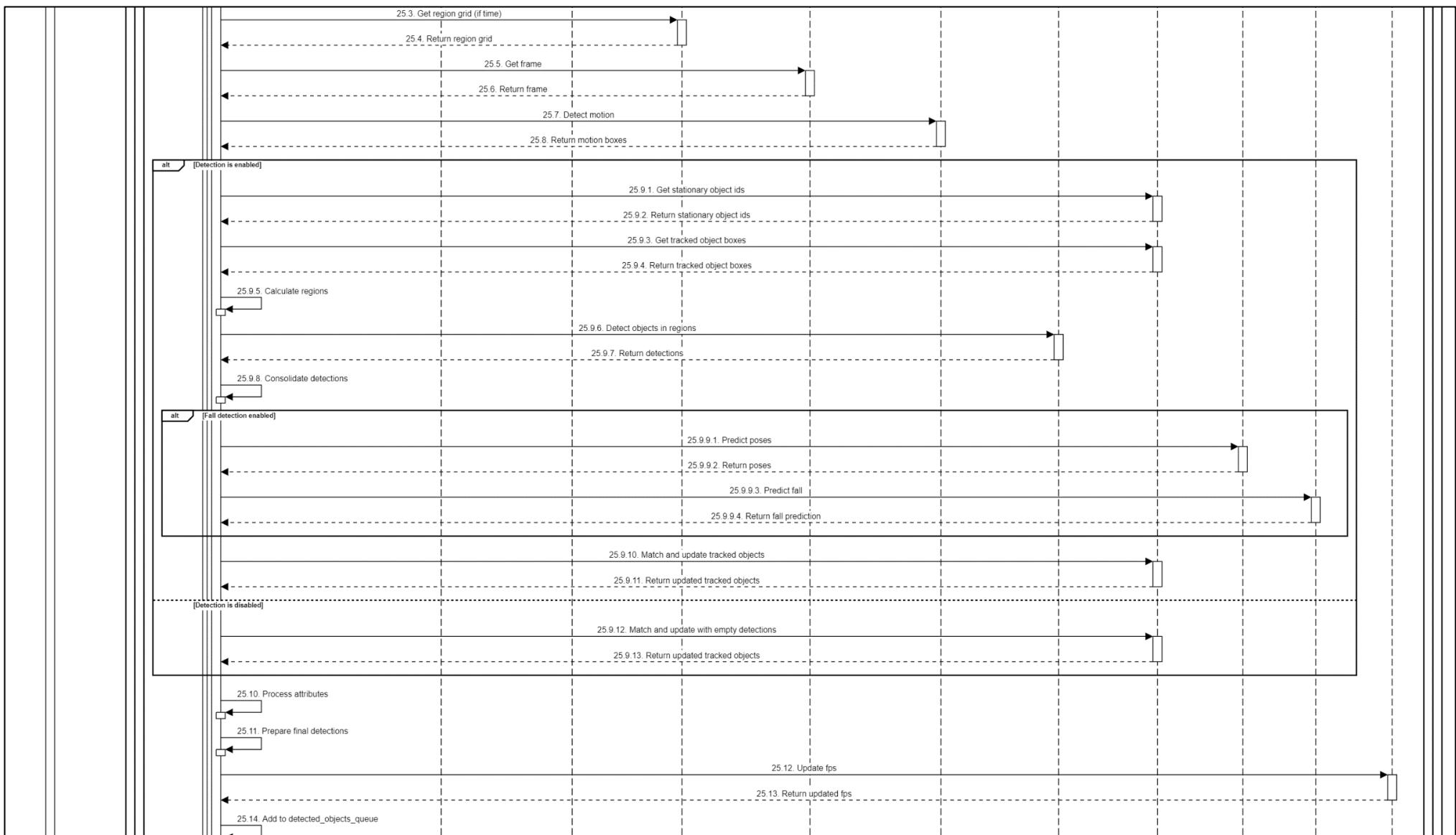


Figure 147. Class diagram - Camera frames processing

2.6.2 Sequence Diagram





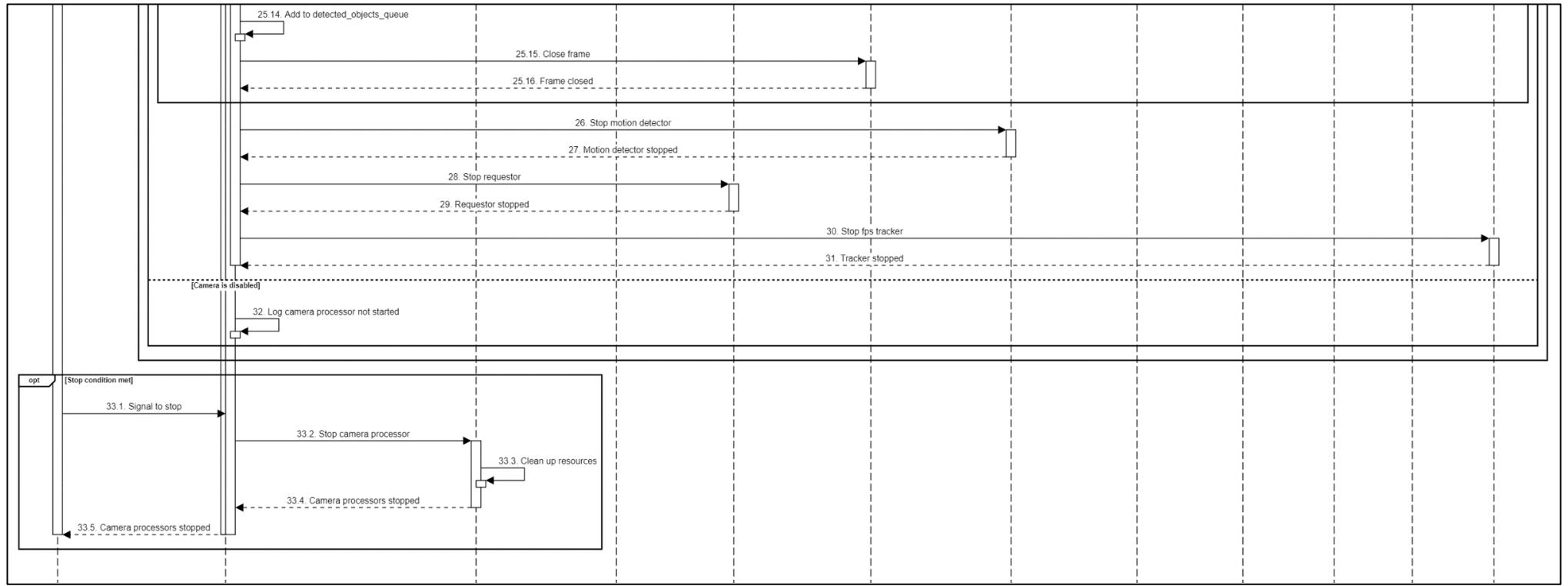


Figure 148. Sequence diagram - Camera frames processing

2.7 Camera capture processing

2.7.1 Class Diagram

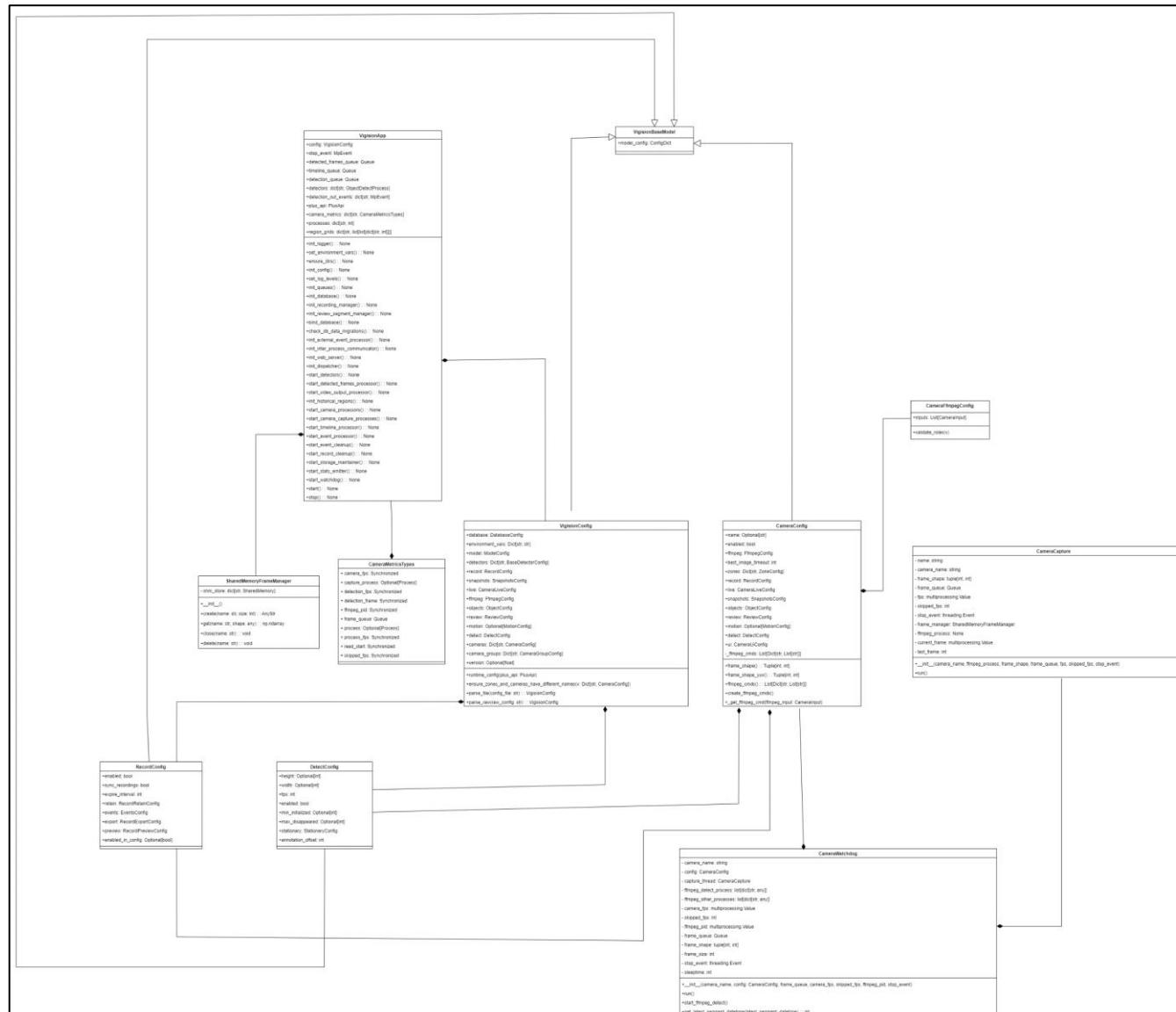
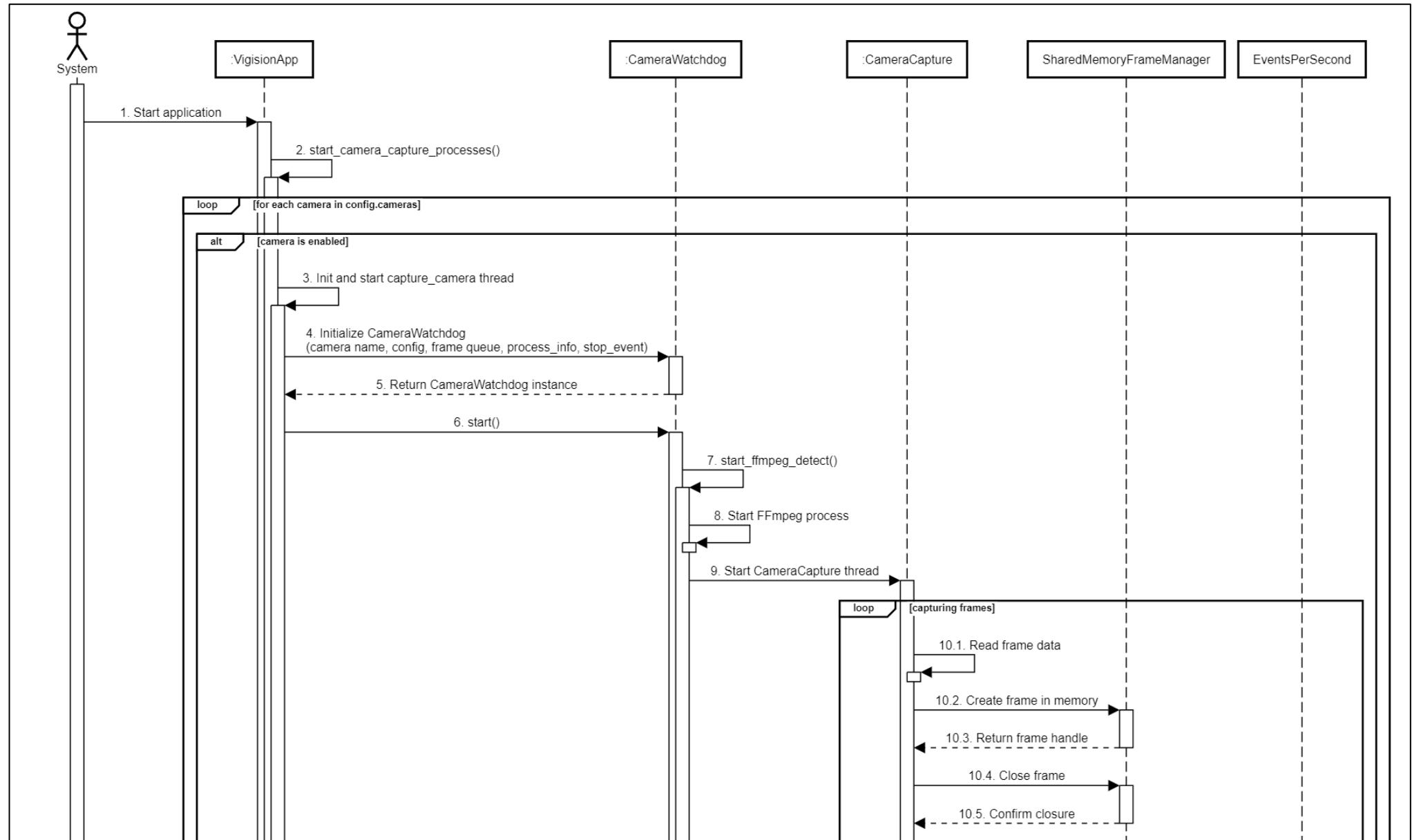
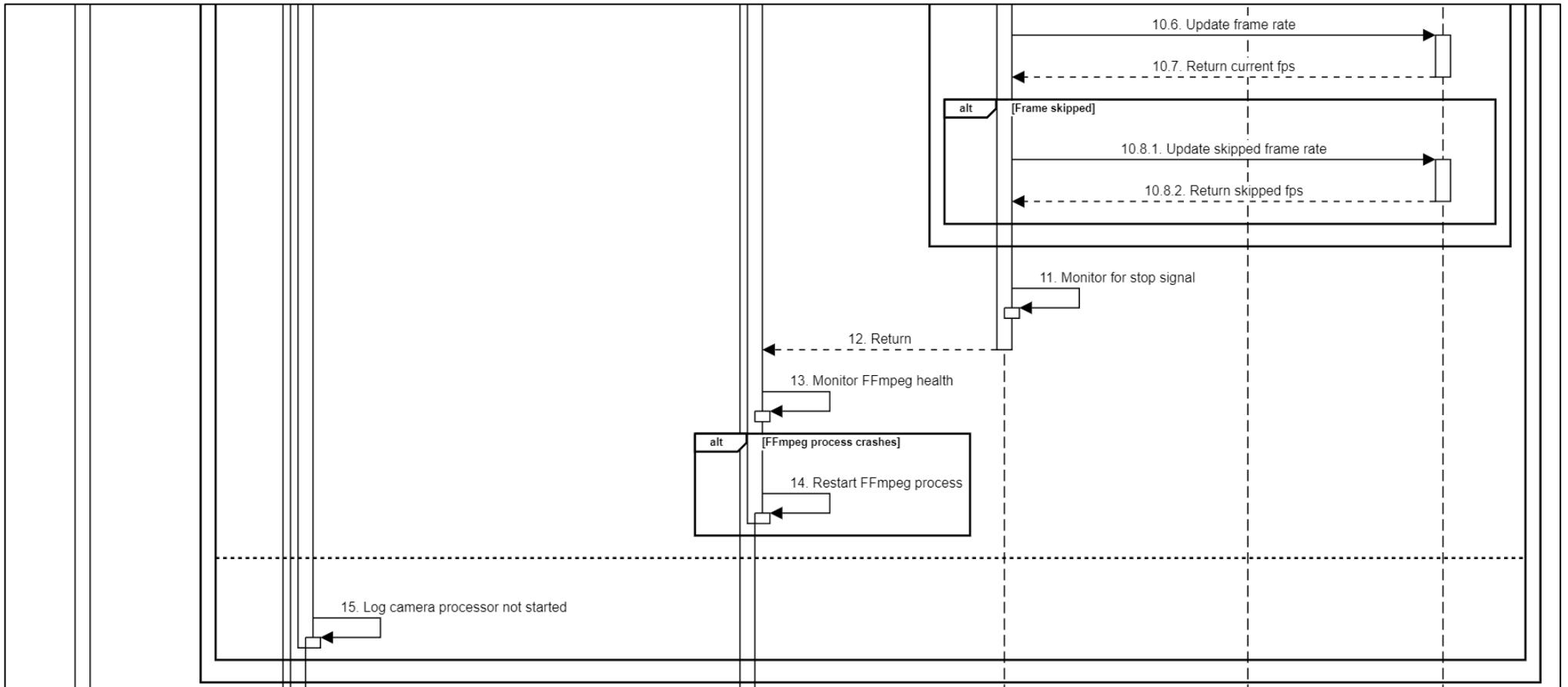


Figure 149. Class diagram - Camera capture processing

2.7.2 Sequence Diagram





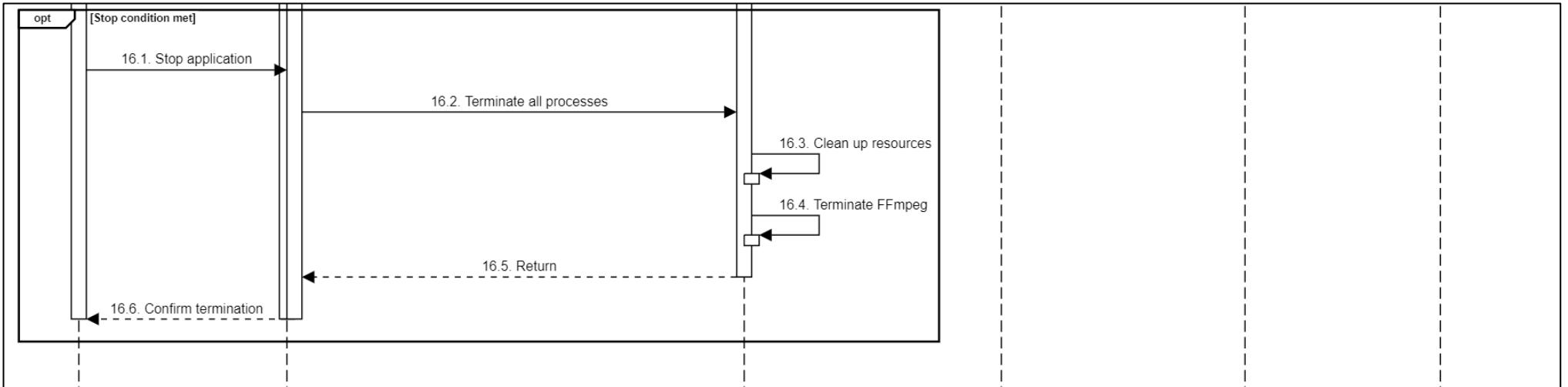


Figure 150. Sequence diagram - Camera capture processing

2.8 Timeline Processing

2.8.1 Class Diagram

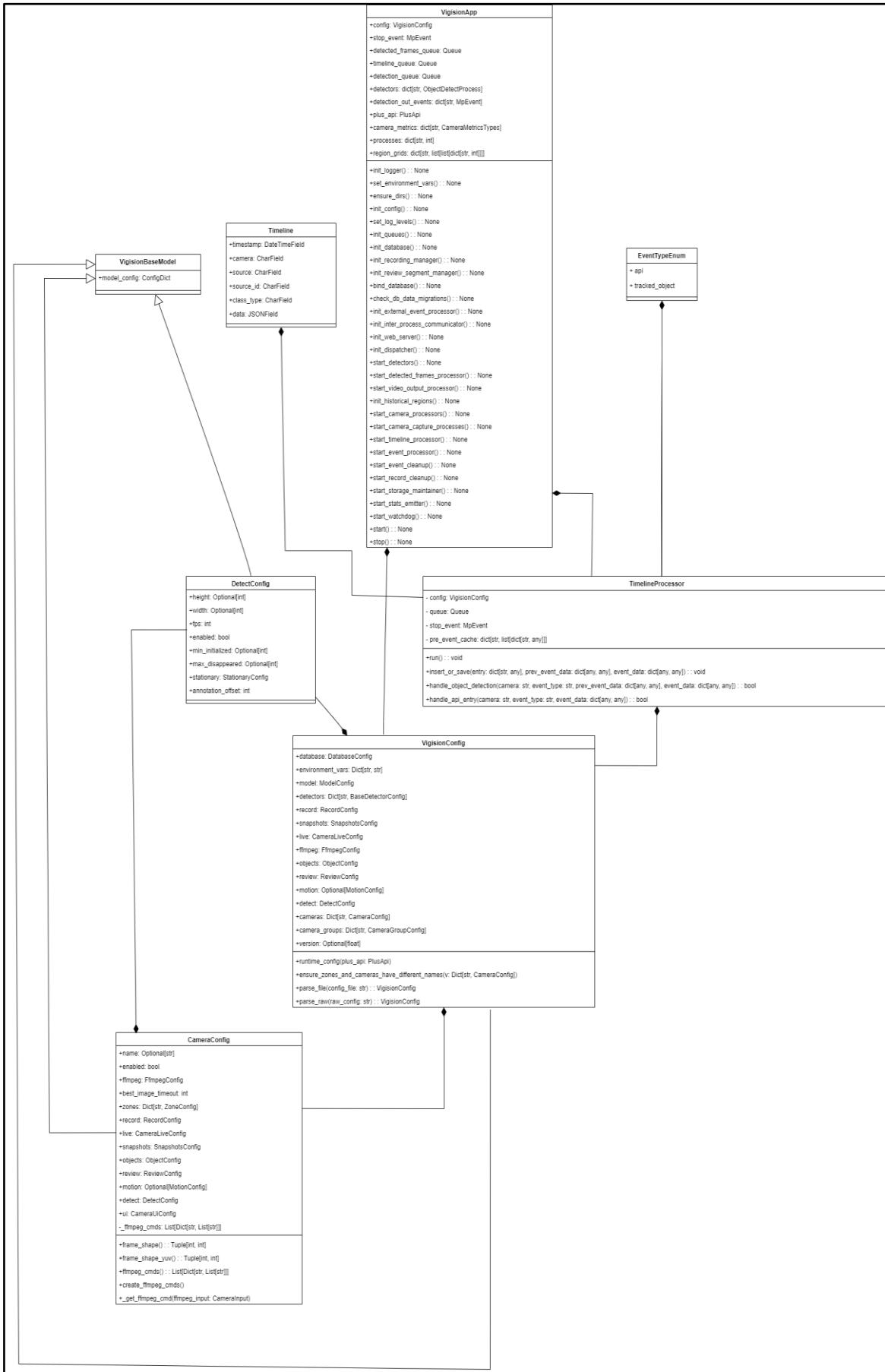
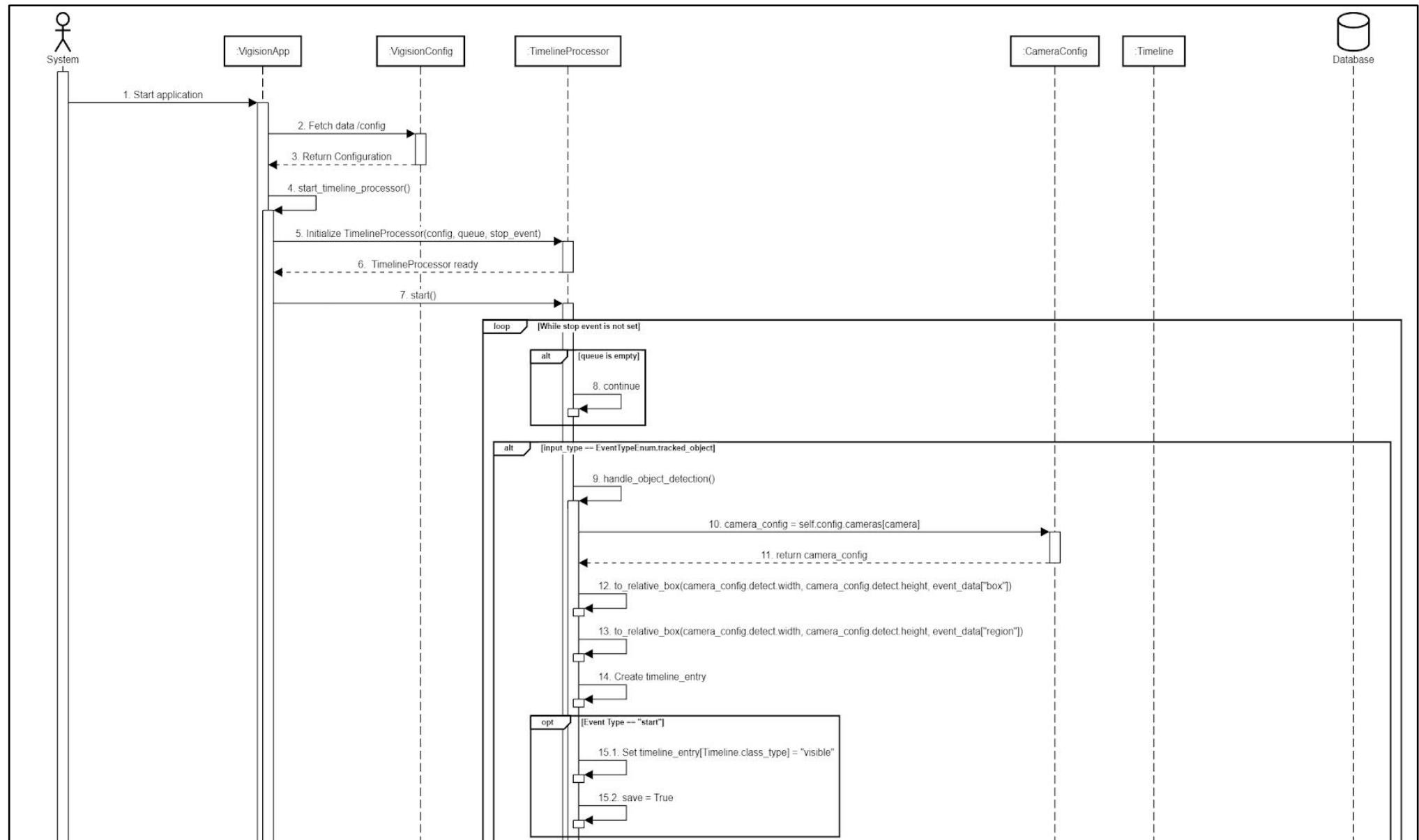
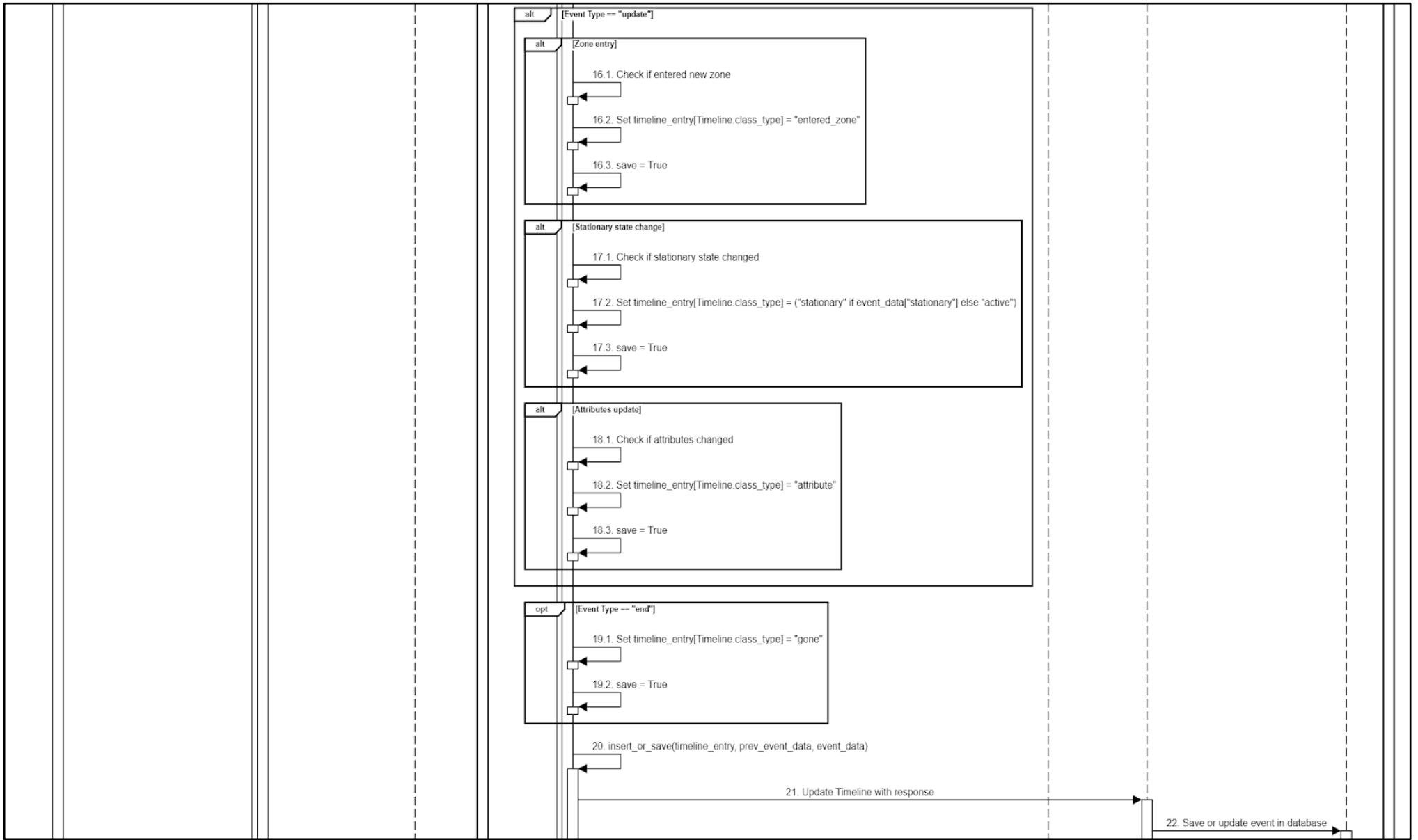


Figure 151. Class diagram - Timeline Processing

2.8.2 Sequence Diagram





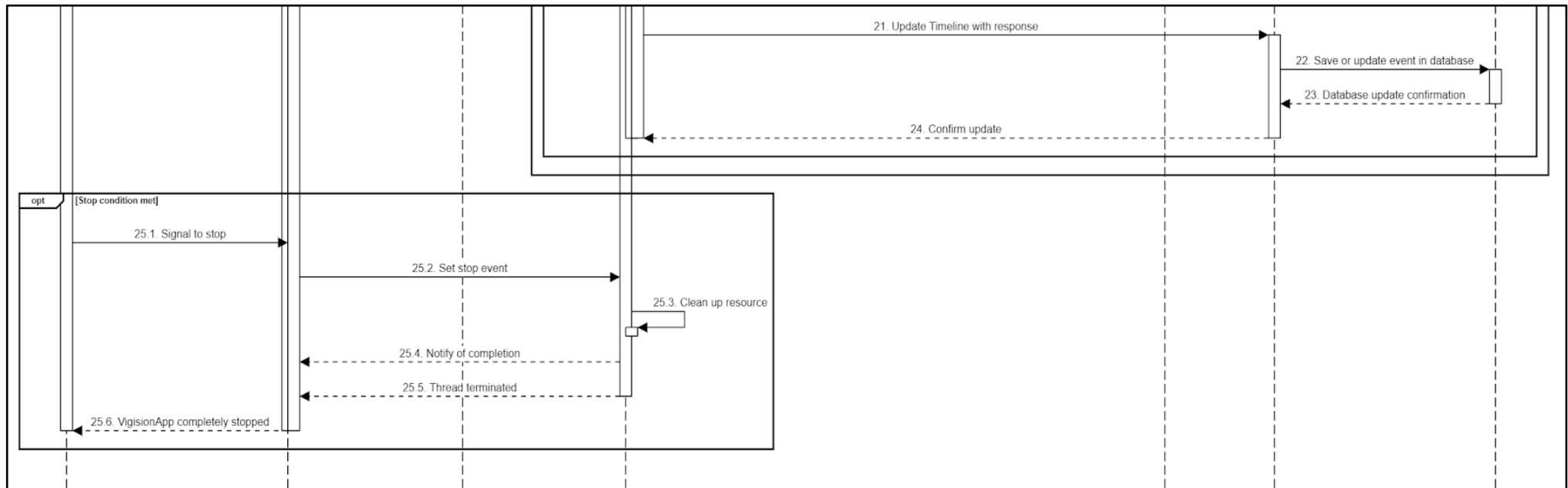


Figure 152. Sequence diagram - Timeline Processing

2.9 Event Processing

2.9.1 Class Diagram

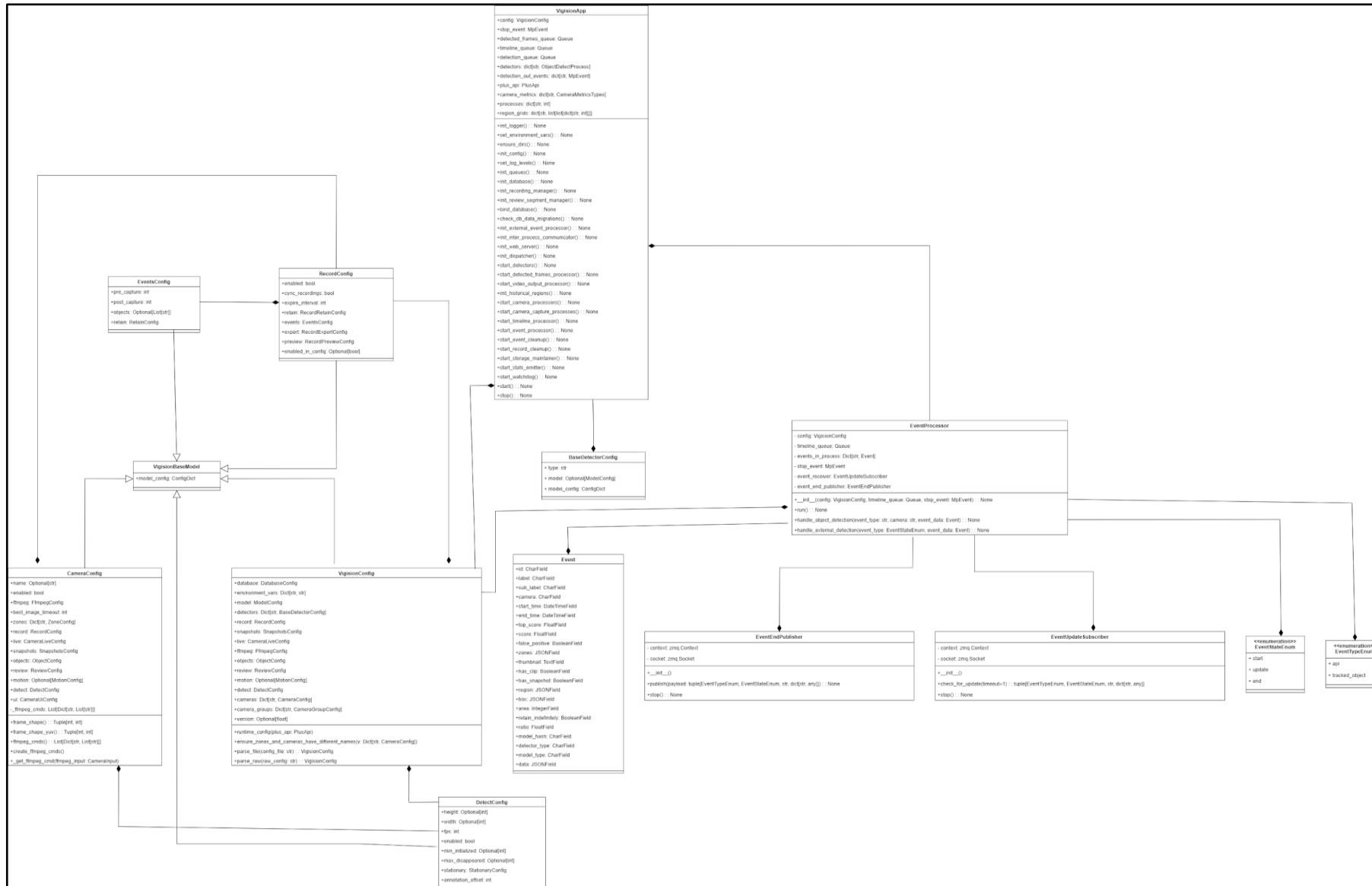
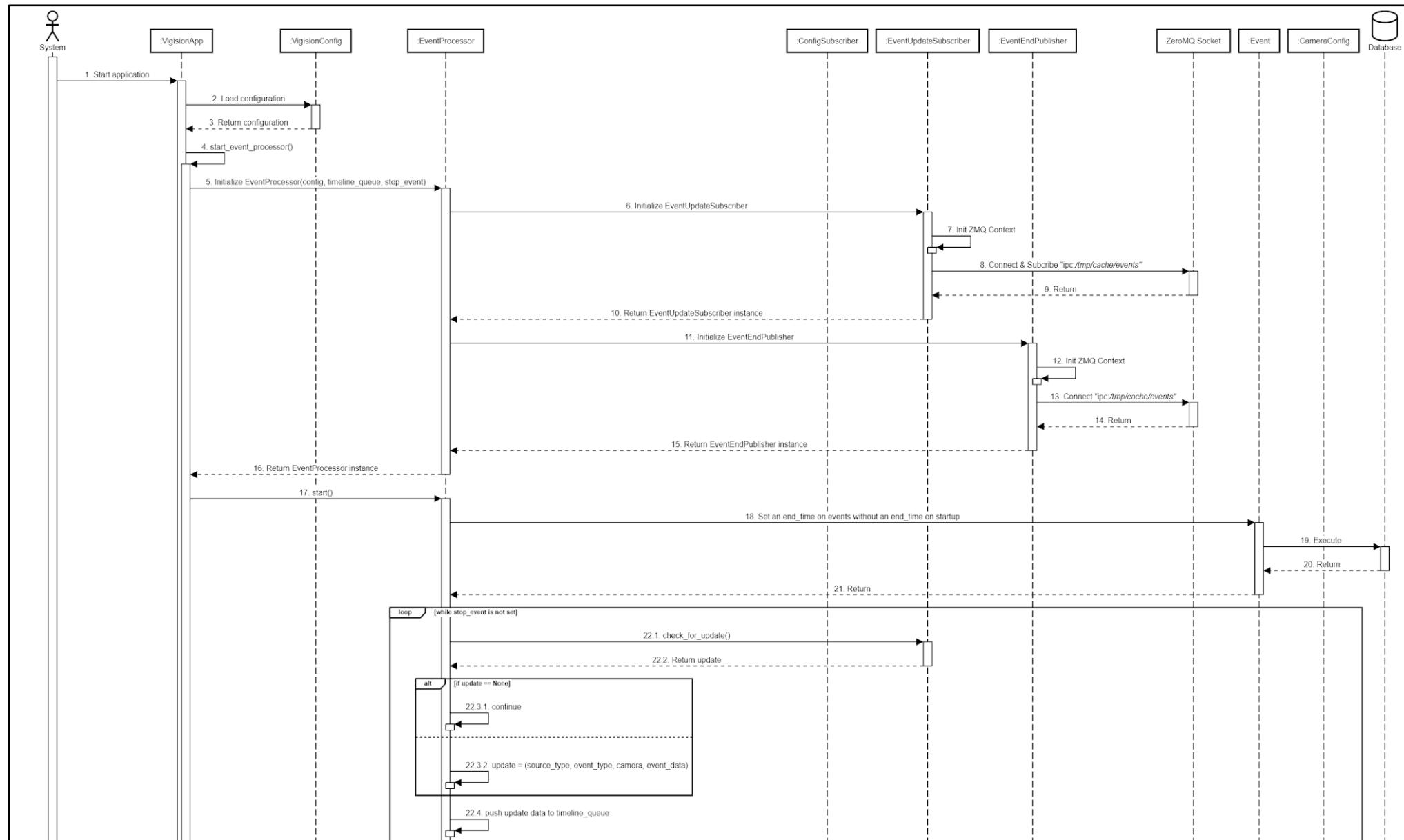


Figure 153. Class diagram - Event Processing

2.9.2 Sequence Diagram



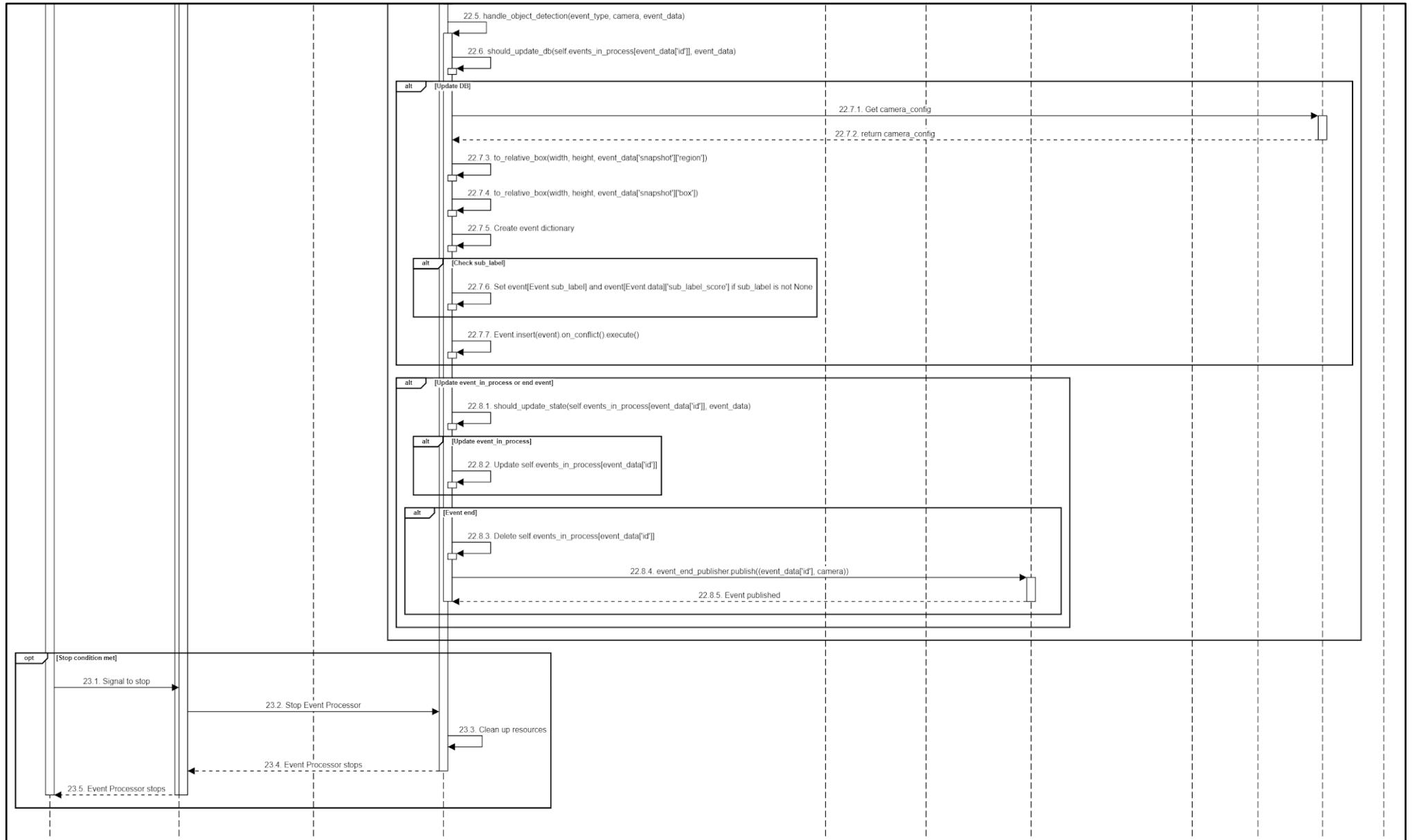


Figure 154. Sequence diagram - Event Processing

2.10 Event Cleanup Processing

2.10.1 Class Diagram

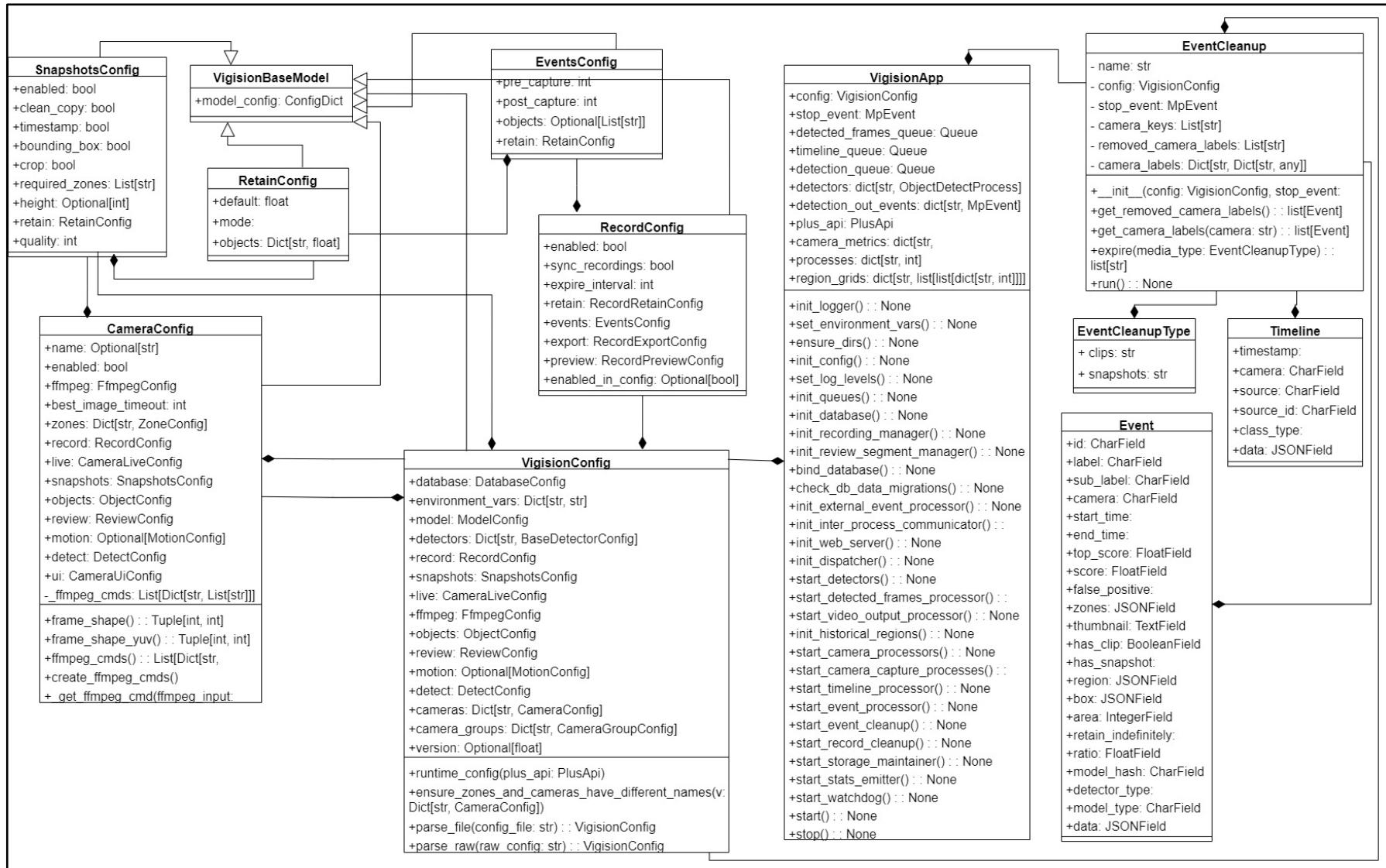
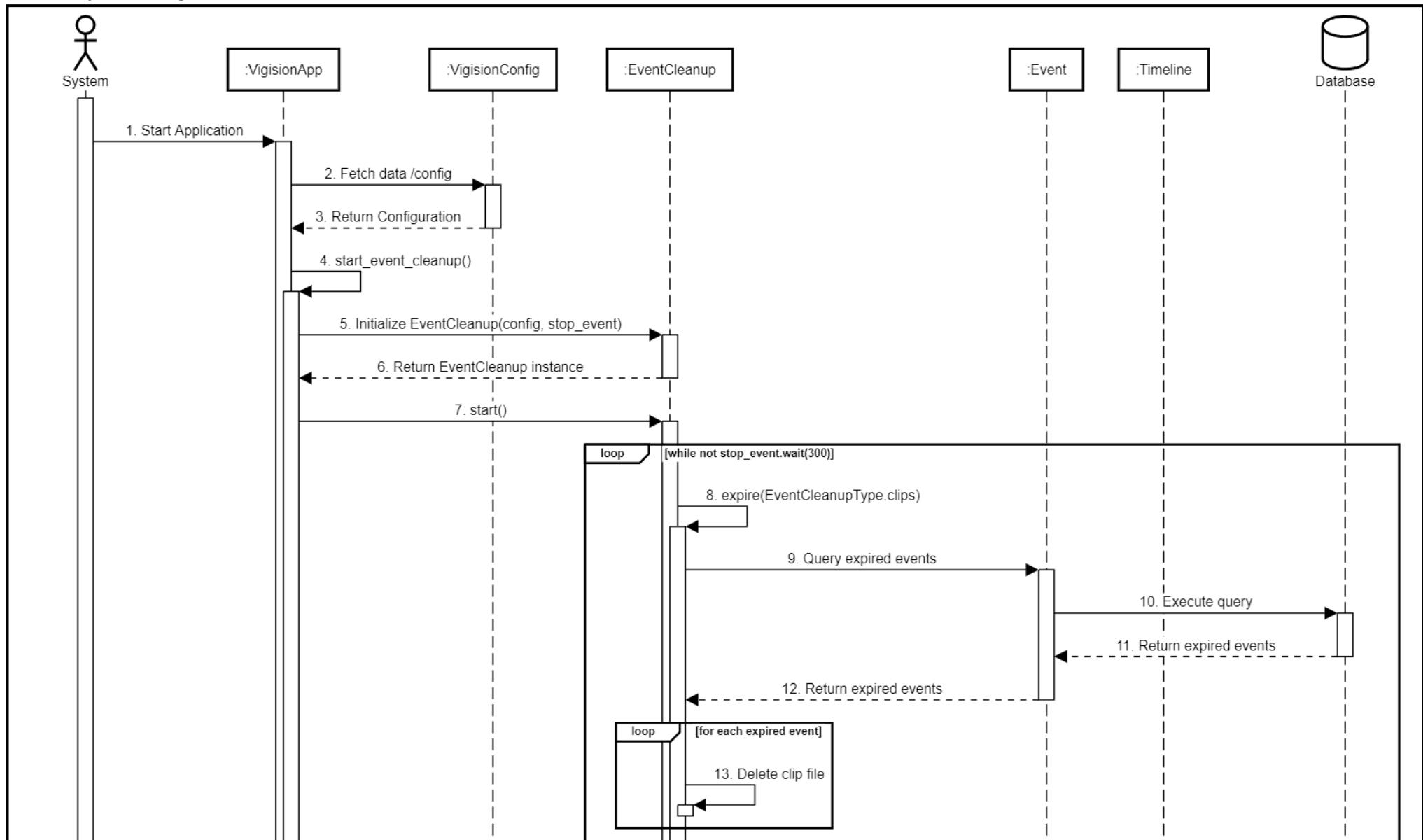
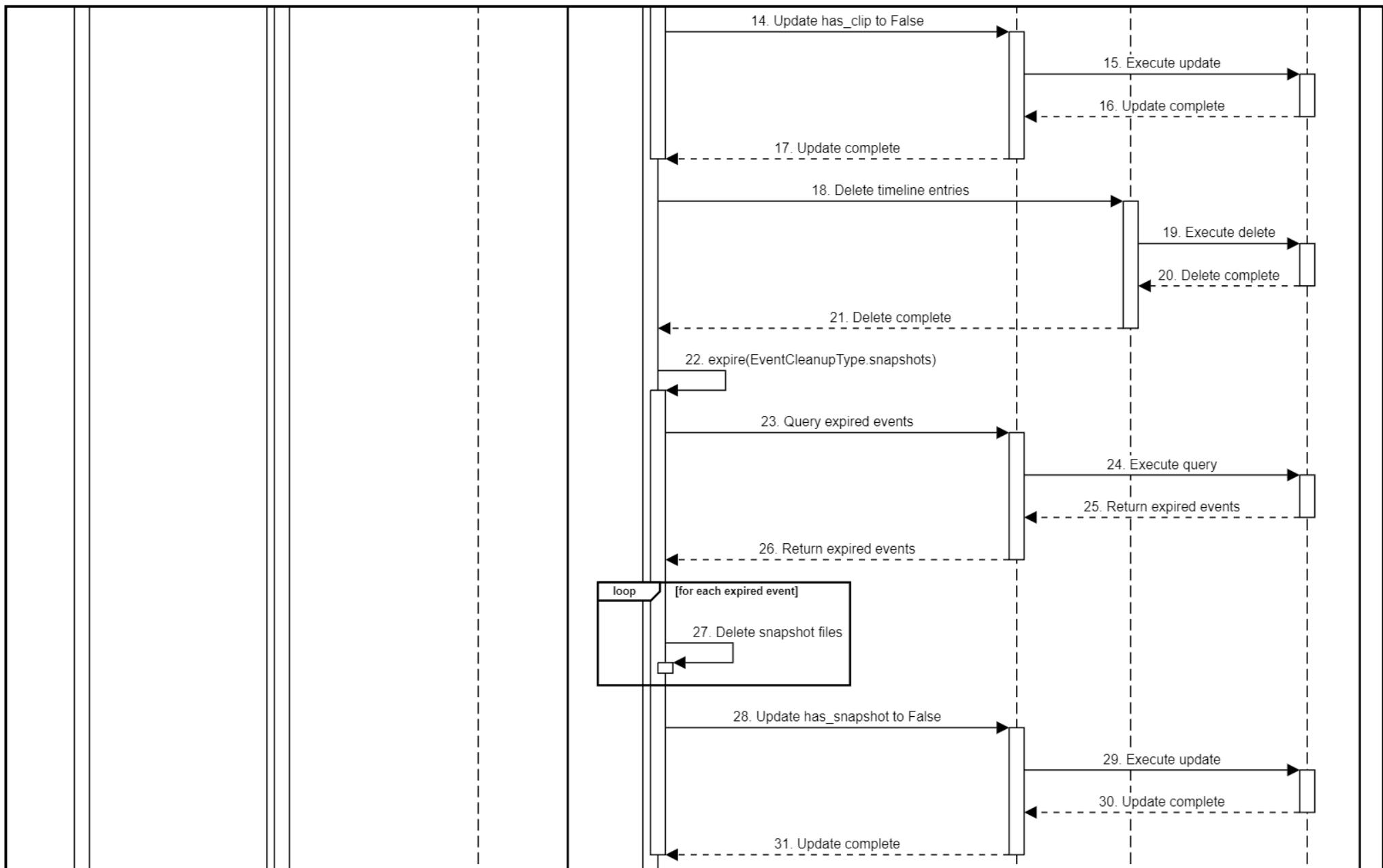


Figure 155. Class diagram - Event Cleanup Processing

2.10.2 Sequence Diagram





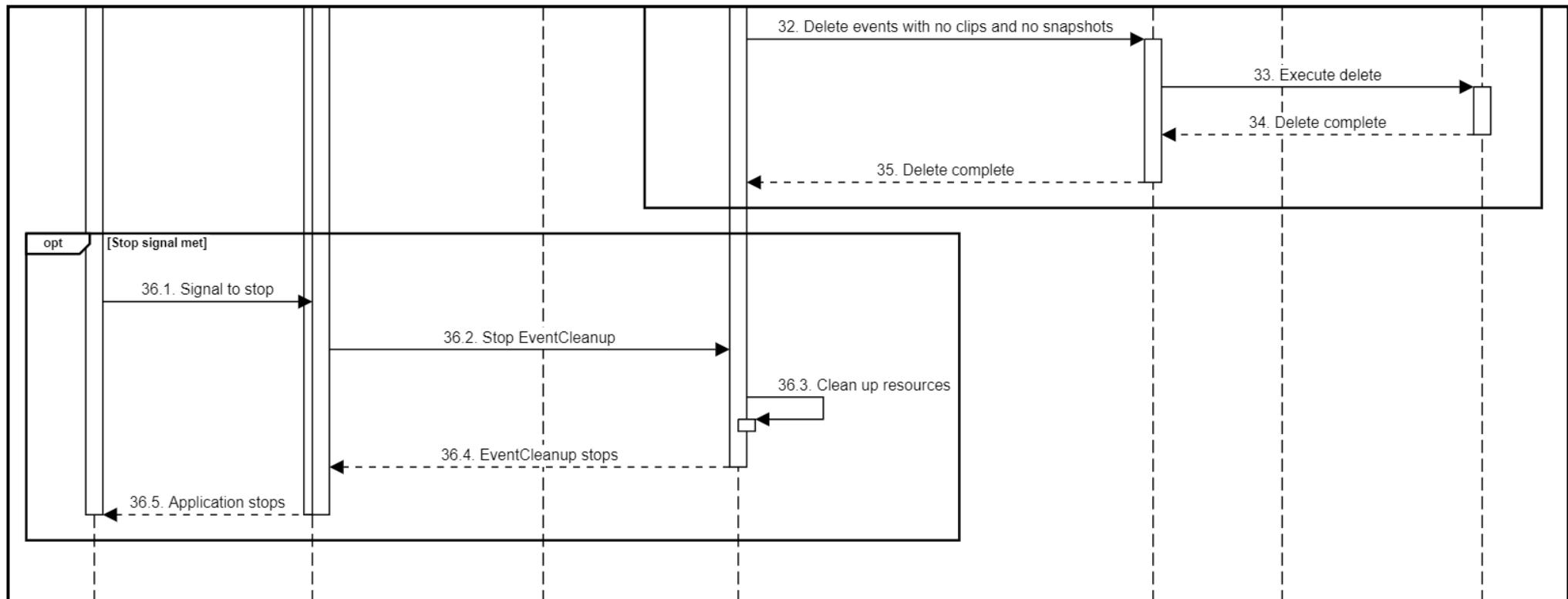


Figure 156. Sequence diagram - Event Cleanup Processing

2.11 Recording Cleanup Processing

2.11.1 Class Diagram

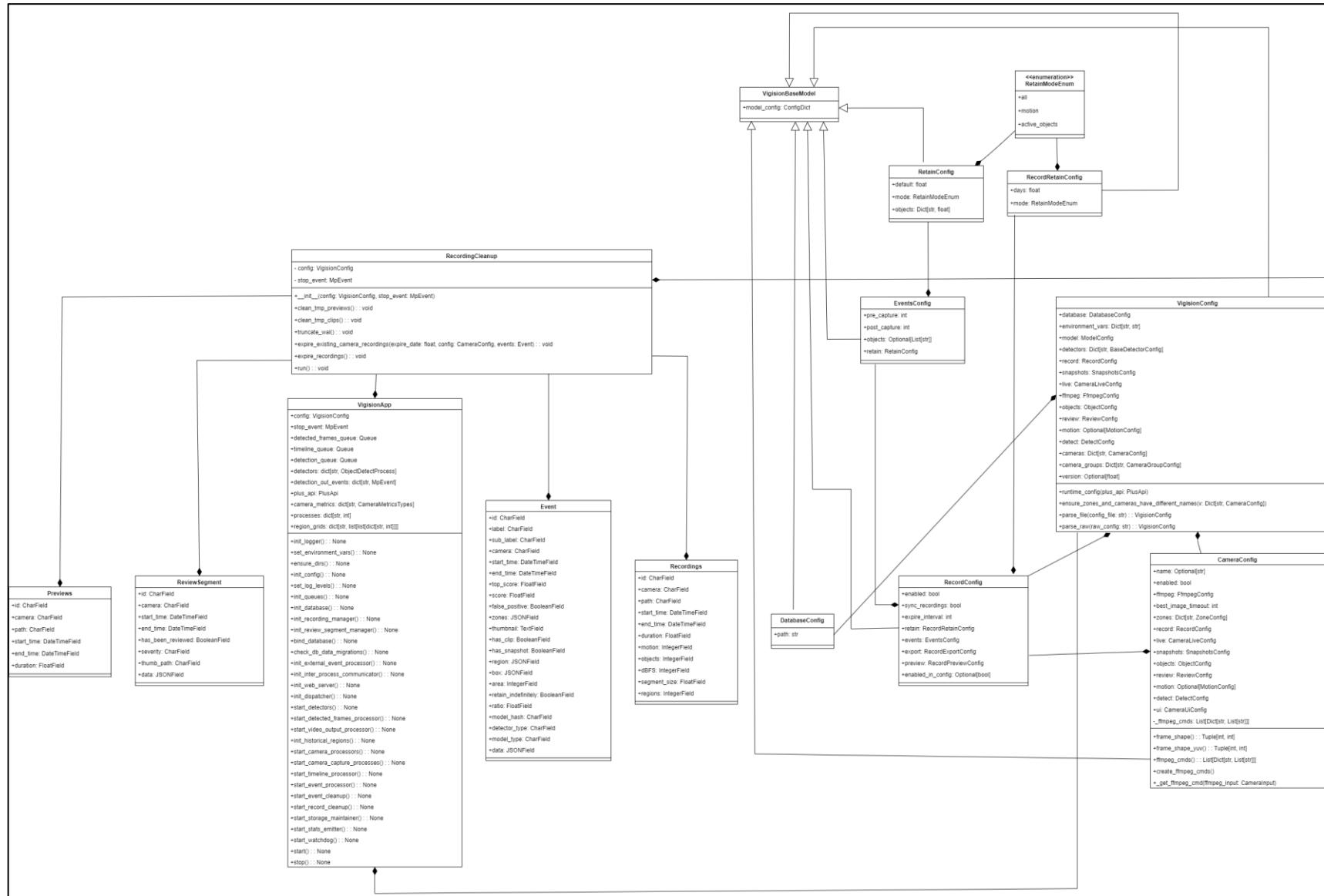
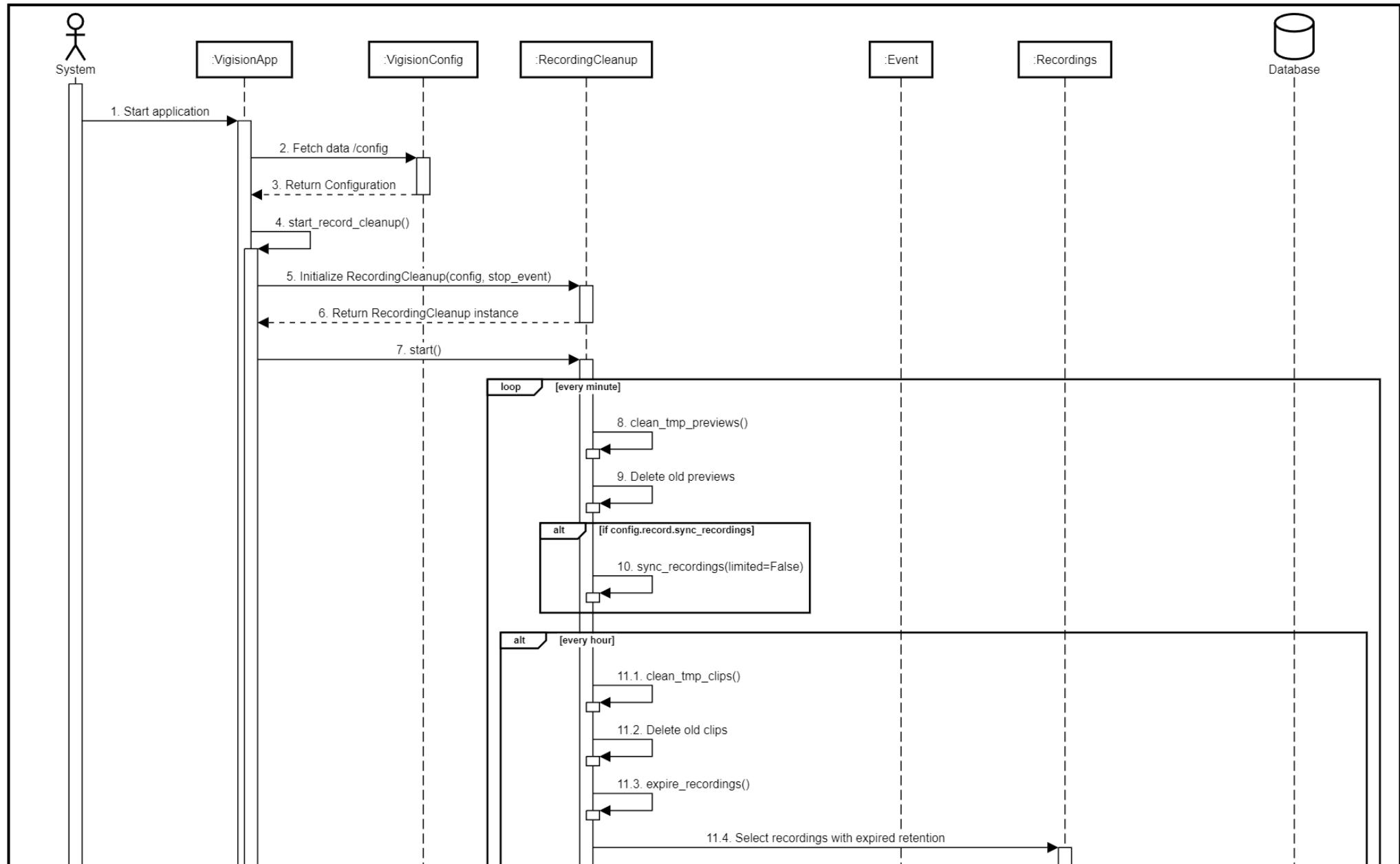
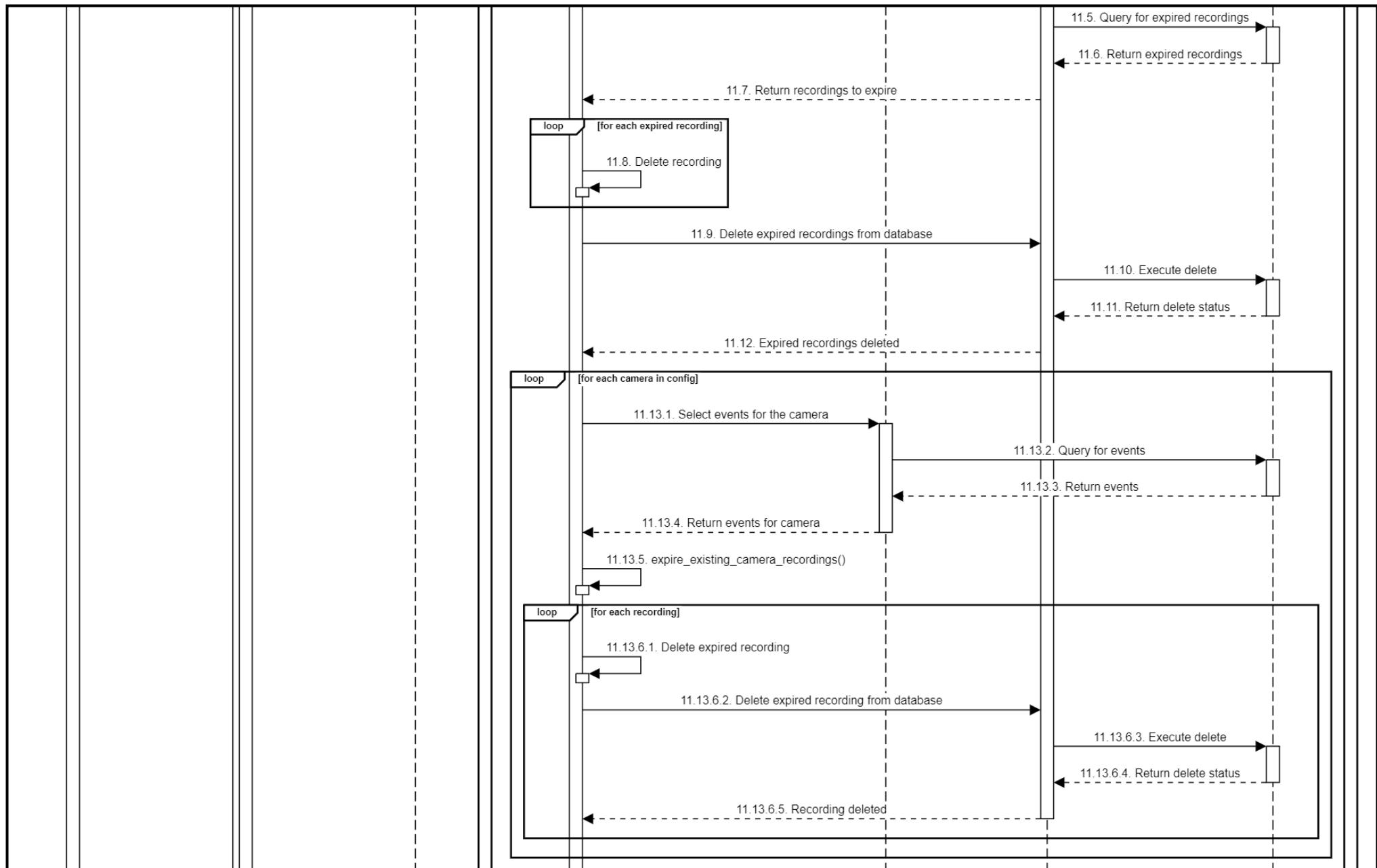


Figure 157. Class diagram - Recording Cleanup Processing

2.11.2 Sequence Diagram





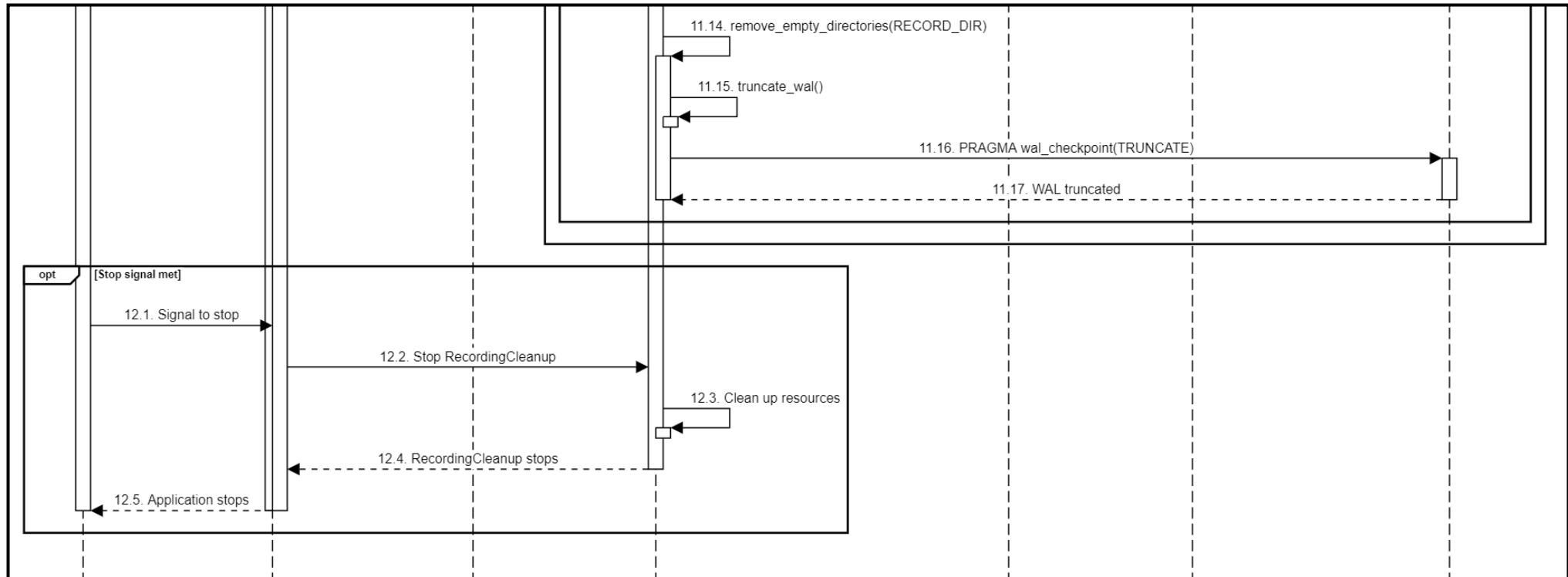


Figure 158. Sequence diagram - Recording Cleanup Processing

2.12 Storage Maintenance Processing

2.12.1 Class Diagram

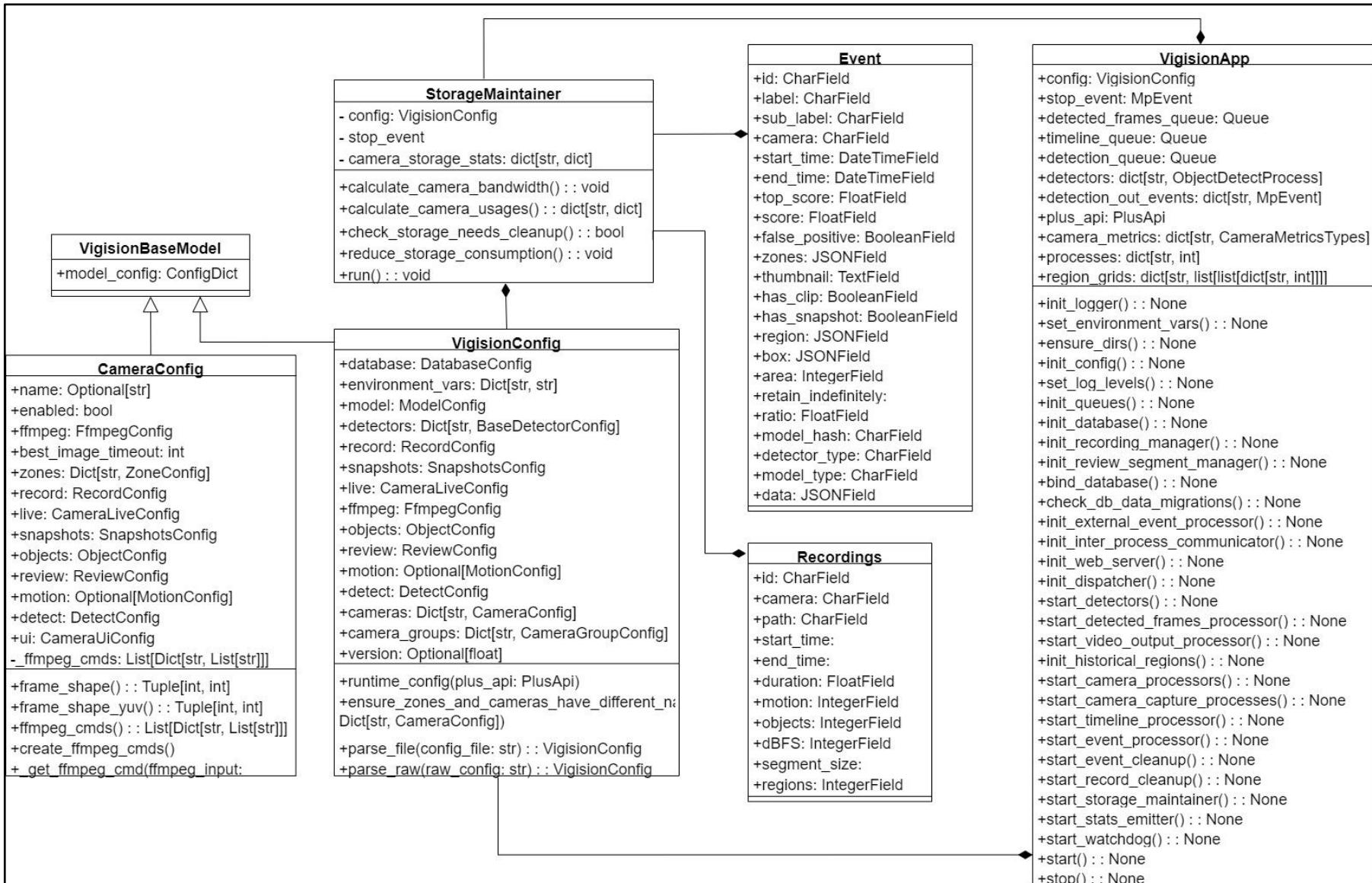
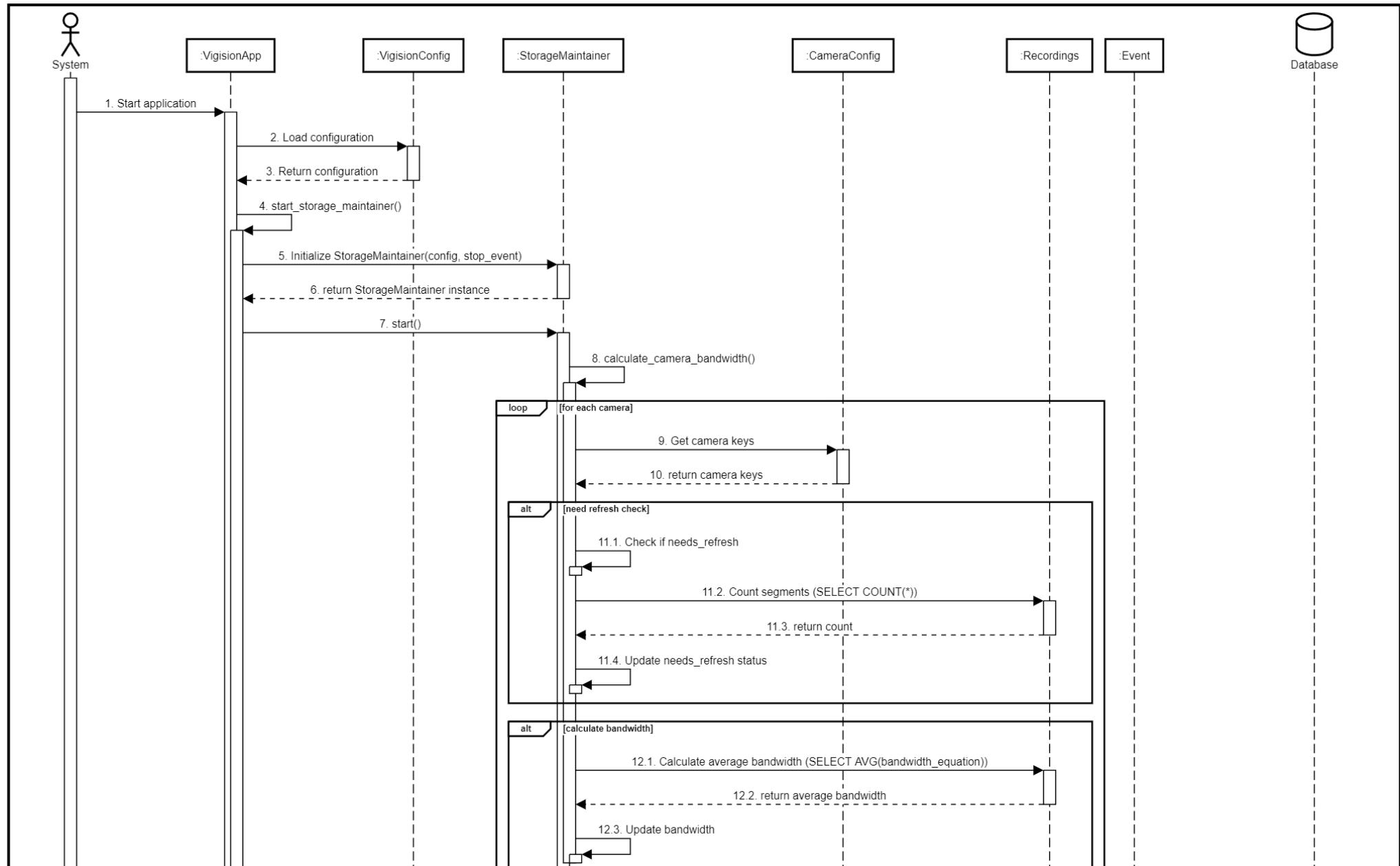
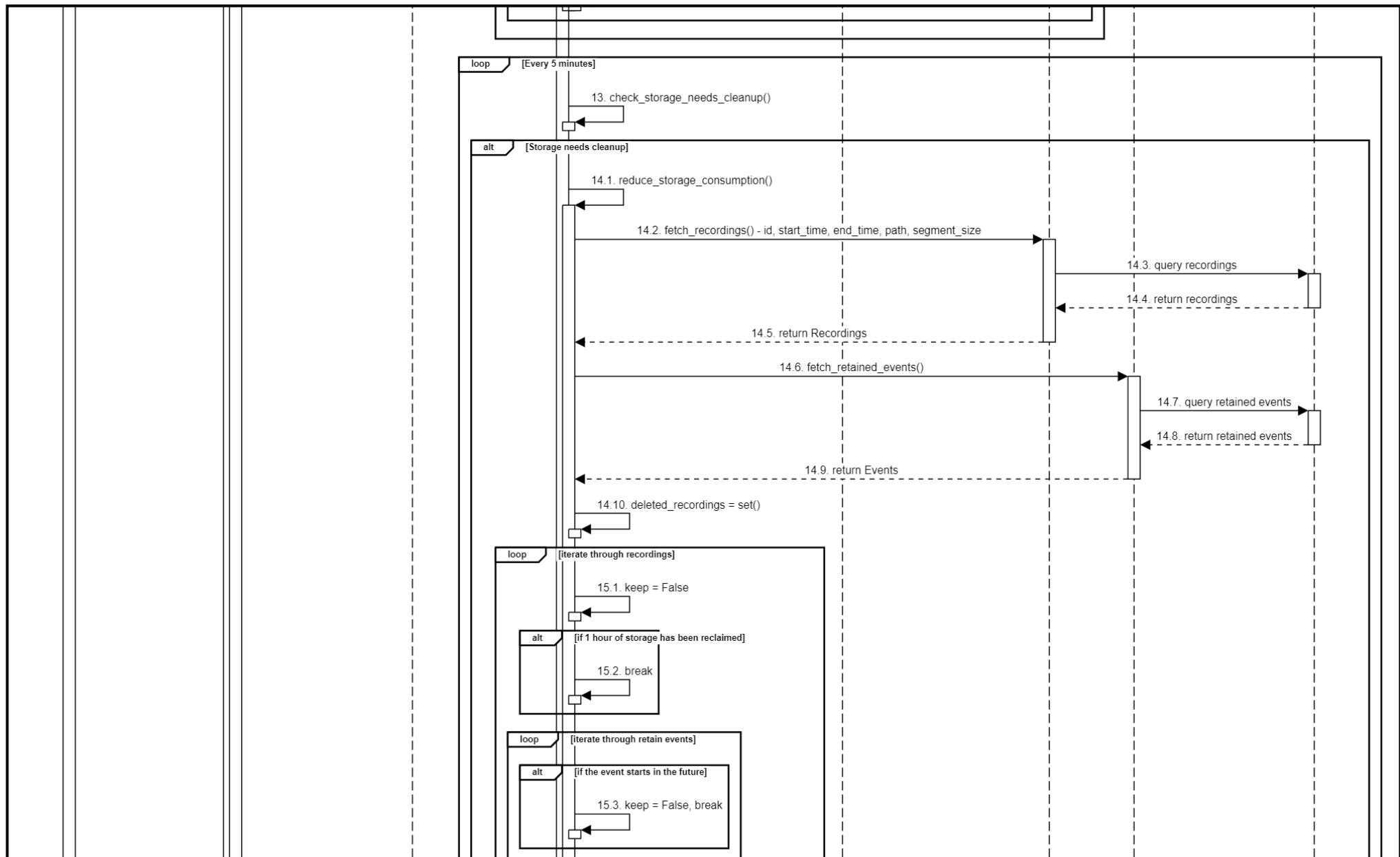
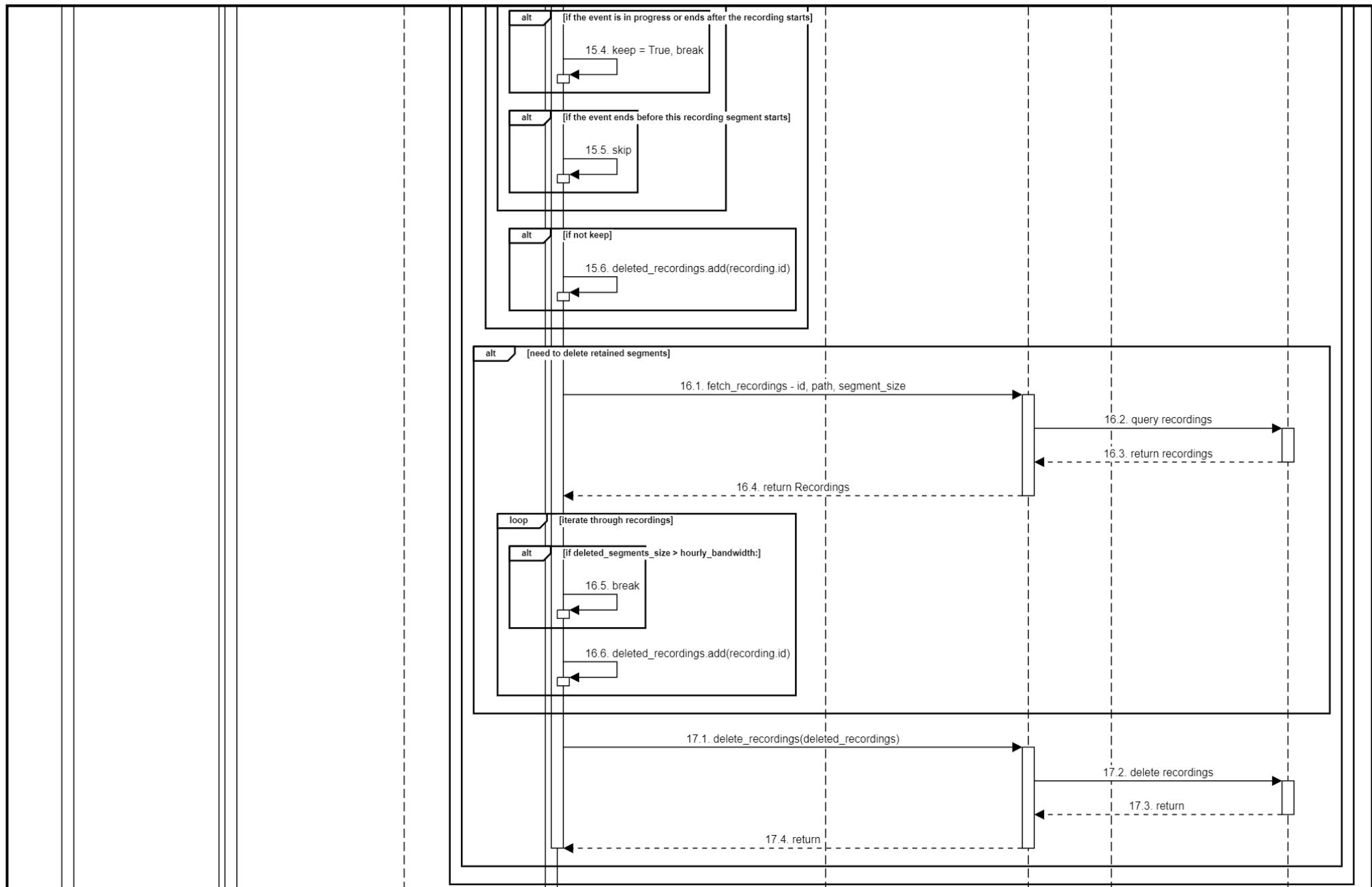


Figure 159. Class diagram - Storage Maintenance Processing

2.12.2 Sequence Diagram







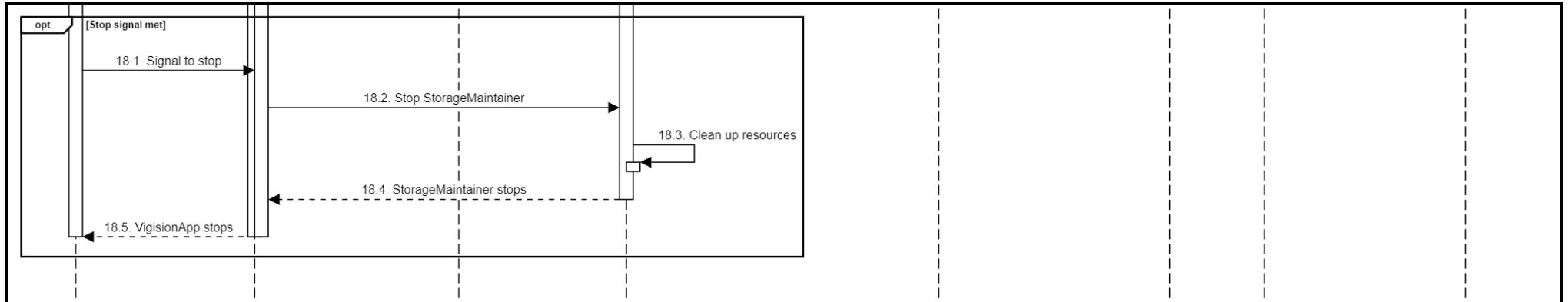


Figure 160. Class diagram - Storage Maintenance Processing

2.13 Stats Emission Processing

2.13.1 Class Diagram

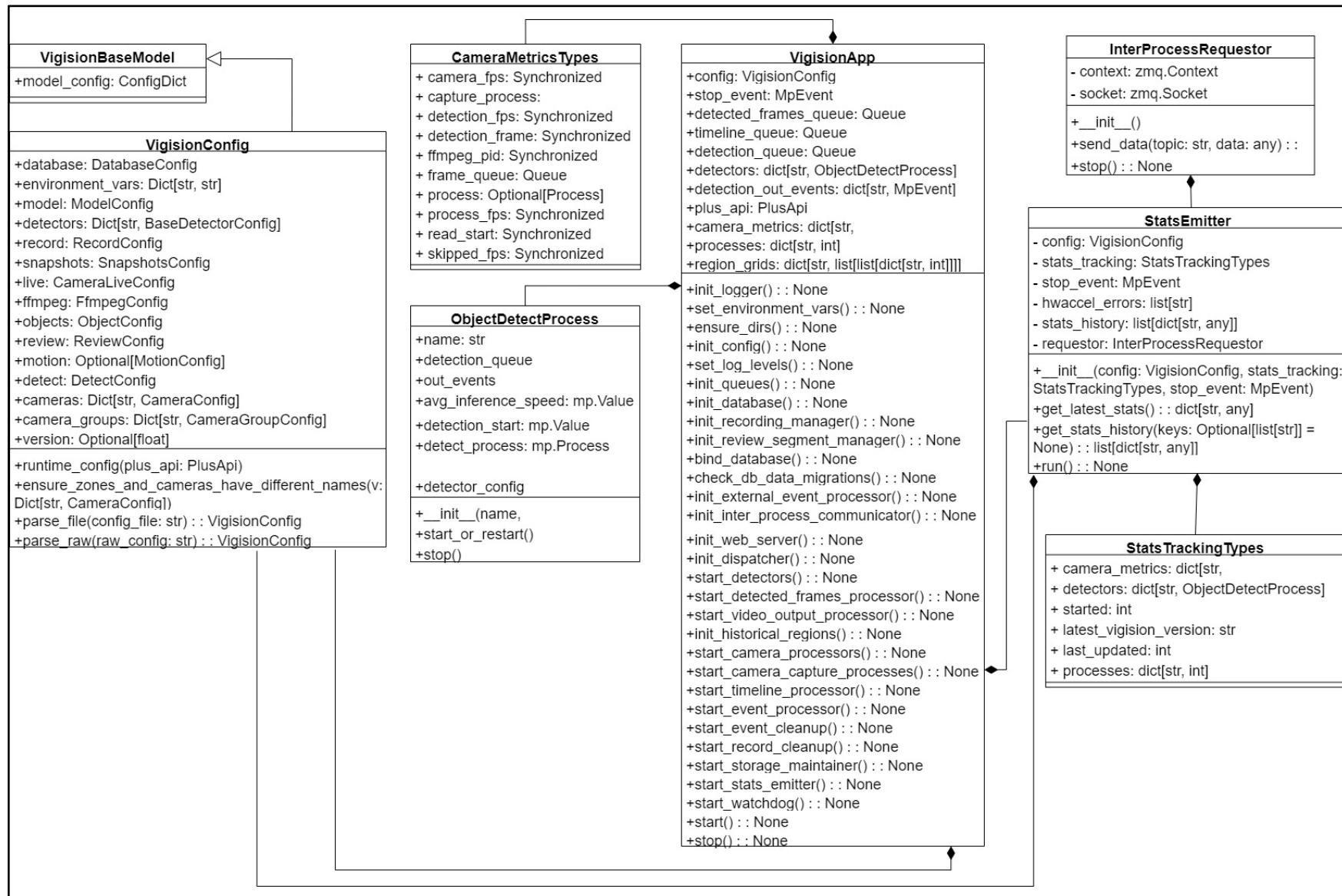
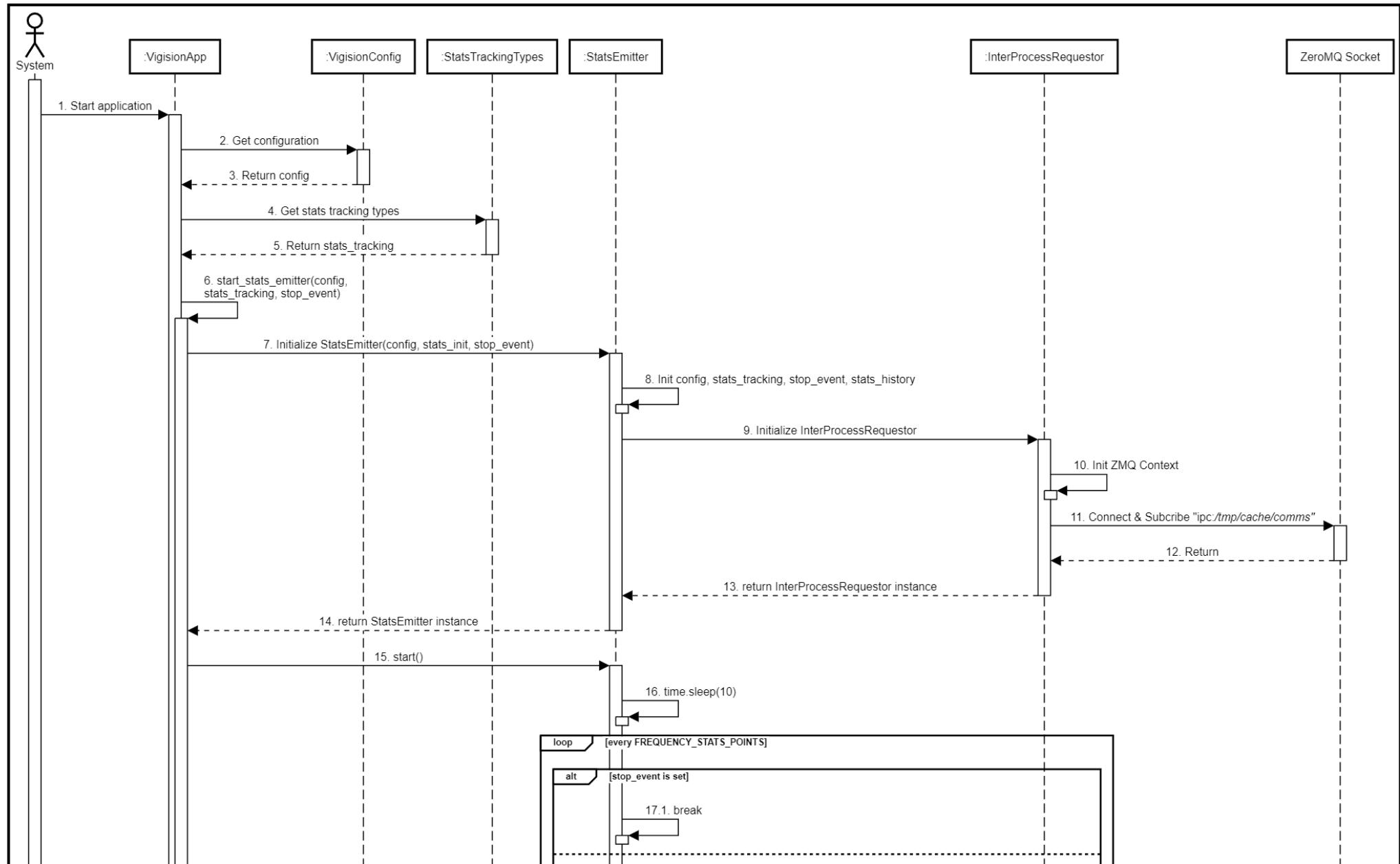


Figure 161. Class diagram - Stats Emission Processing

2.13.2 Sequence Diagram



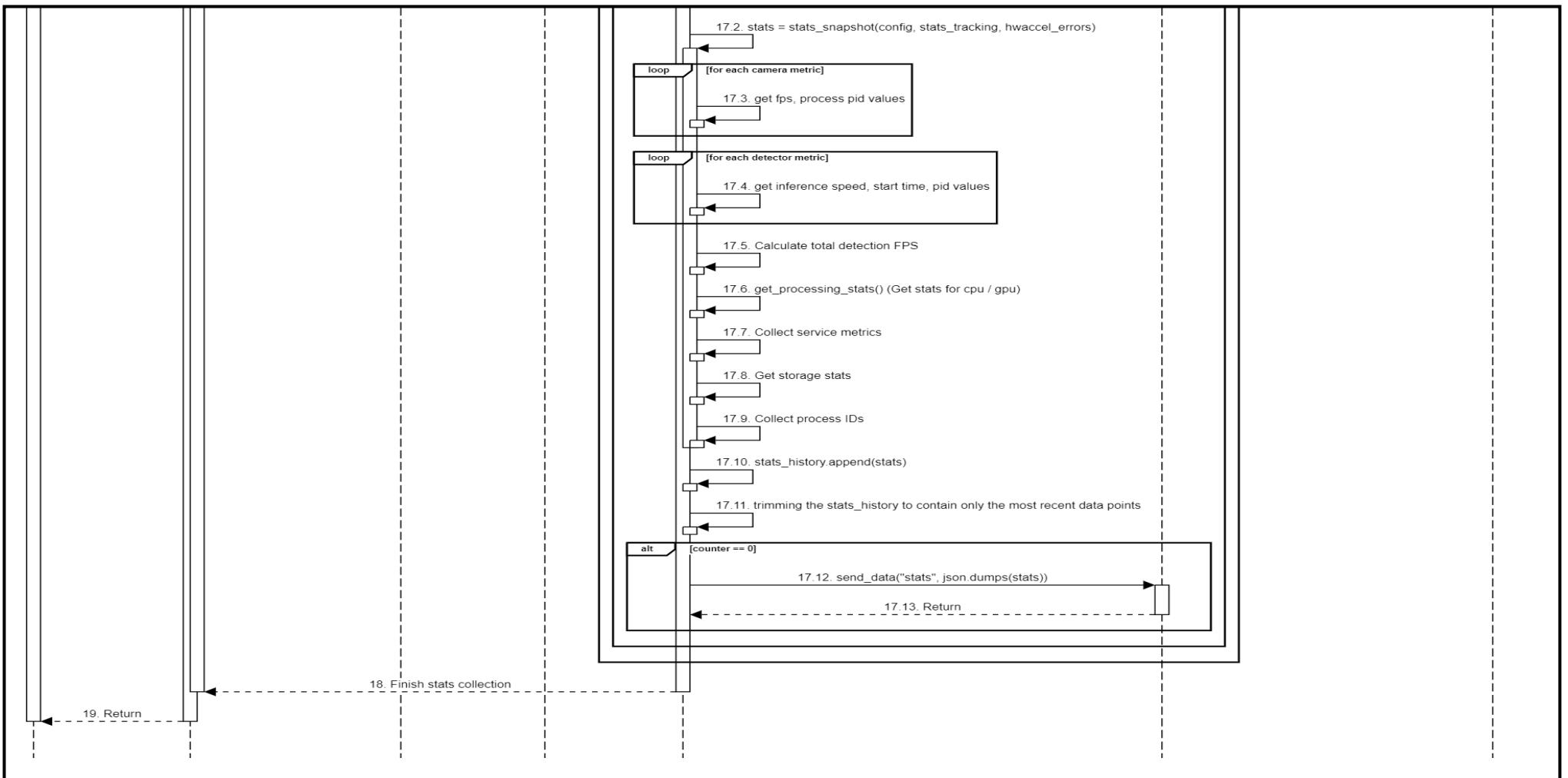


Figure 162. Sequence diagram - Stats Emission Processing

2.14 Start Vigision Watchdog

2.14.1 Class Diagram

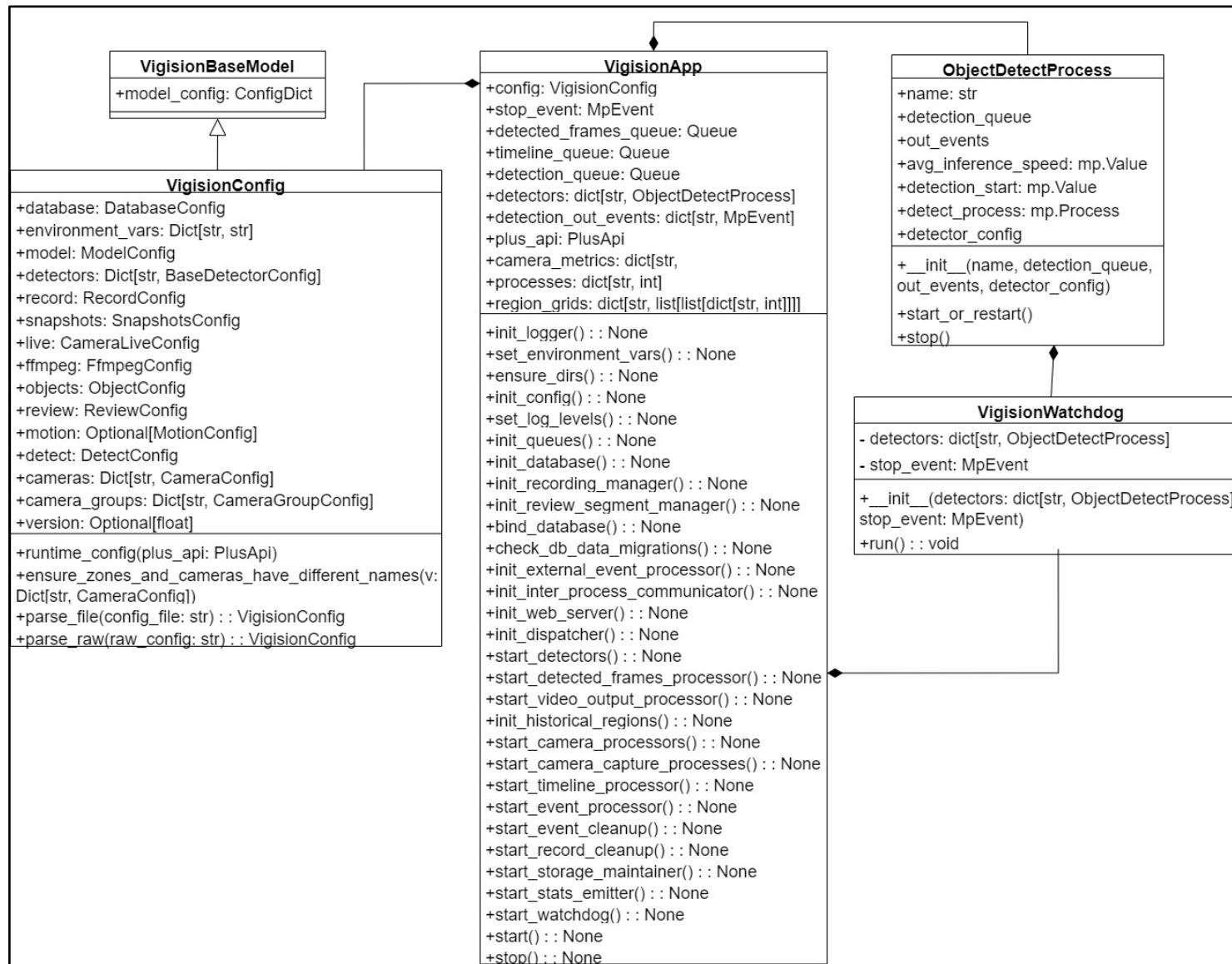


Figure 163. Class diagram - Start Vigision Watchdog

2.14.2 Sequence Diagram

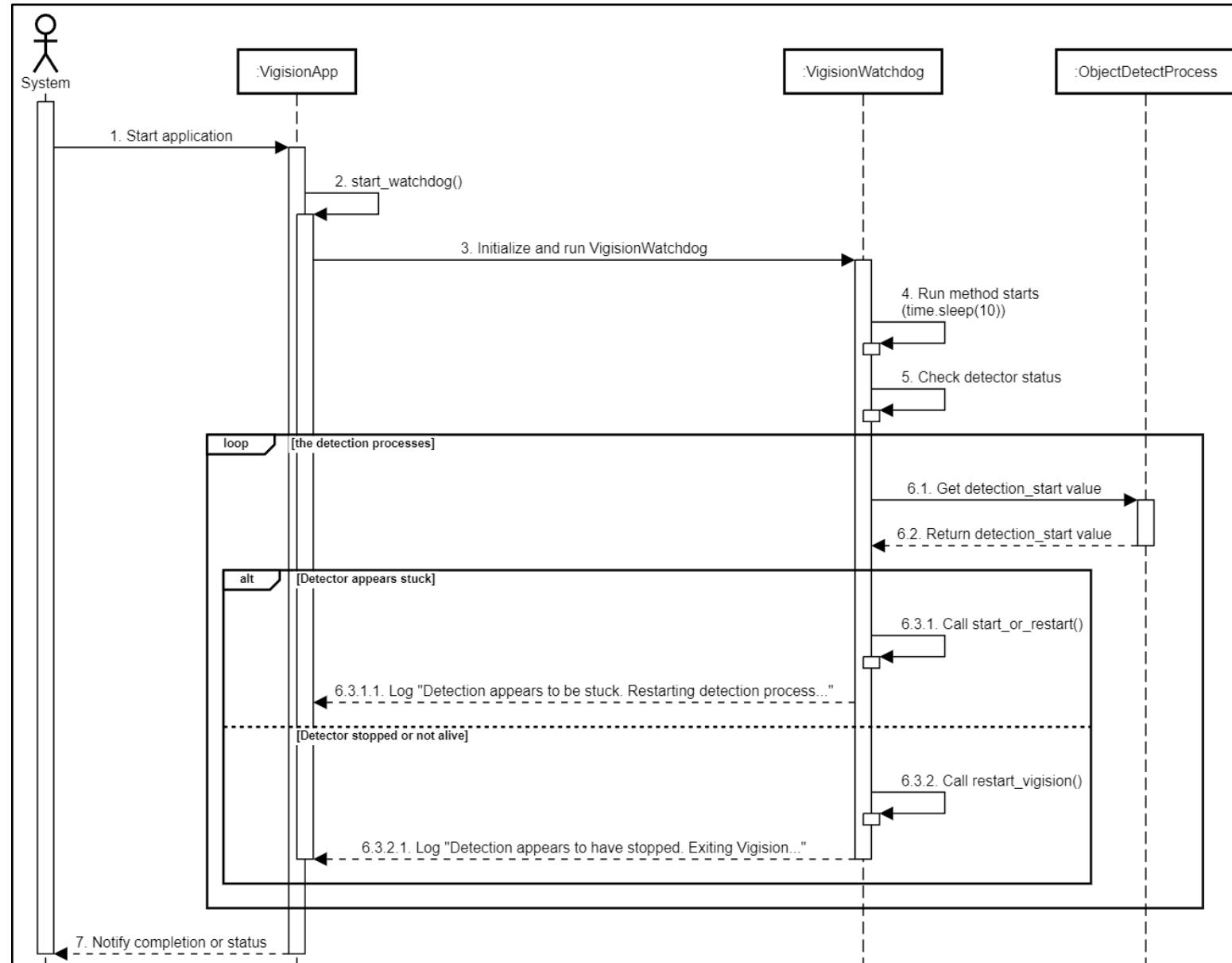


Figure 164. Sequence diagram - Start Vigision Watchdog

3 Detailed Design

3.1 Login

3.1.1 Class Diagram

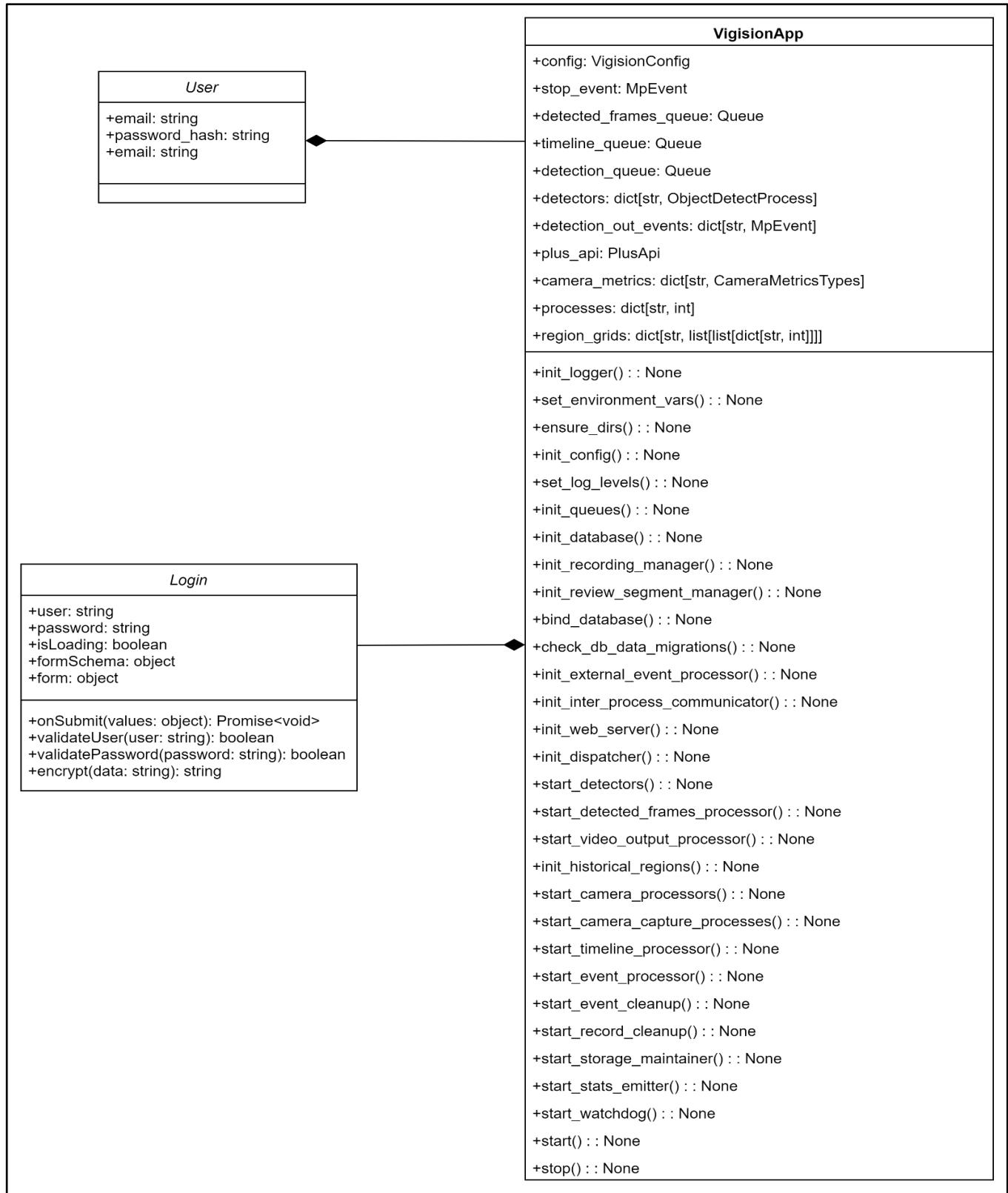


Figure 165. Class diagram - Login

3.1.2 Sequence Diagram

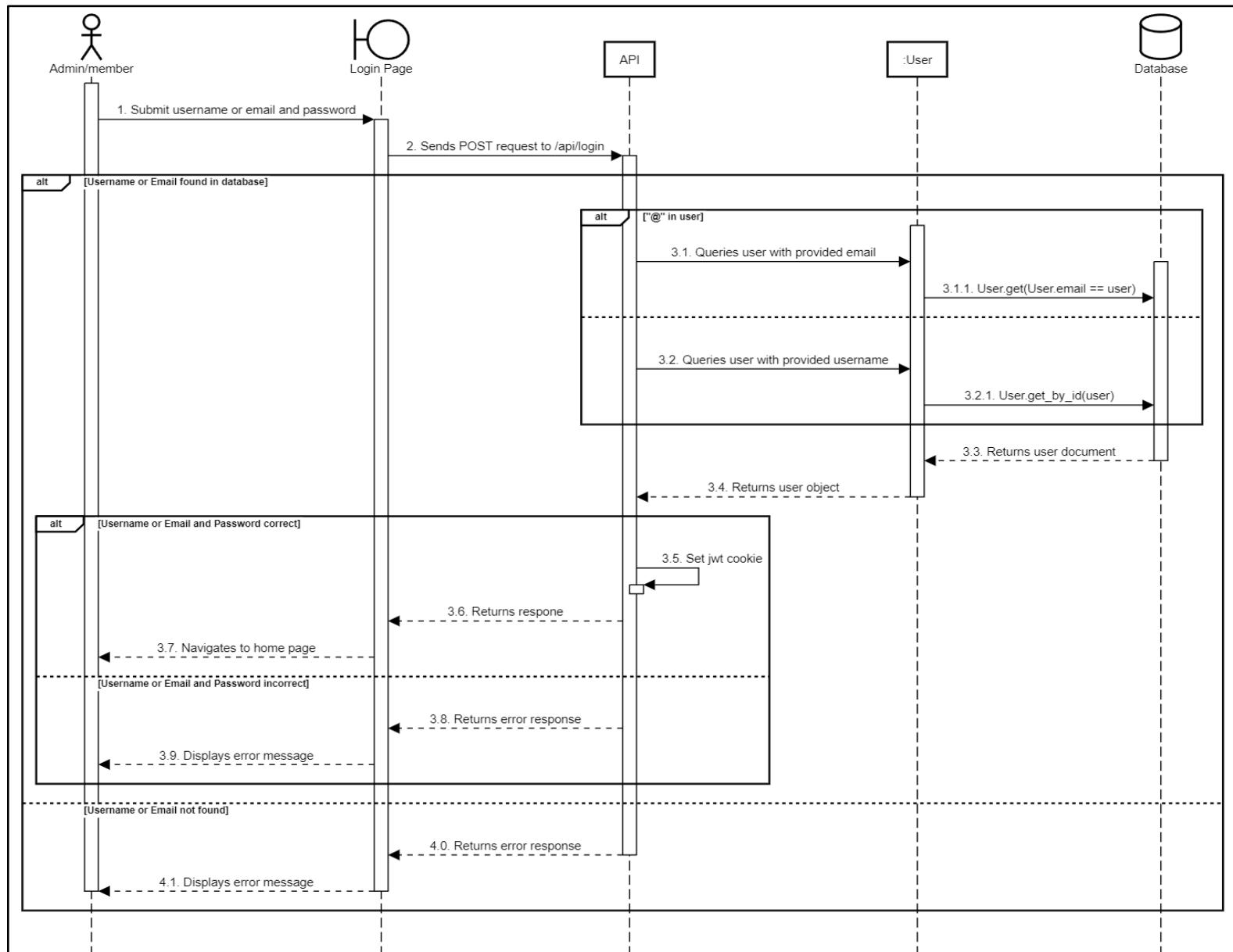


Figure 166. Sequence diagram - Login

3.2 Logout

3.2.1 Class Diagram

Logout is performed by removing the user's cookie. This class has been defined in above section that store on the cookie so it is not shown in the class diagram.

3.2.2 Sequence Diagram

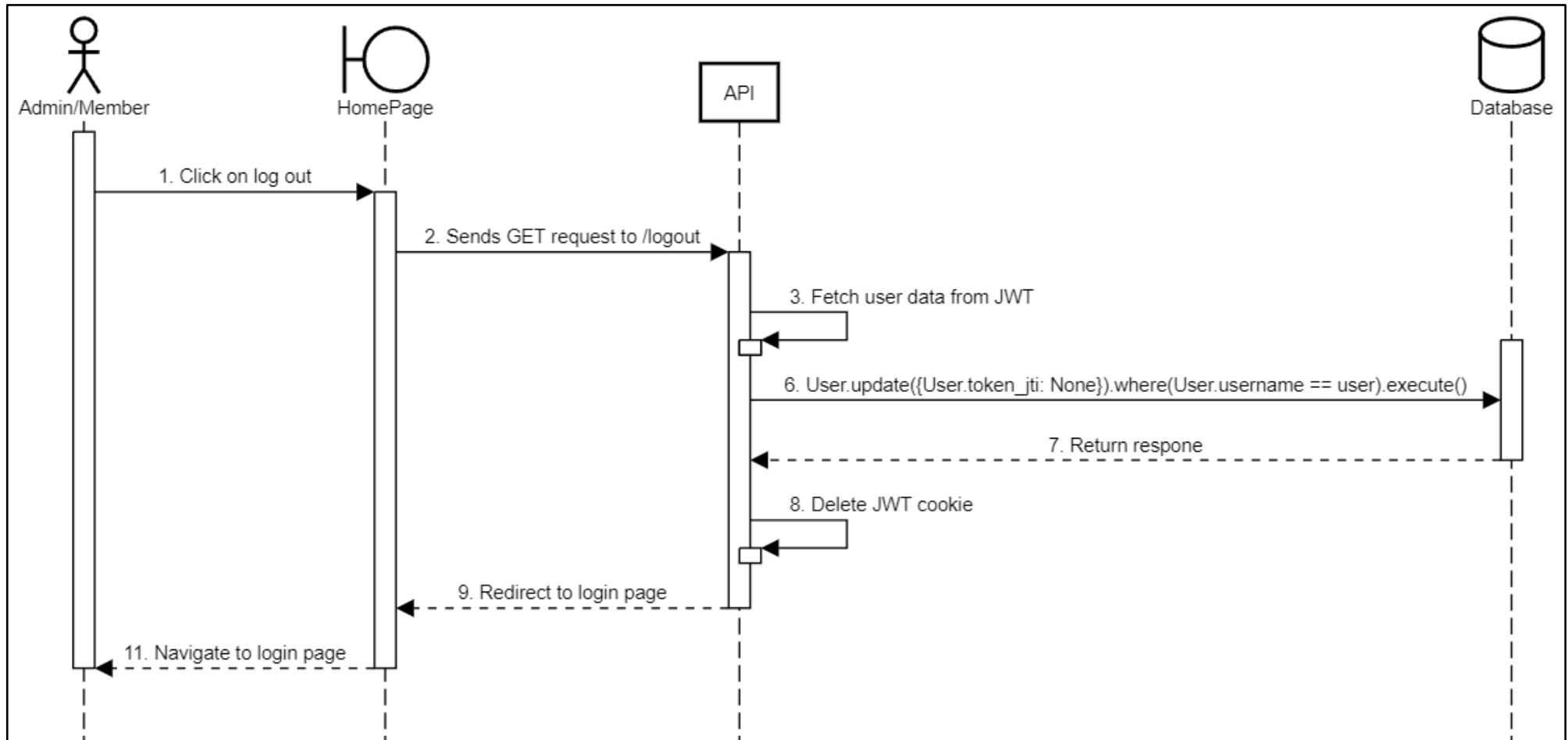


Figure 167. Sequence diagram - Logout

3.3 Forgot password

3.3.1 Class Diagram

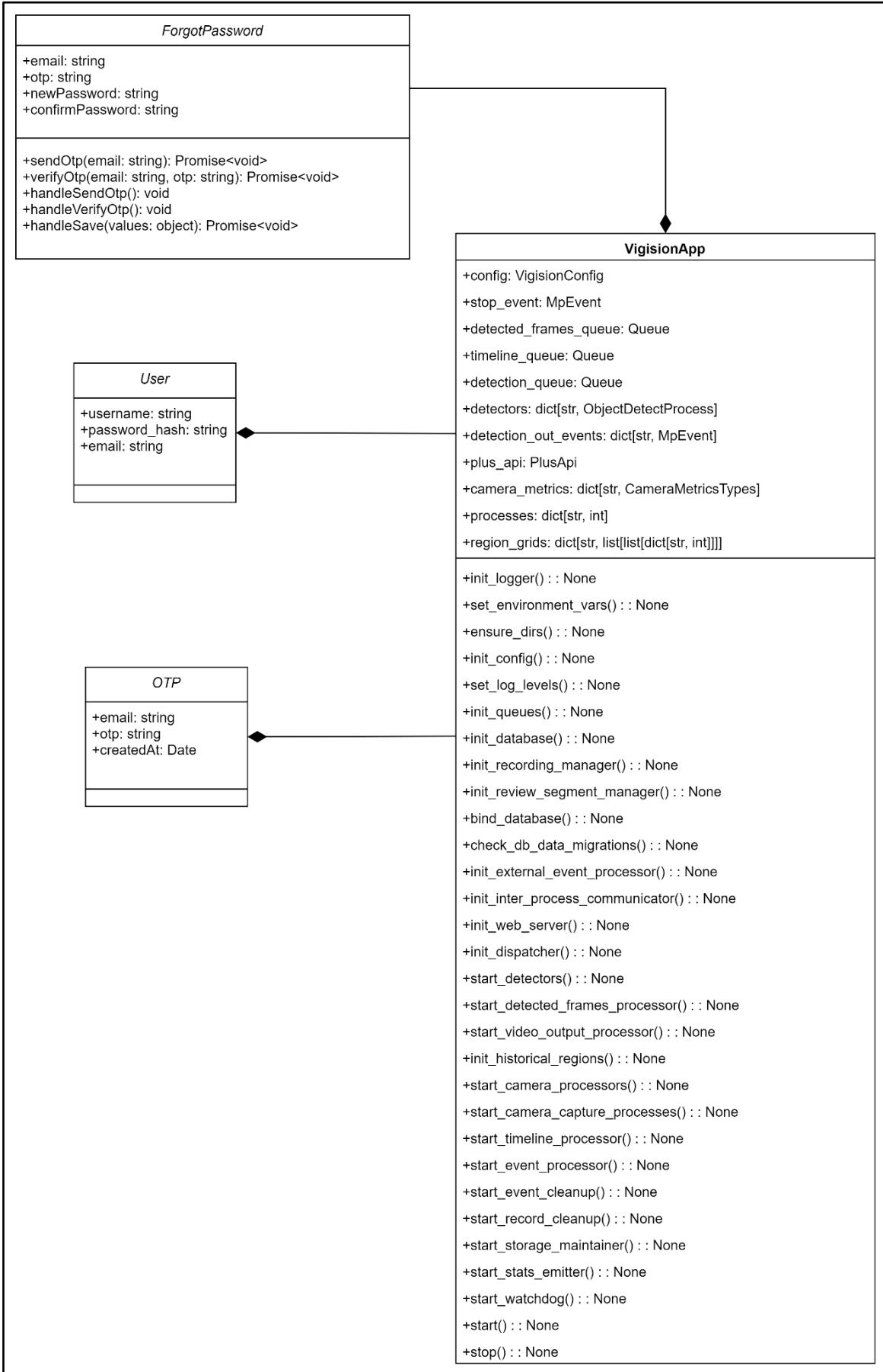
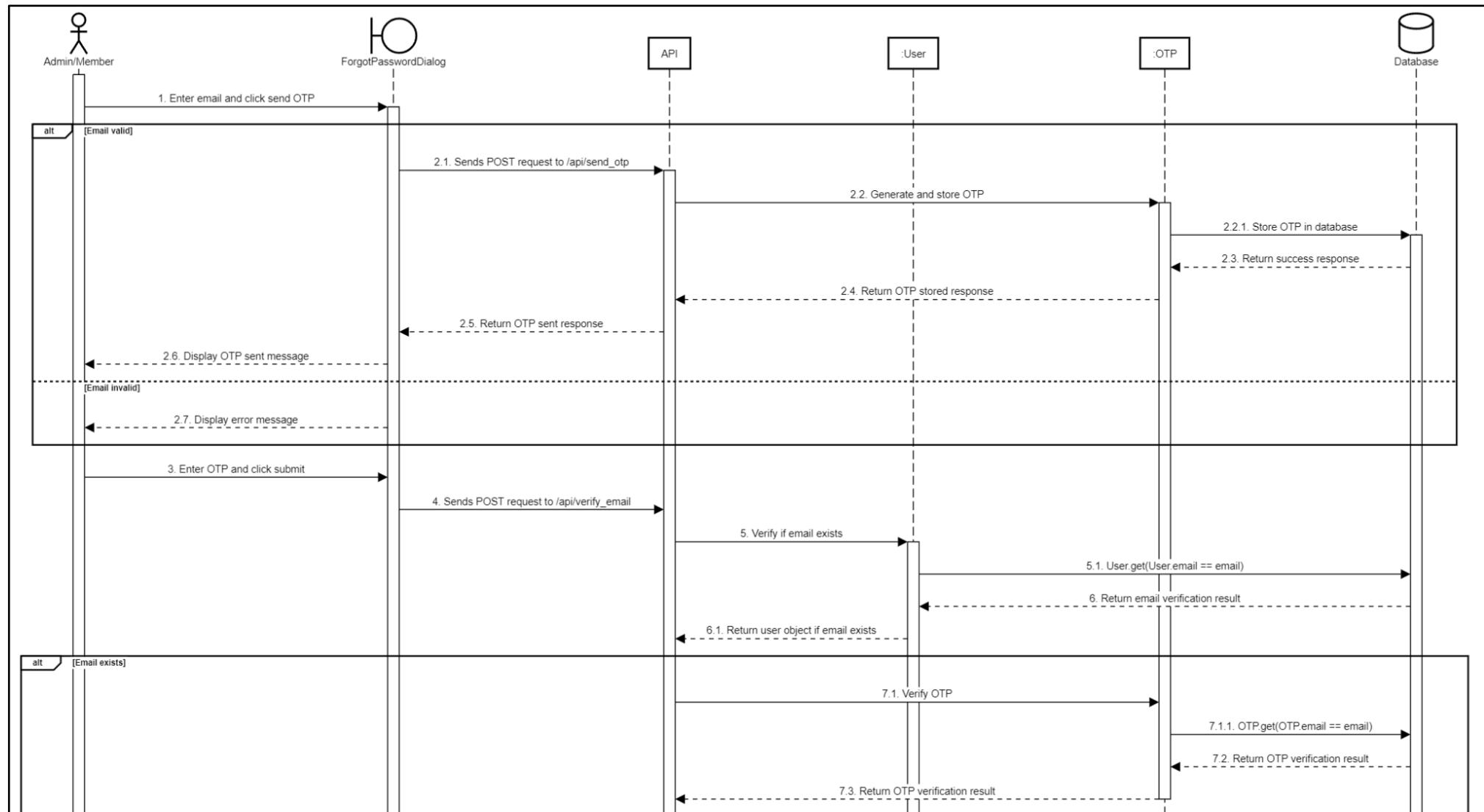


Figure 168. Class Diagram - Forgot password

3.3.2 Sequence Diagram



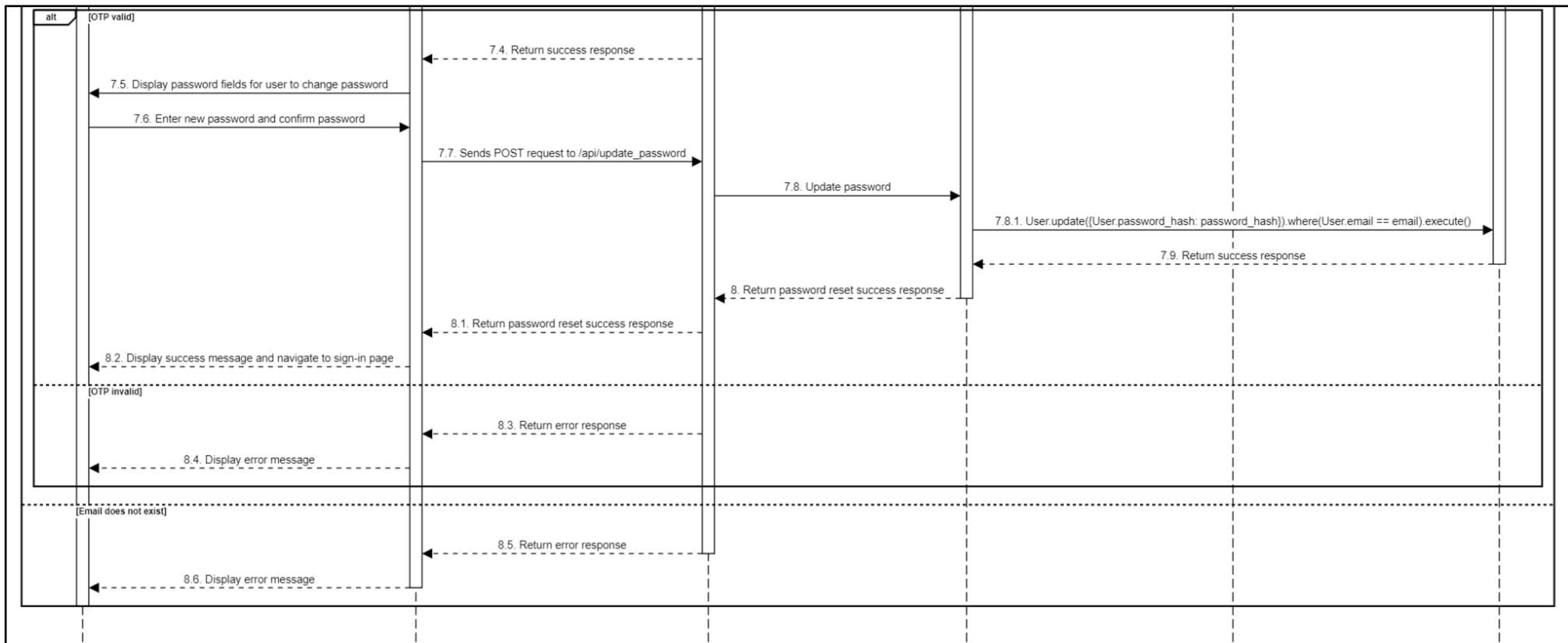


Figure 169. Sequence Diagram - Forgot password

3.4 Manage users

3.4.1 Class Diagram

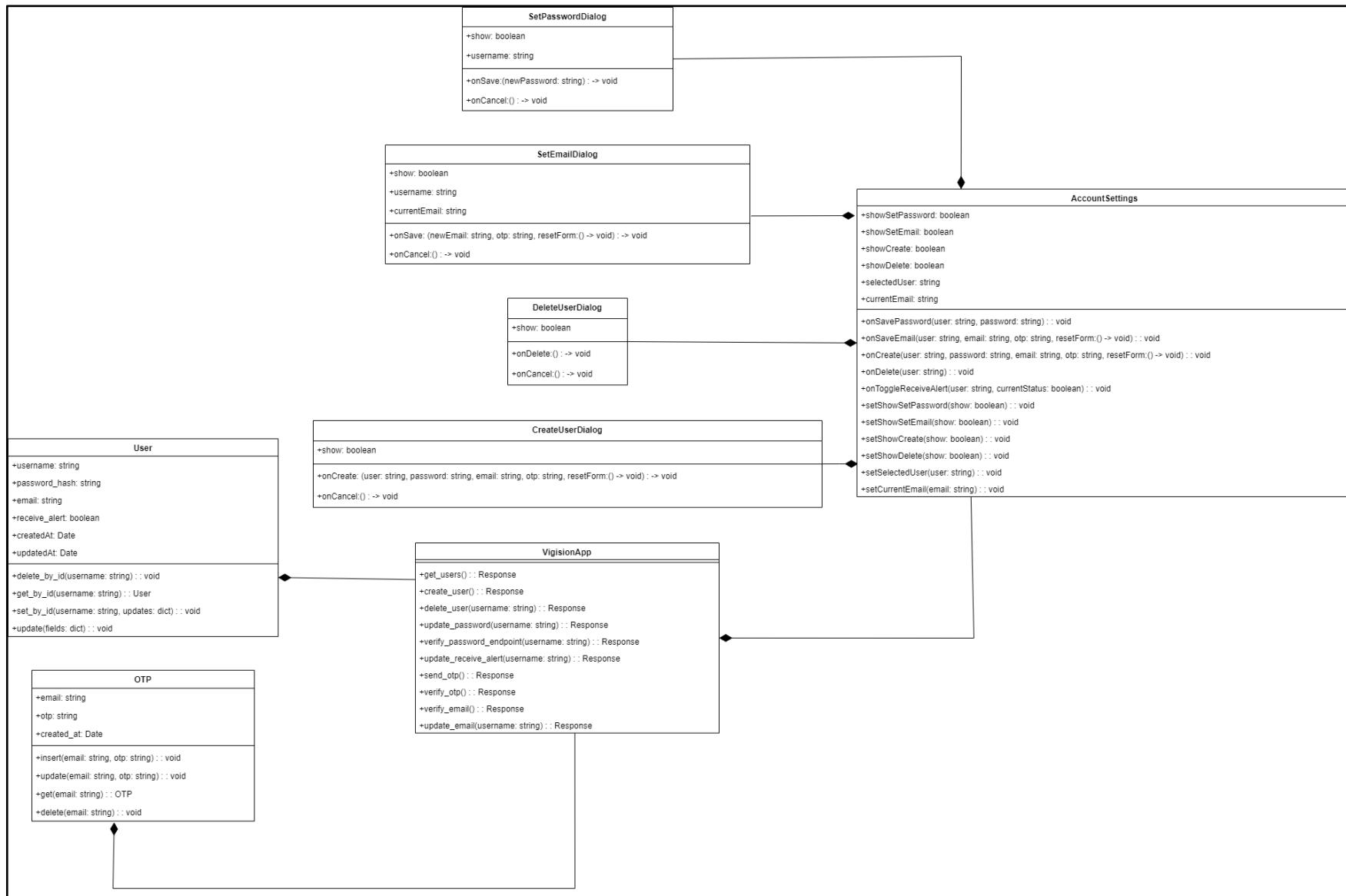


Figure 170. Class Diagram - Manage users

3.4.2 Sequence Diagram

- View a list of user

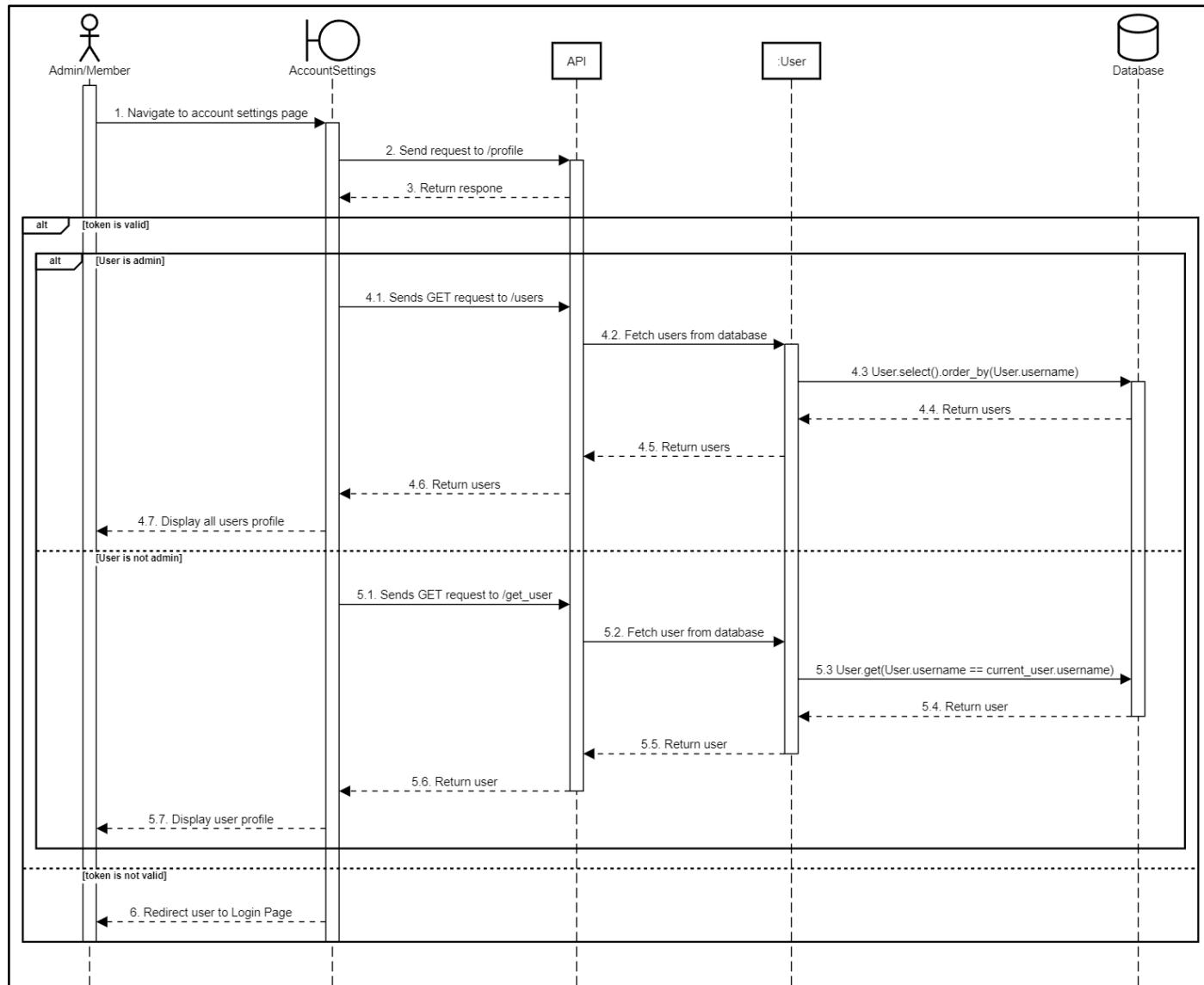
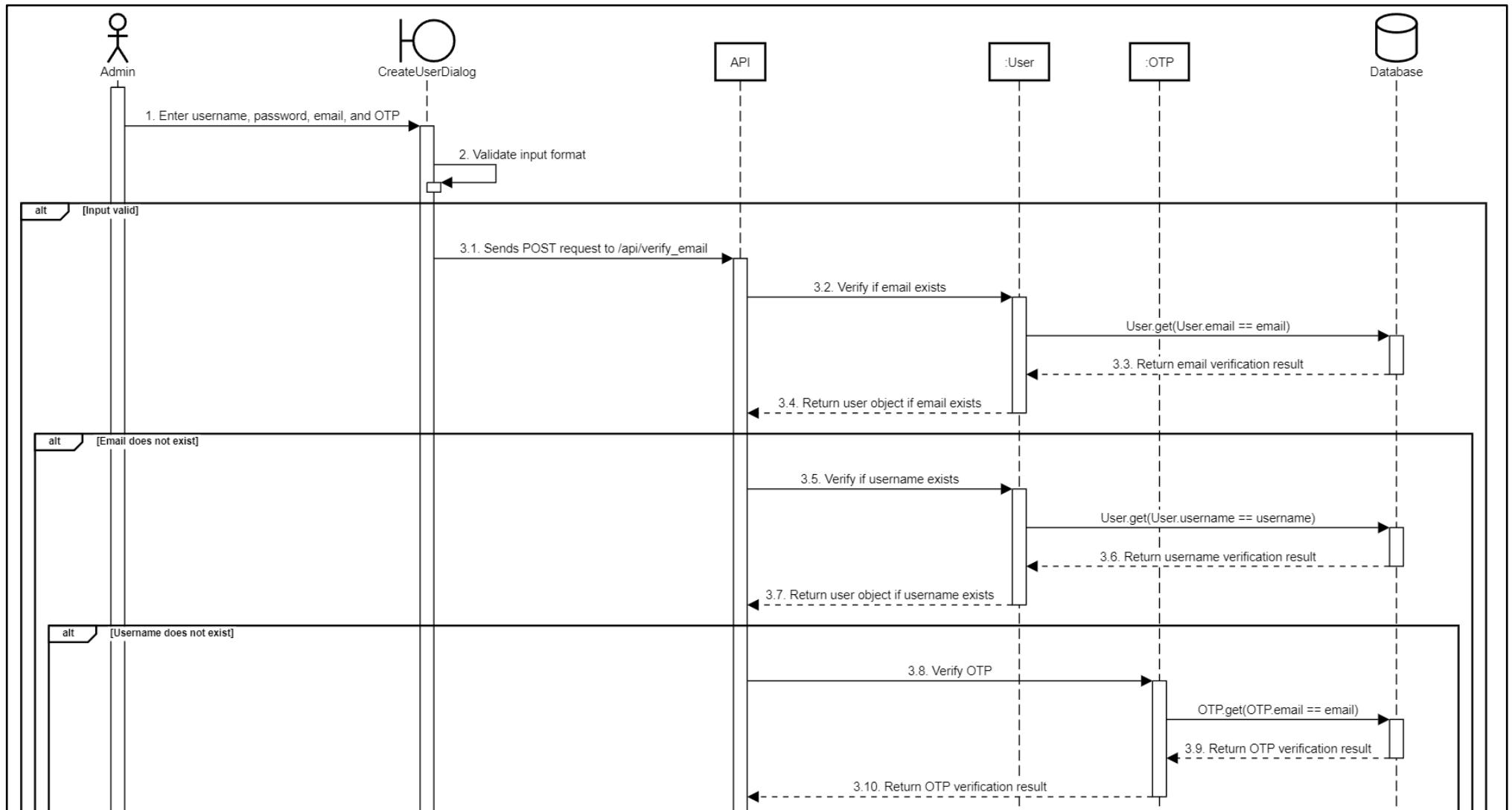


Figure 171. Sequence Diagram - View a list of user

- Create user



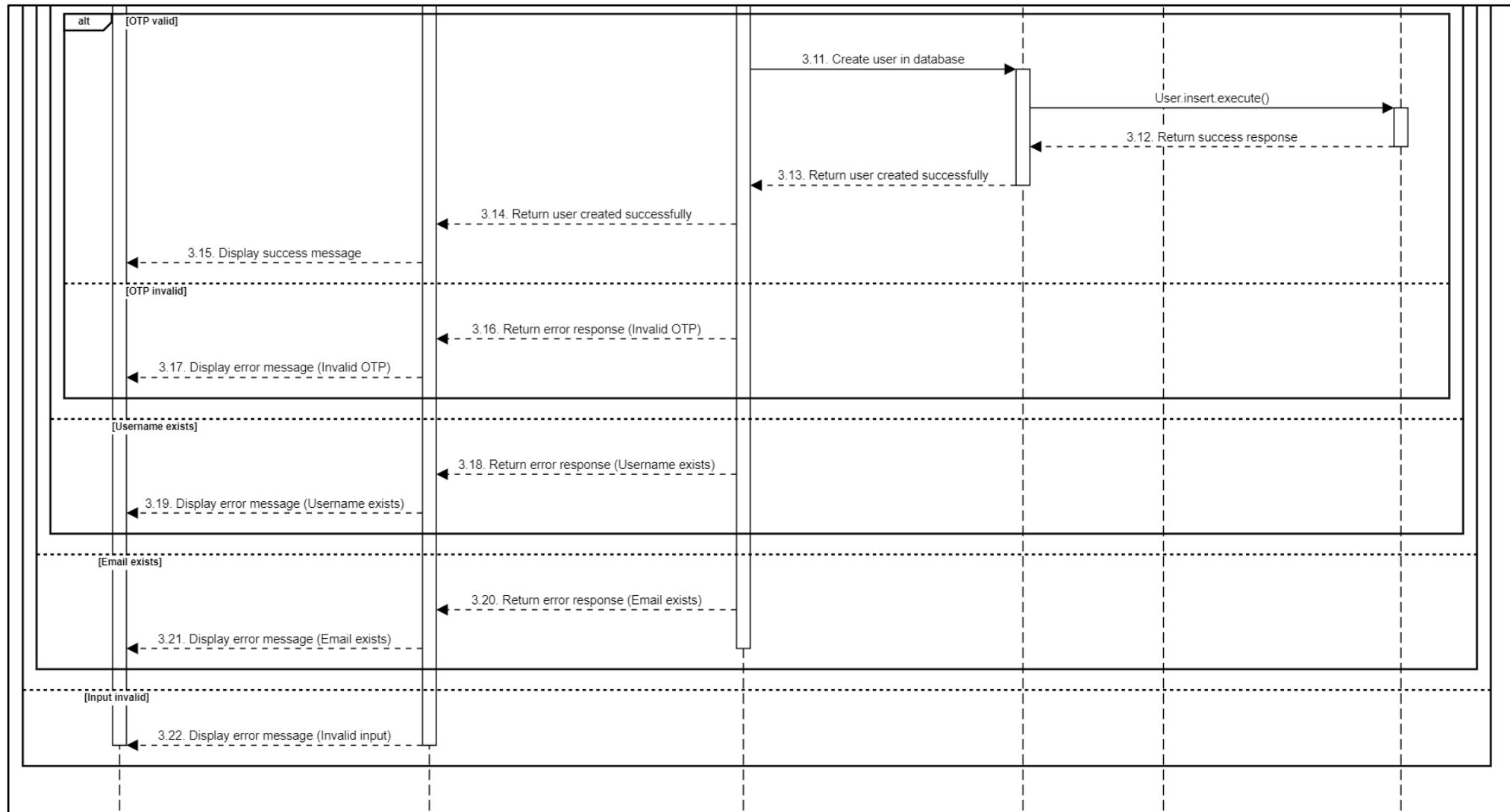


Figure 172. Sequence Diagram - Create user

- Update password

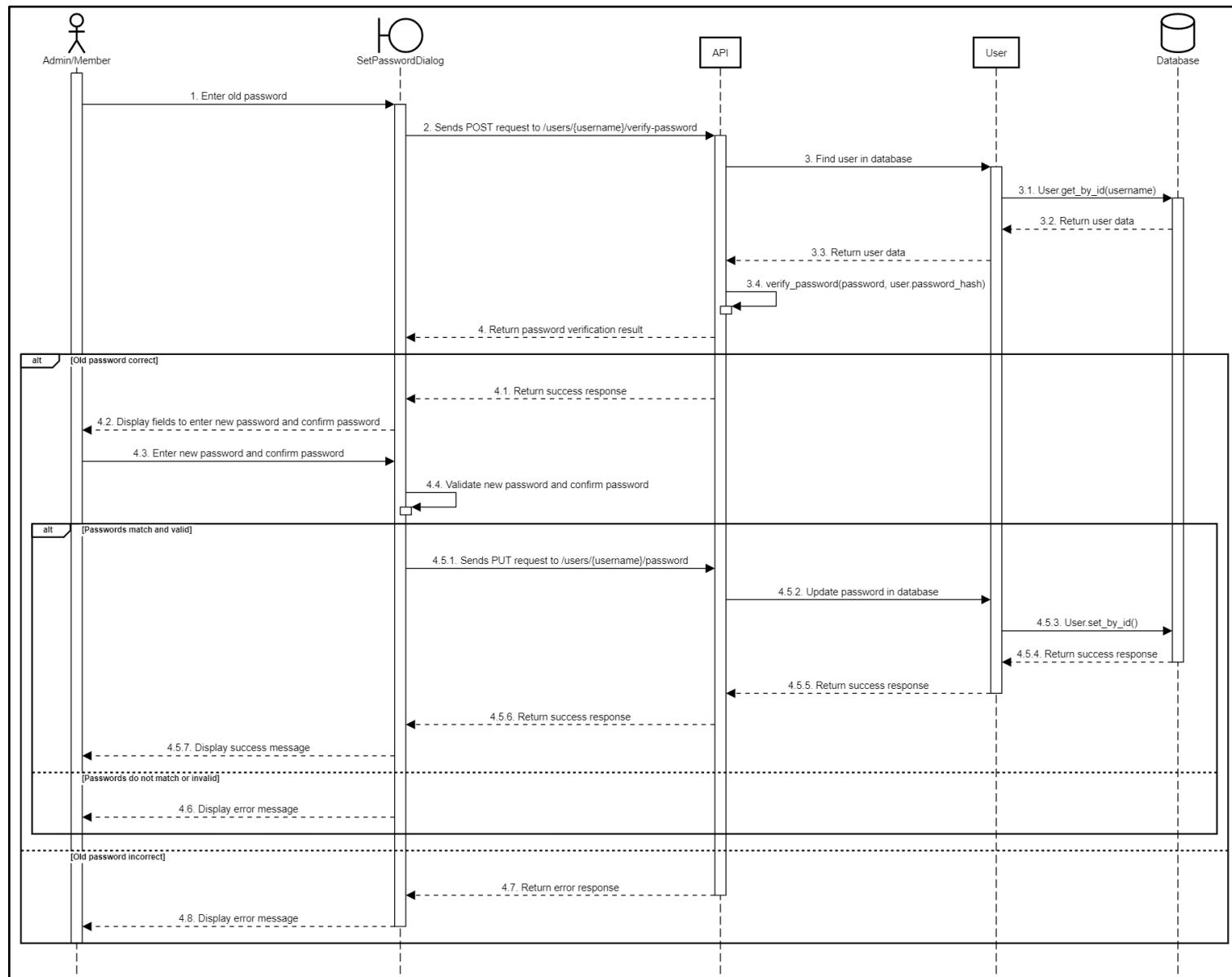
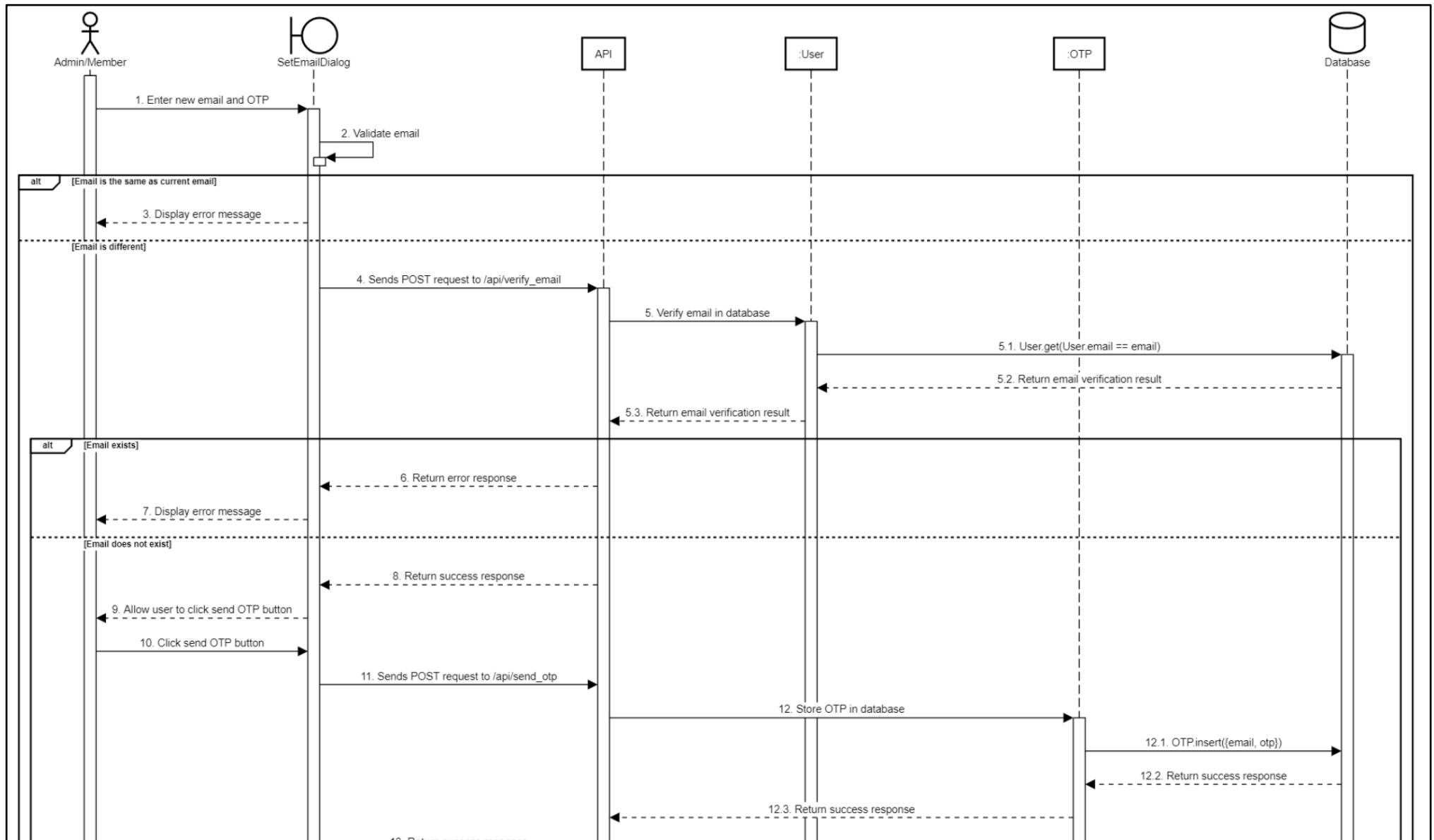


Figure 173. Sequence Diagram - Update password

- Update email:



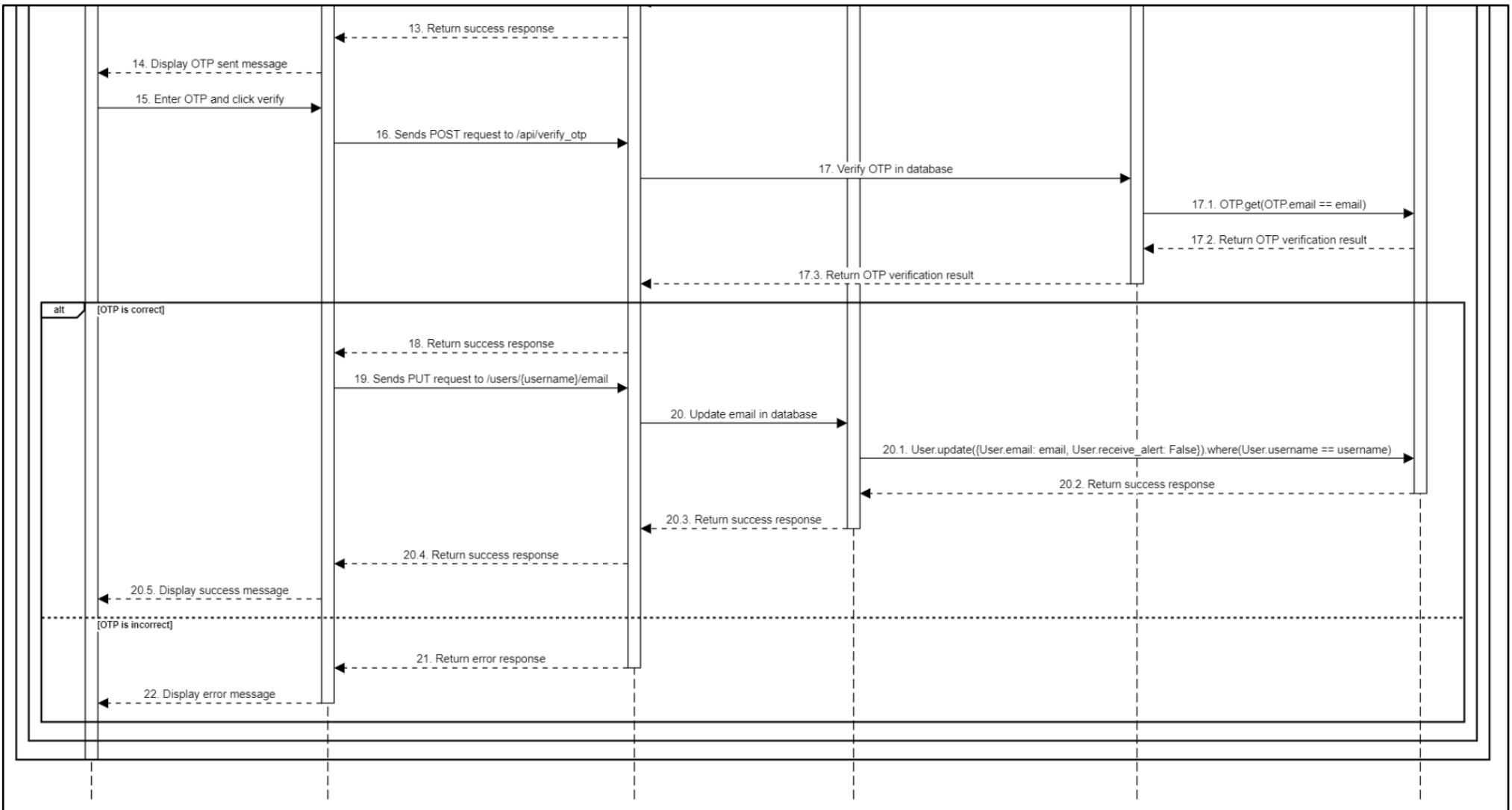


Figure 174. Sequence Diagram - Update email

- Delete user:

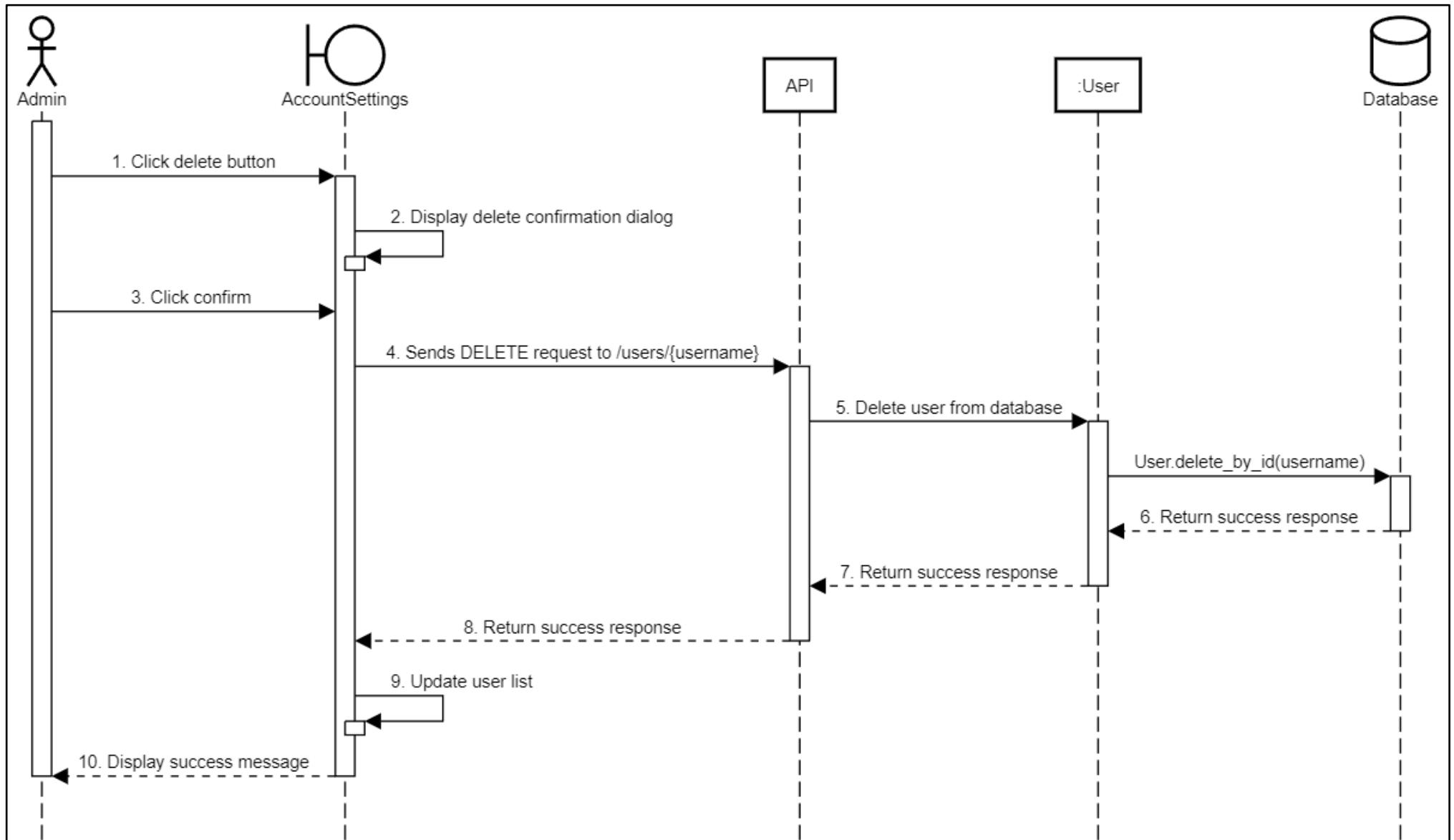


Figure 175. Sequence Diagram - Delete user

- Enable/disable receive alert by email

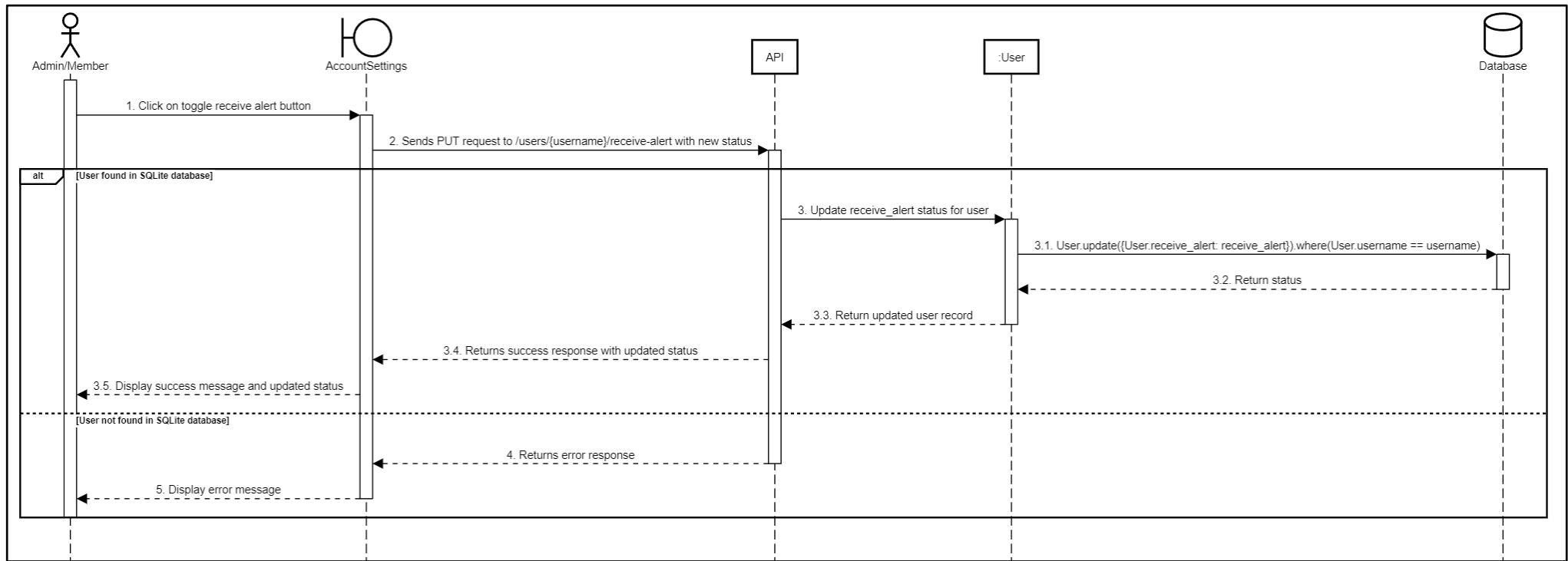


Figure 176. Sequence Diagram - Enable/disable receive alert by email

3.5 View live cameras

3.5.1 Class Diagram

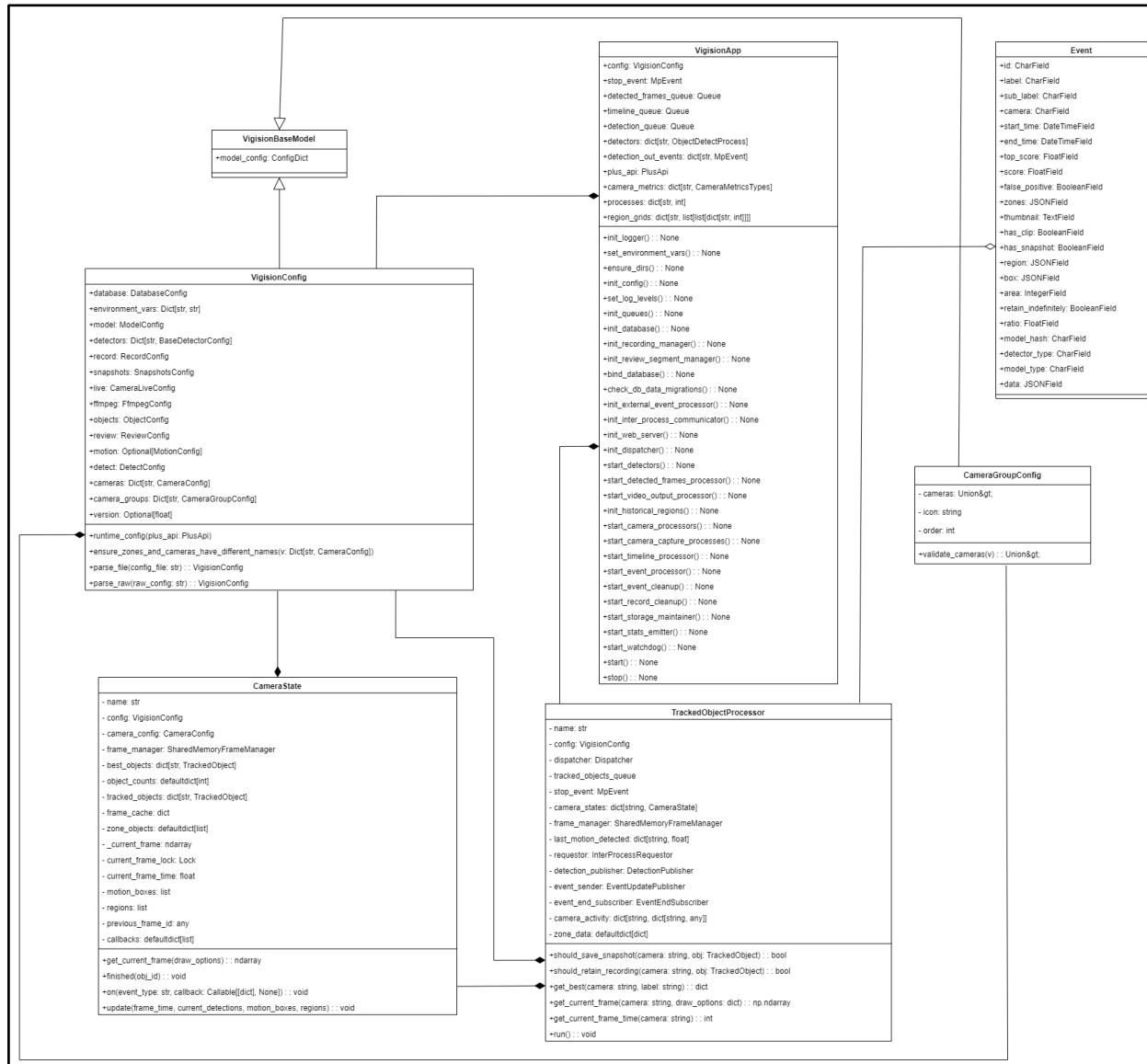
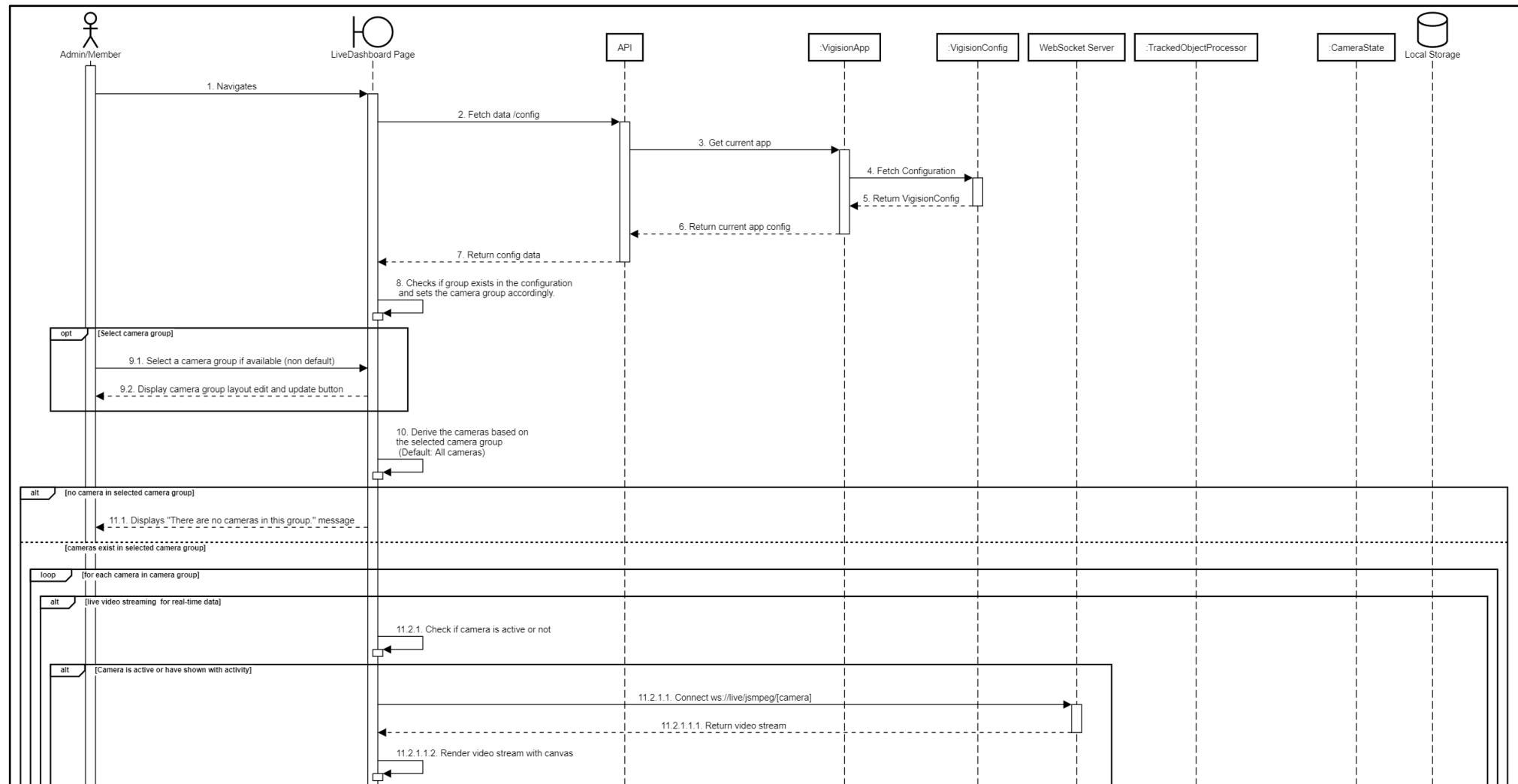


Figure 177. Class Diagram - View live cameras

3.5.2 Sequence Diagram



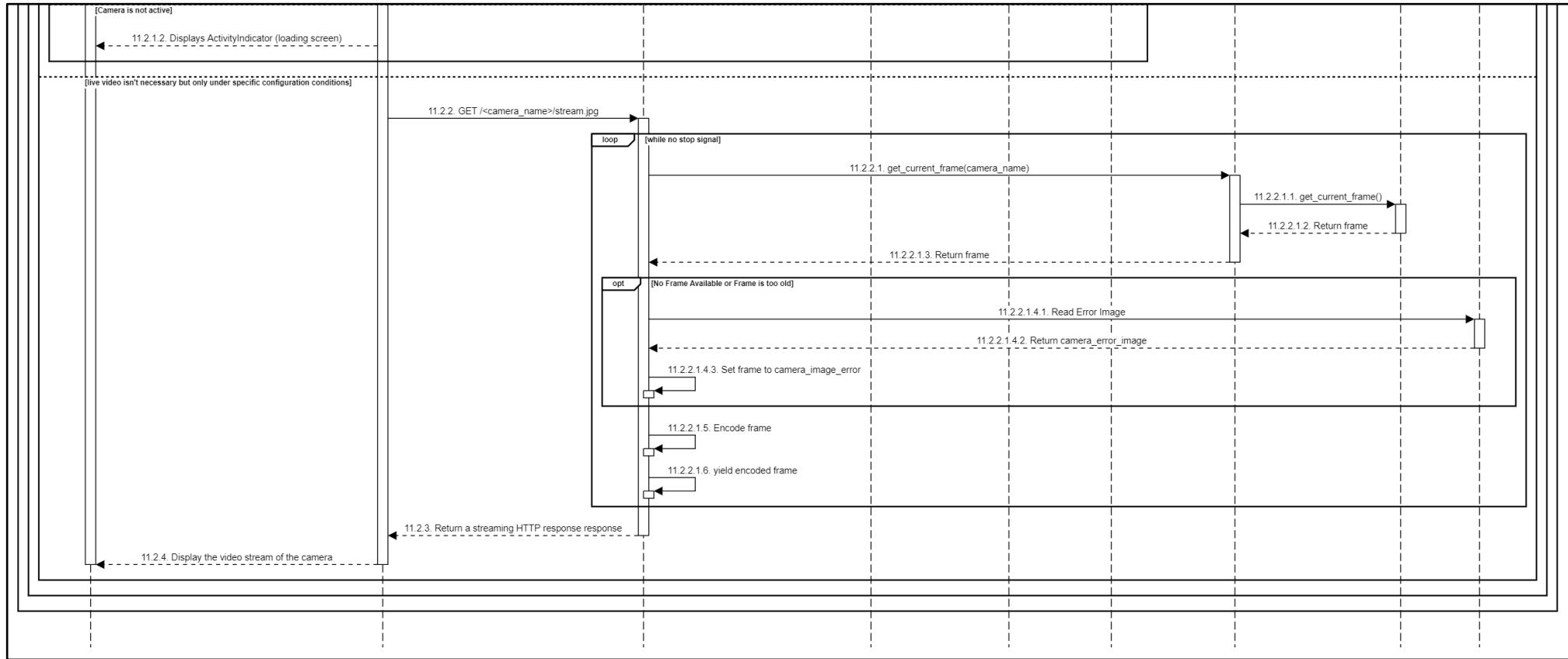


Figure 178. Sequence Diagram - View live cameras

3.6 Manage cameras

3.6.1 Class Diagram

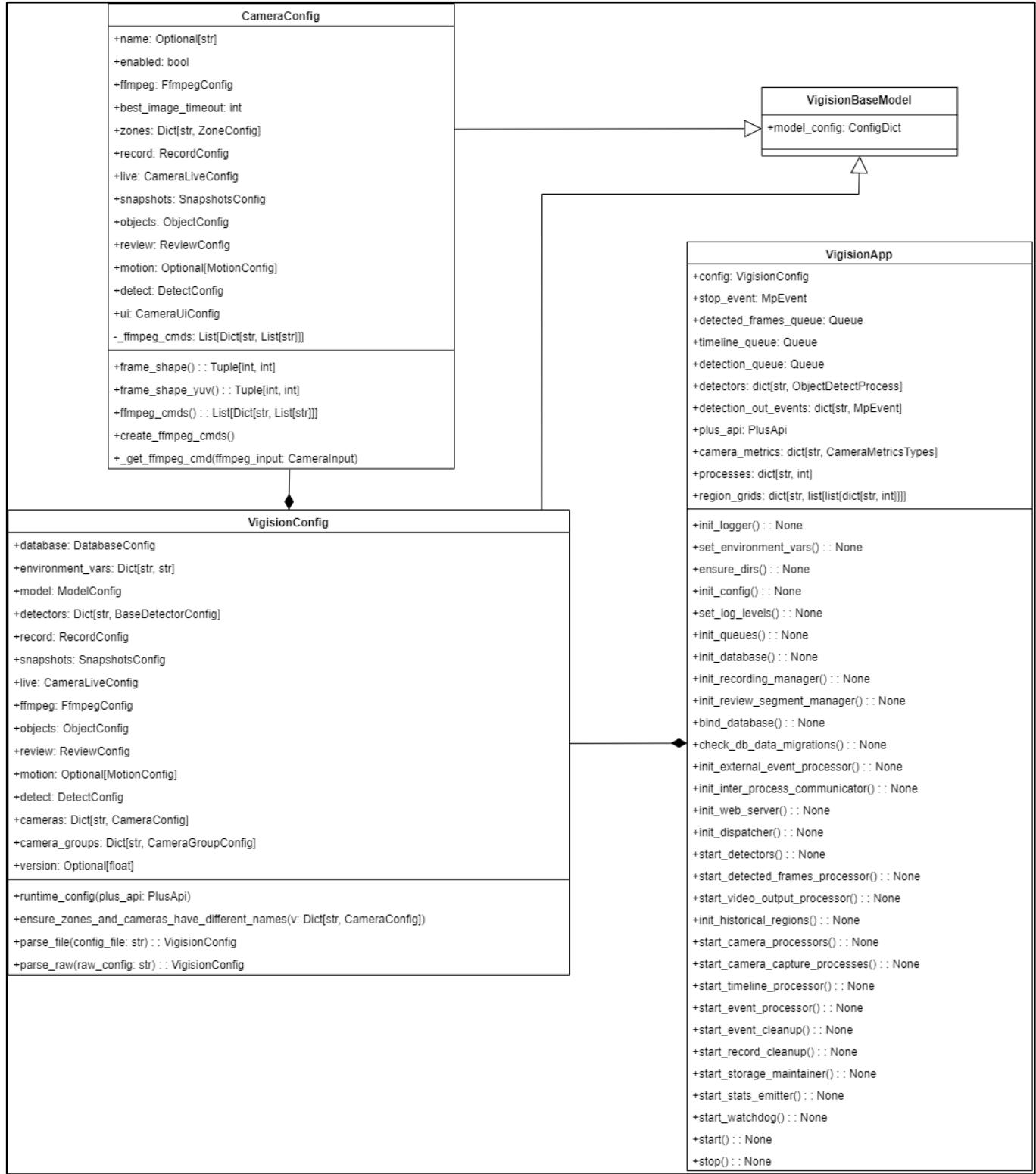


Figure 179. Class Diagram - Manage cameras

3.6.2 Sequence Diagram

a. 3.6.2.1 View list of cameras

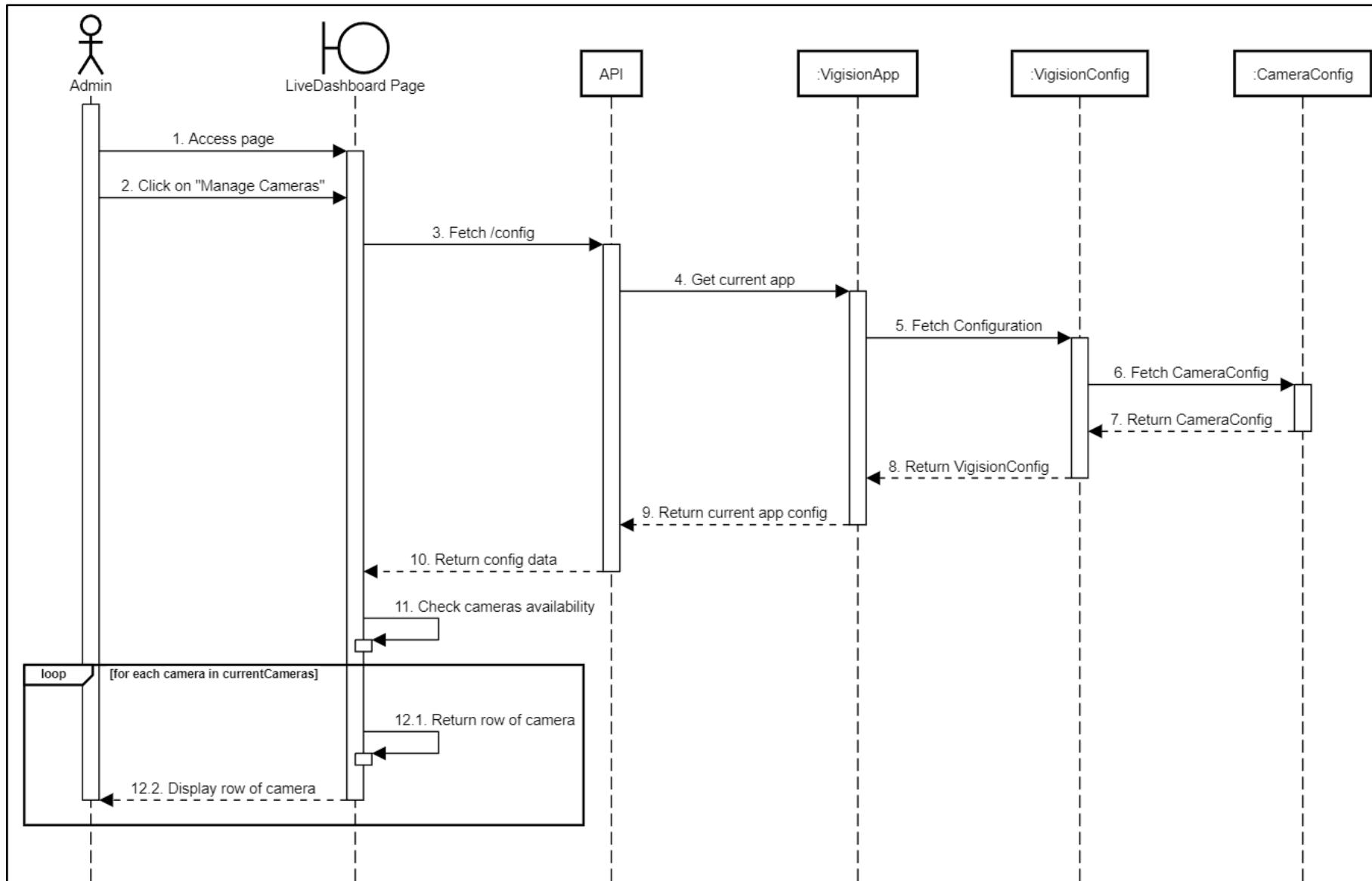
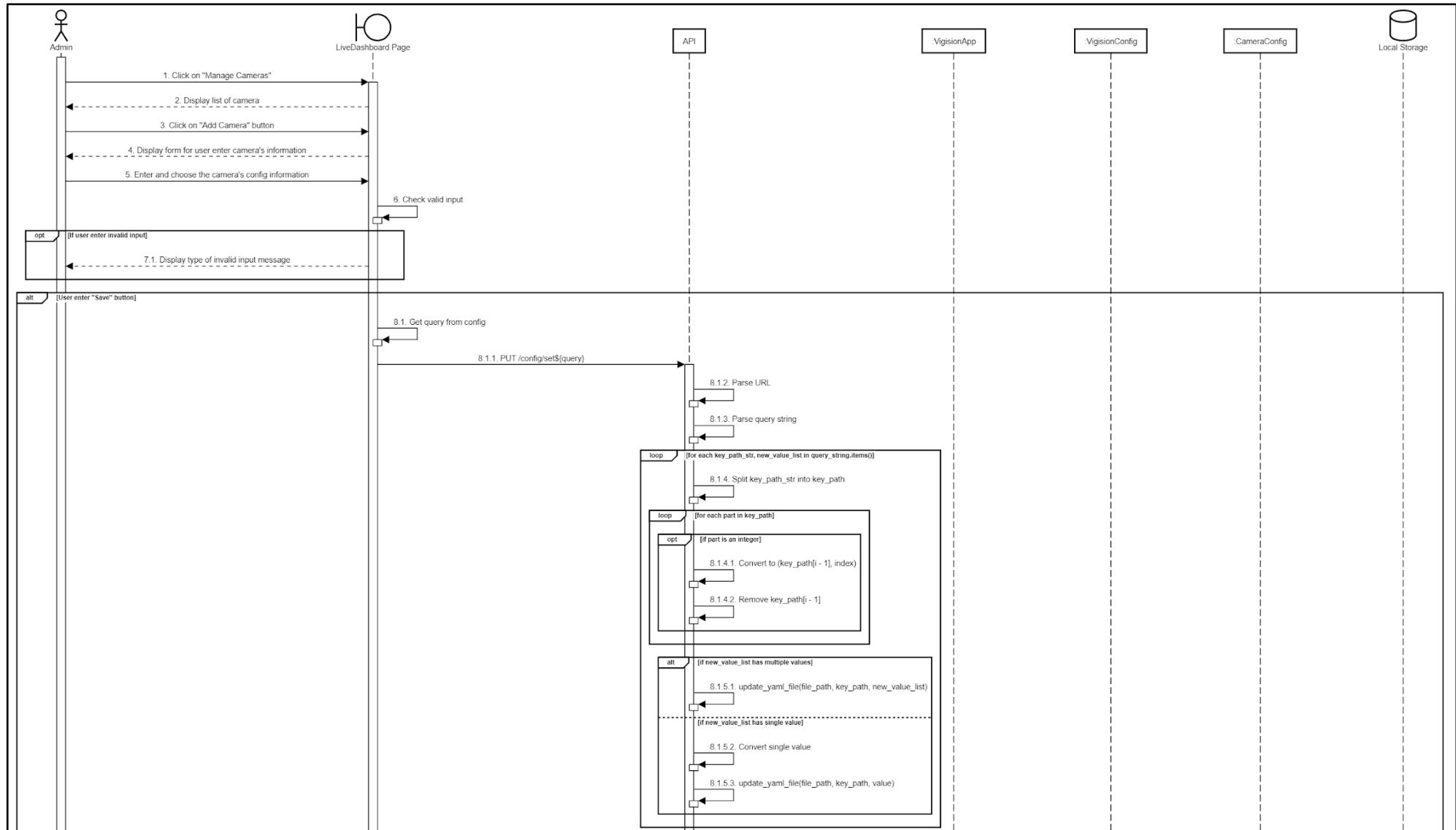


Figure 180. Sequence Diagram - View list of cameras

b. 3.6.2.2 Add camera



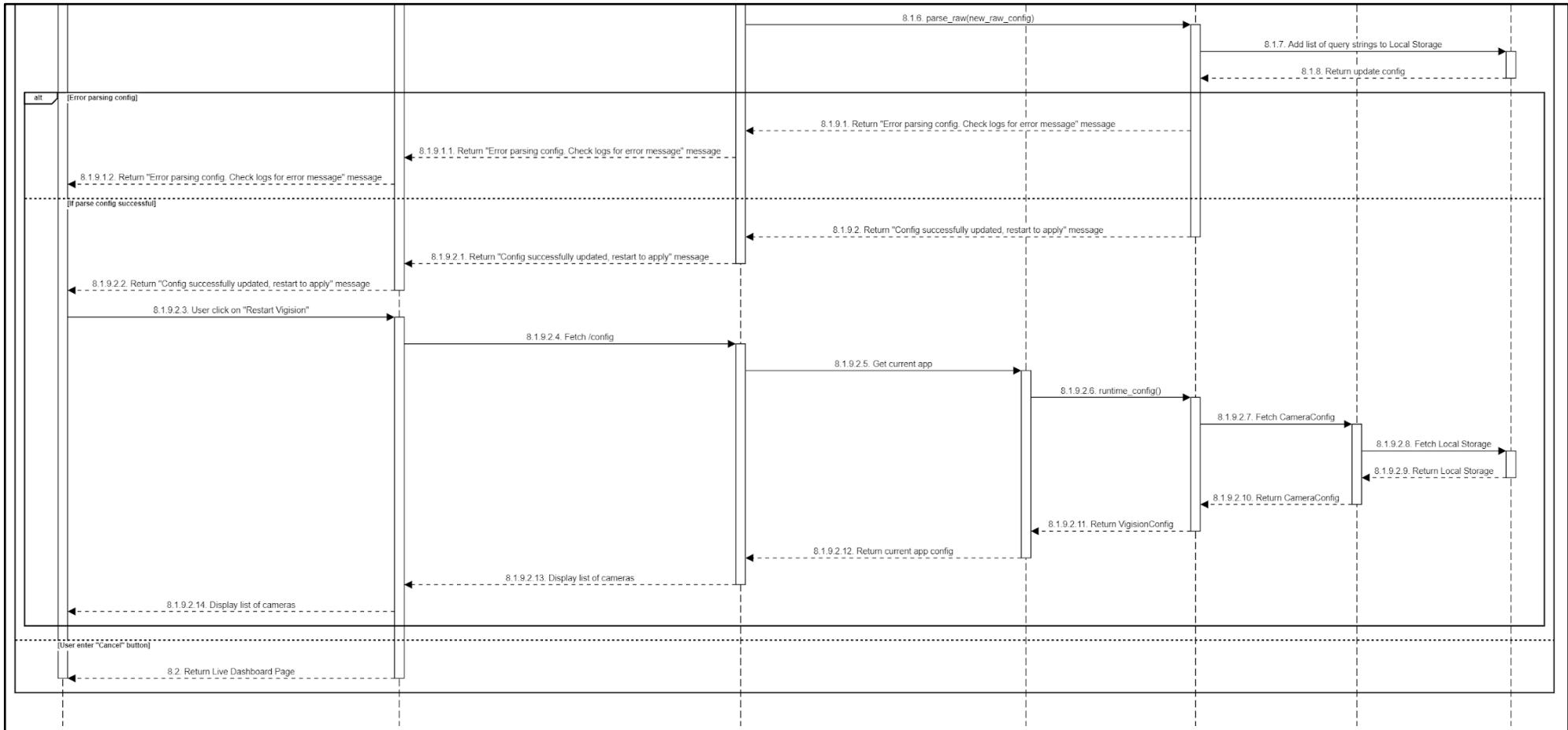
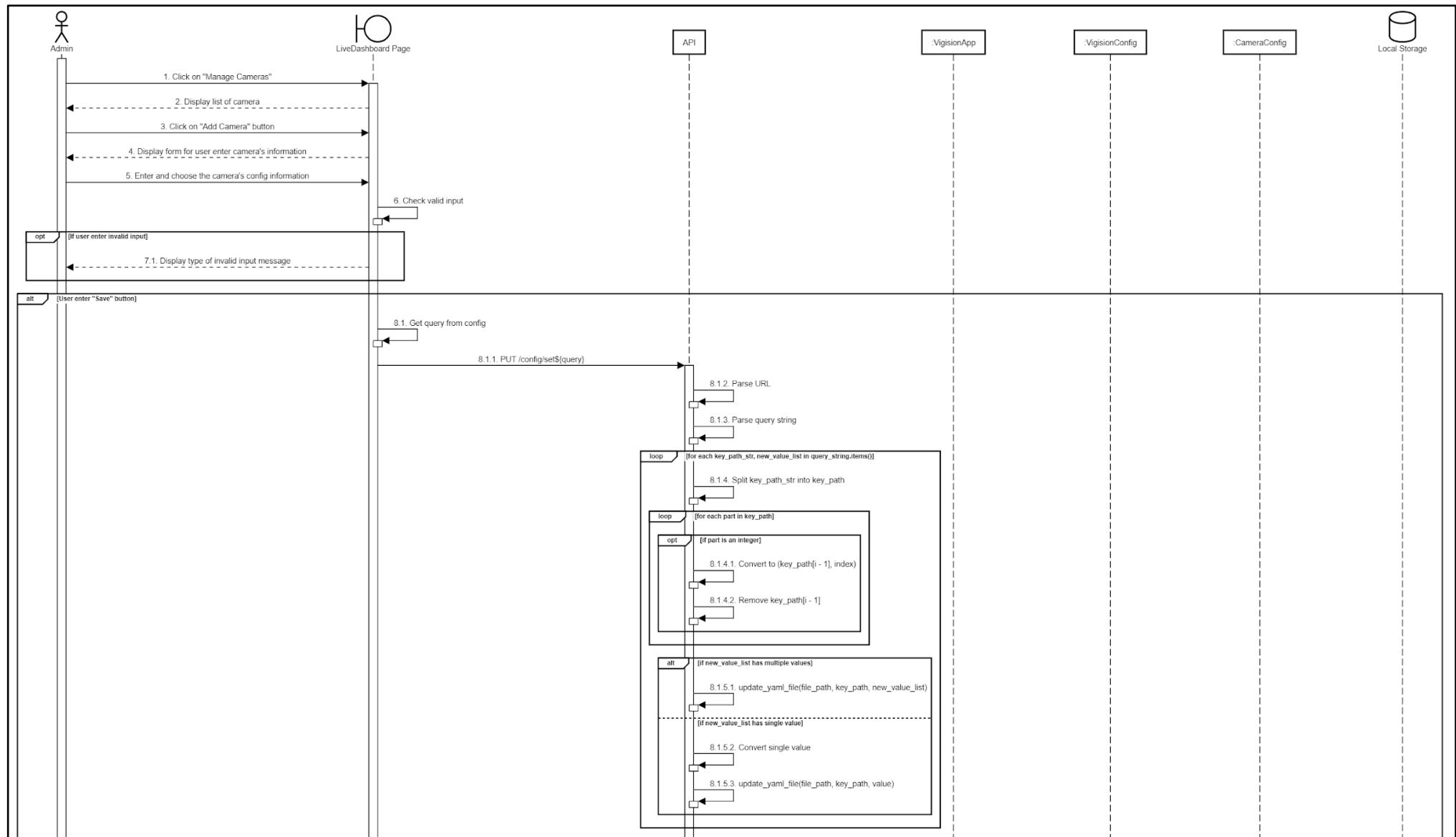


Figure 181. Sequence Diagram - Add camera

c. 3.6.2.3 Update camera



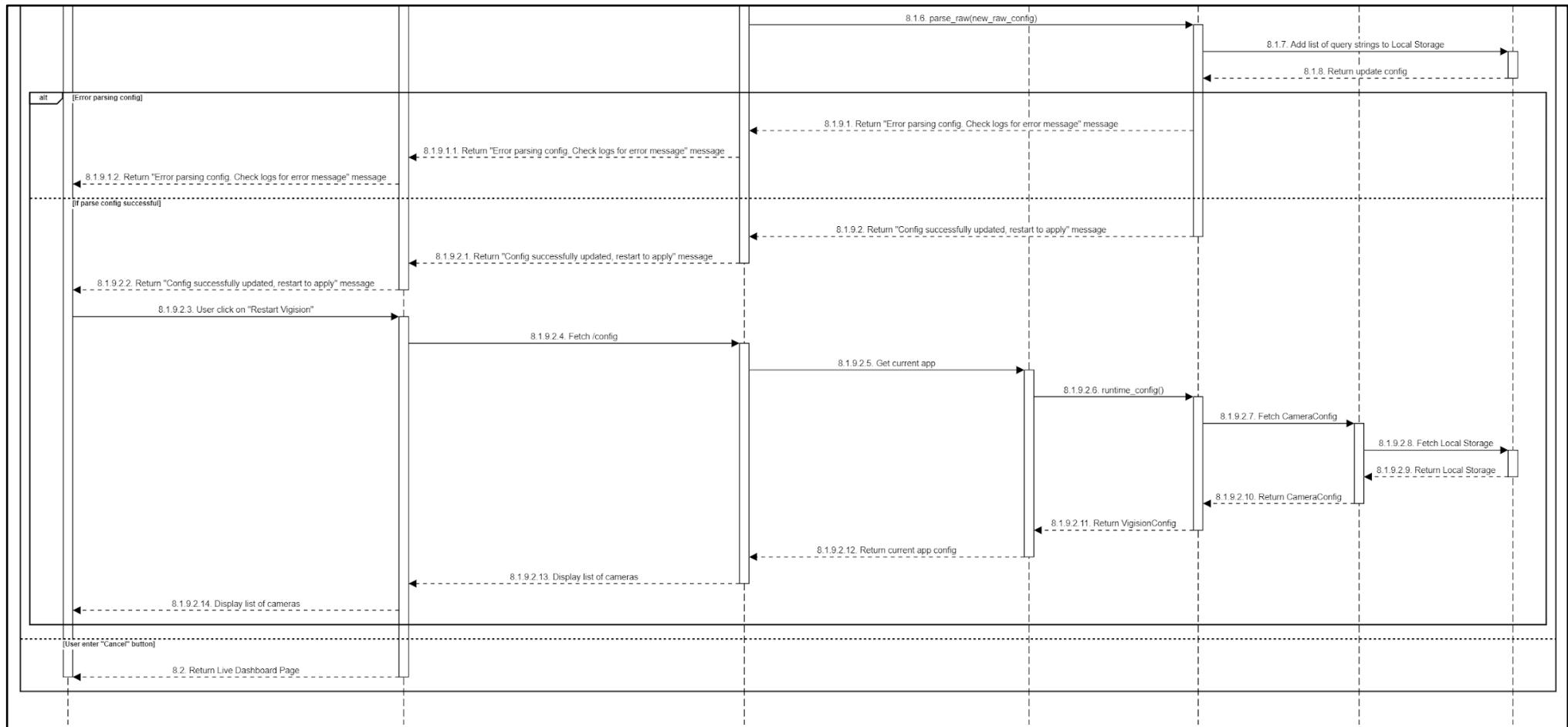
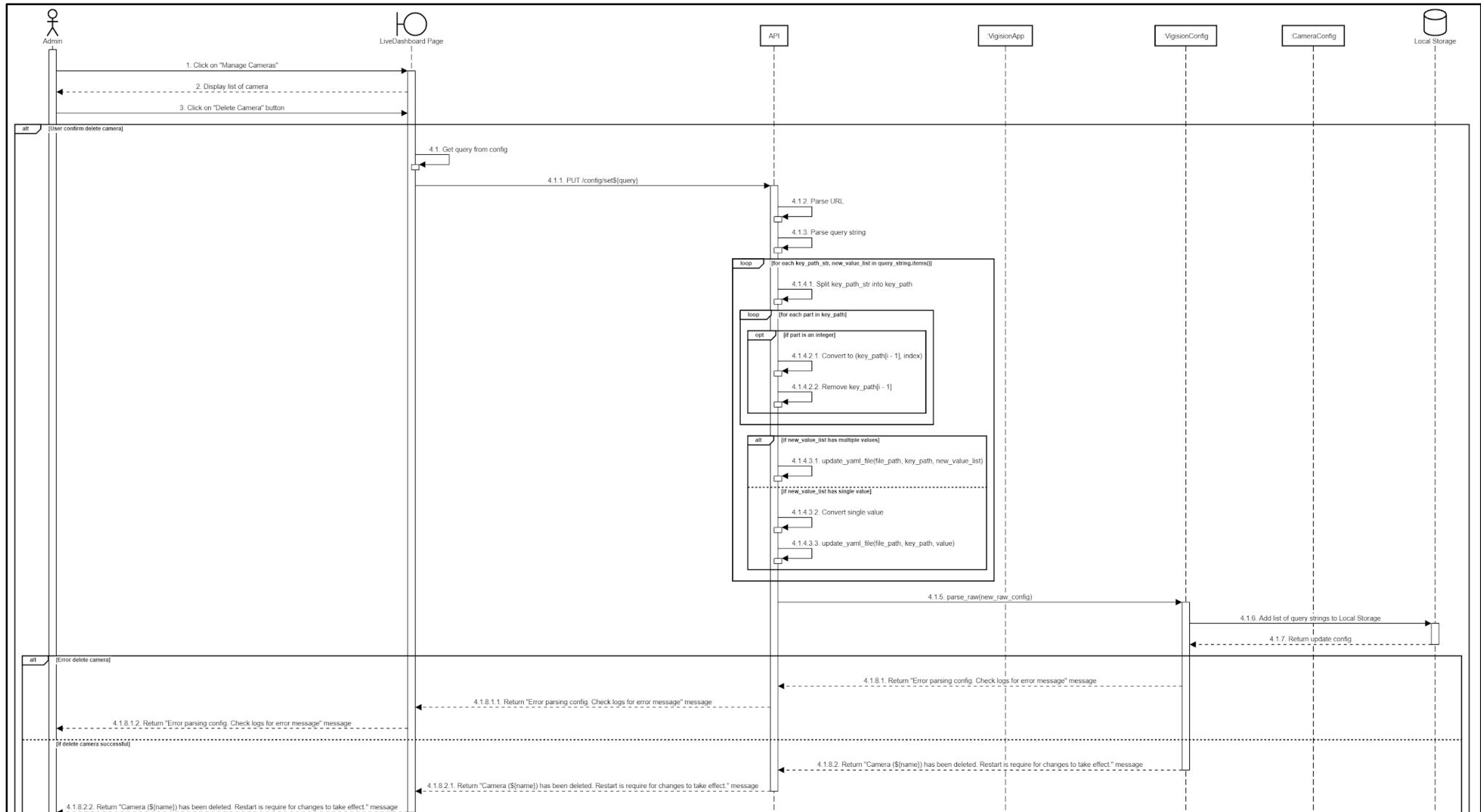


Figure 182. Sequence Diagram - Update camera

d. 3.6.2.4 Delete camera



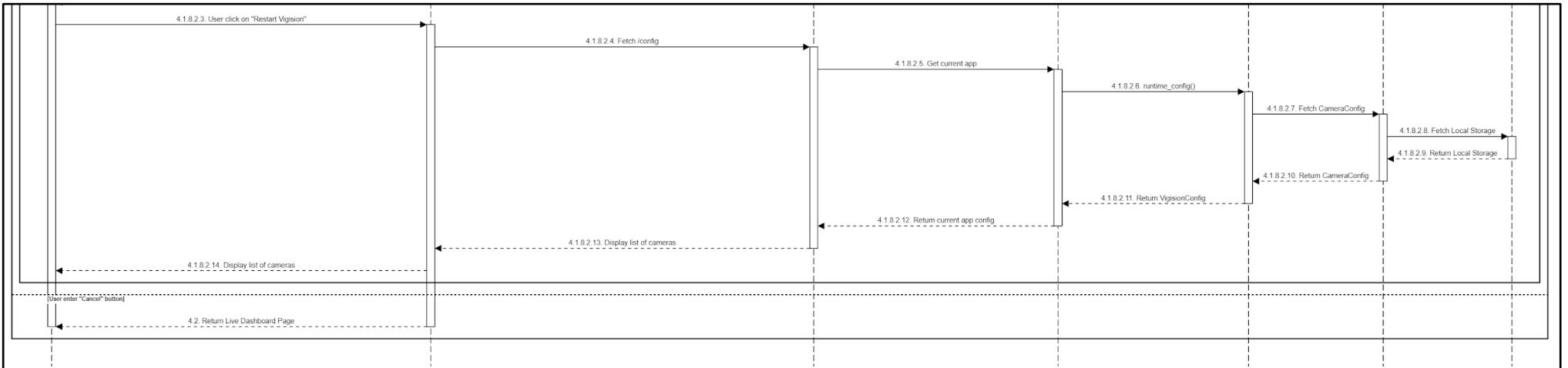


Figure 183. Sequence Diagram - Delete camera

3.7 Manage camera groups

3.7.1 Class Diagram

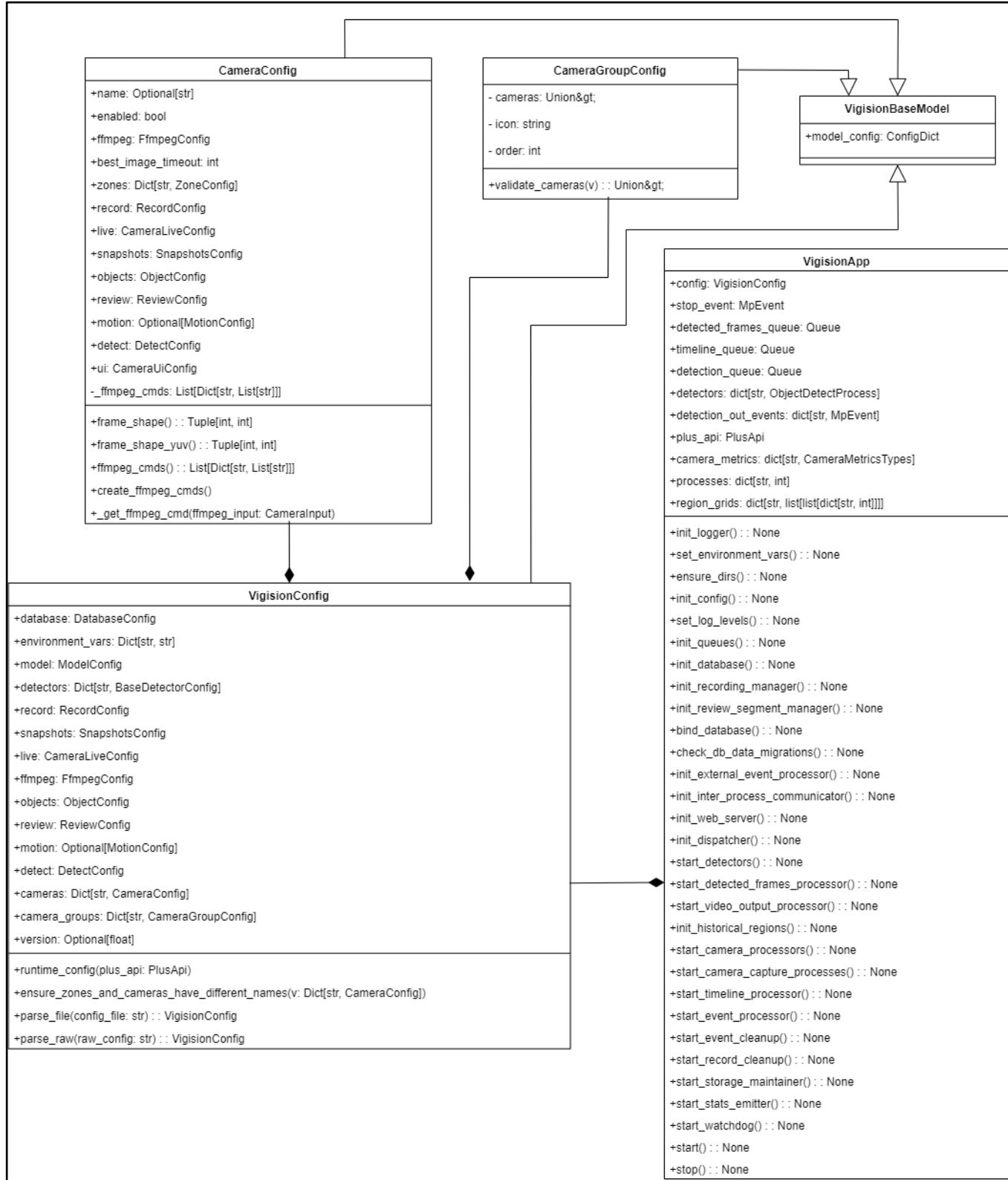


Figure 184. Class Diagram - Manage camera group

3.7.2 Sequence Diagram

a. View list camera group

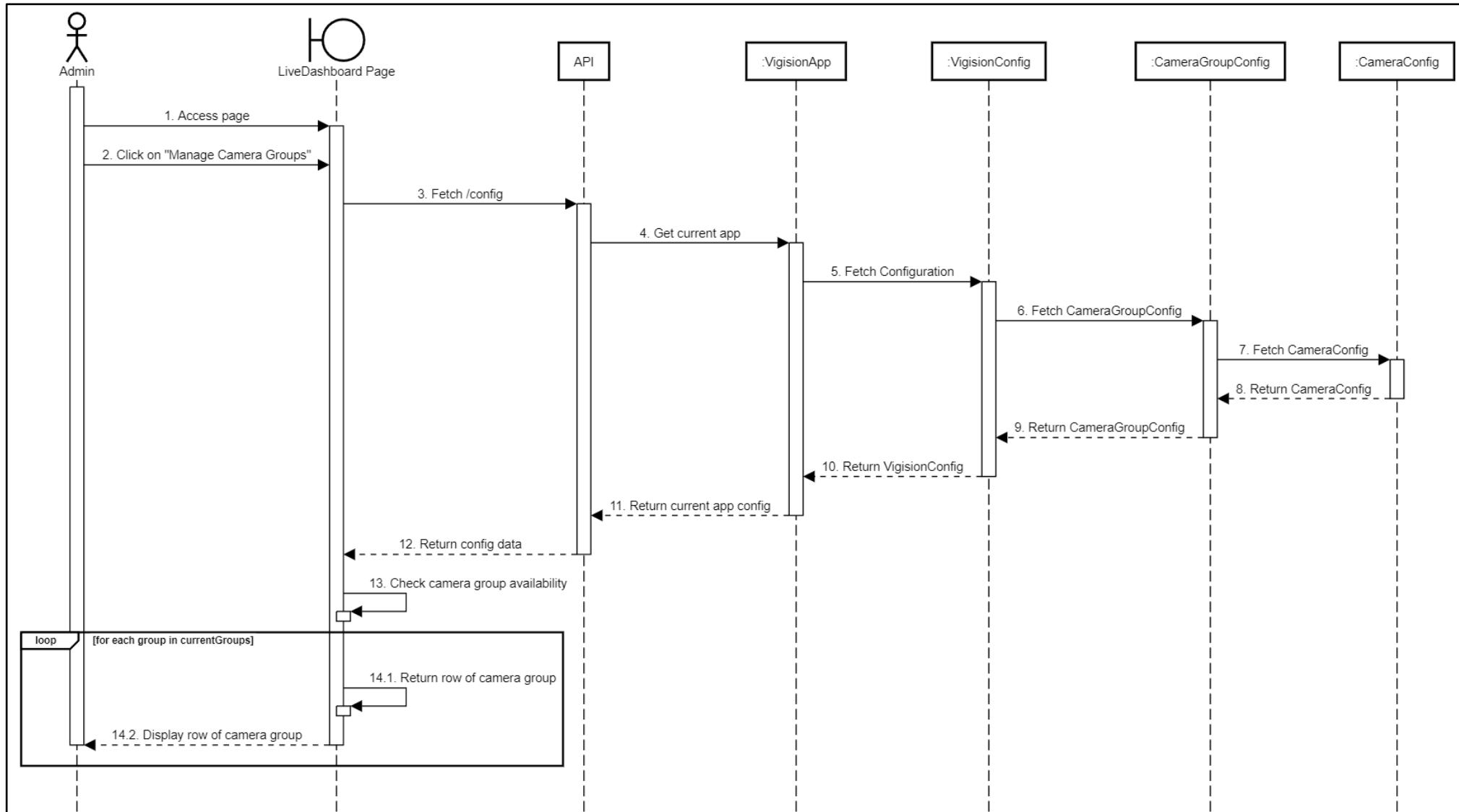


Figure 185. Sequence Diagram - View list camera group

b. Add camera group

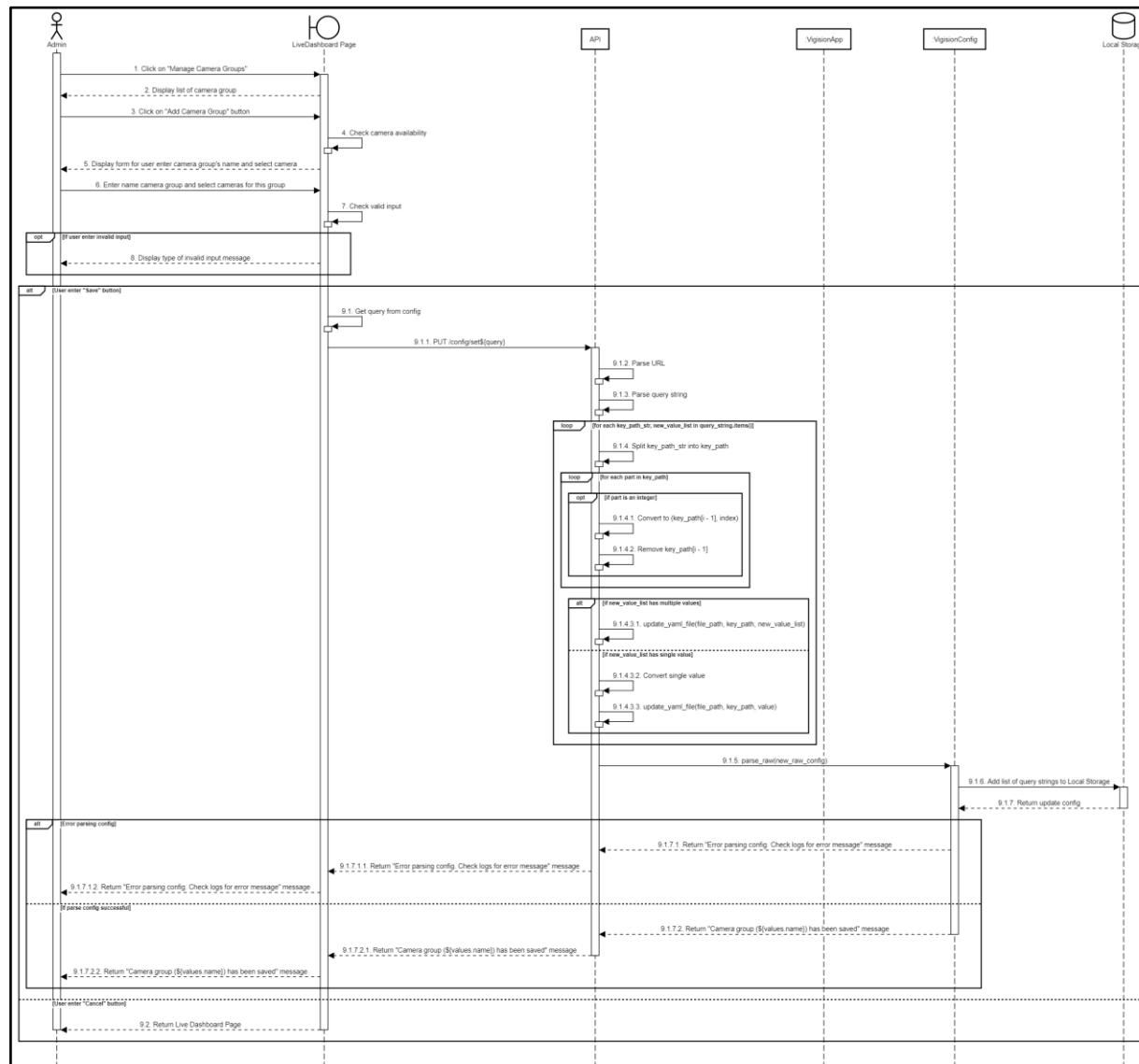


Figure 186. Sequence Diagram - Add camera group

c. Update camera group

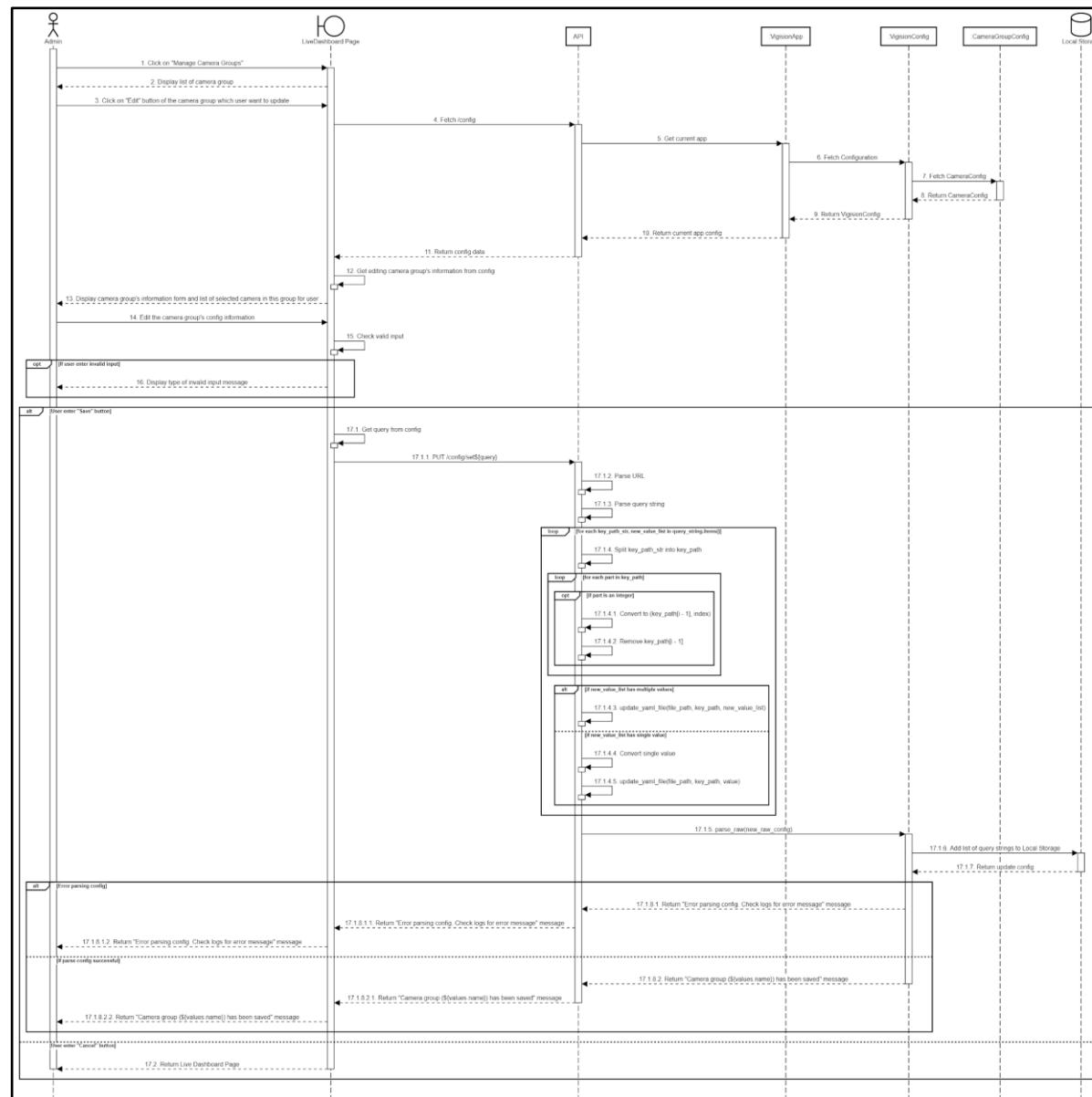


Figure 187. Sequence Diagram - Update camera group

d. Delete camera group

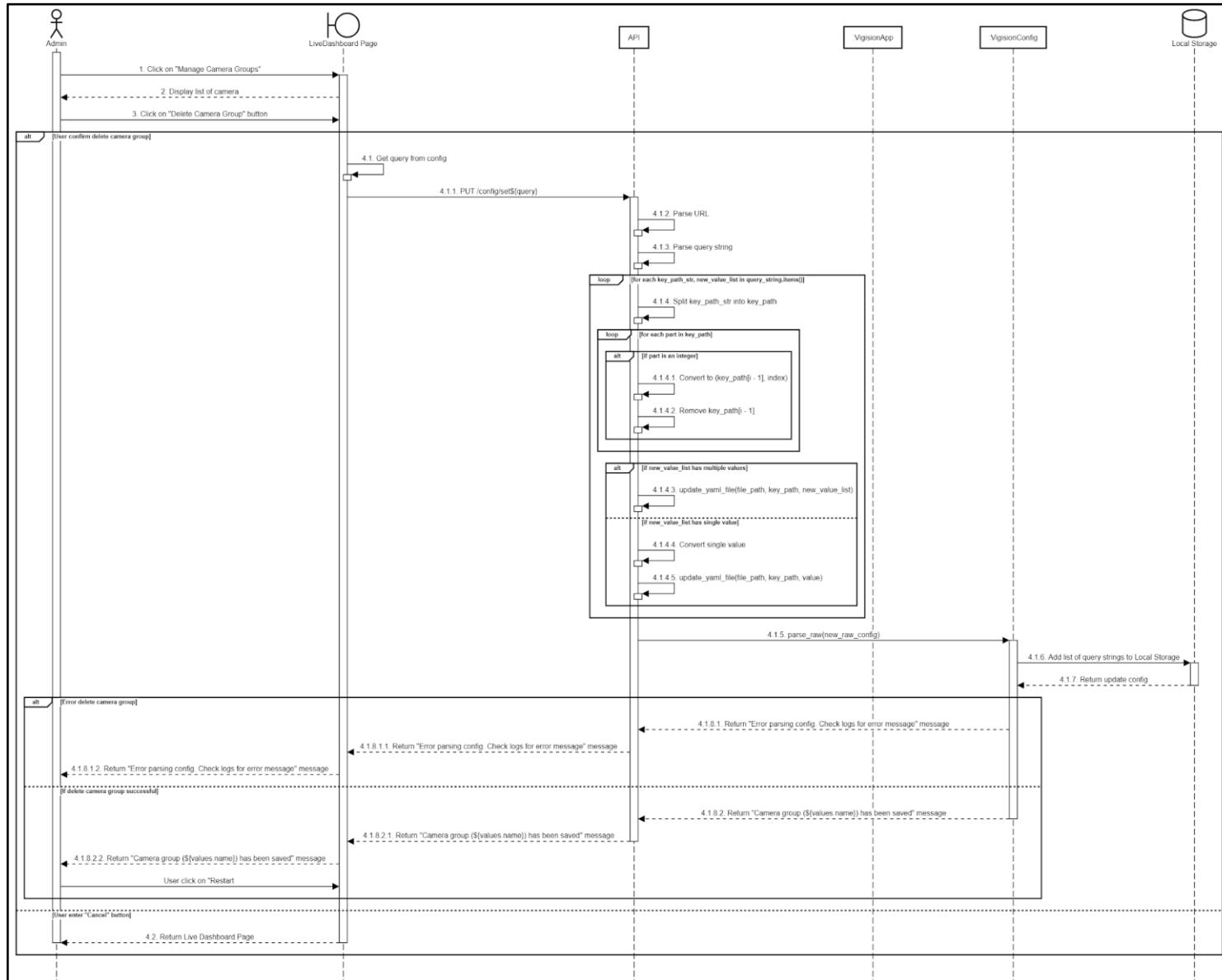


Figure 188. Sequence Diagram - Delete camera group

3.8 Edit camera group layout

3.8.1 Class Diagram

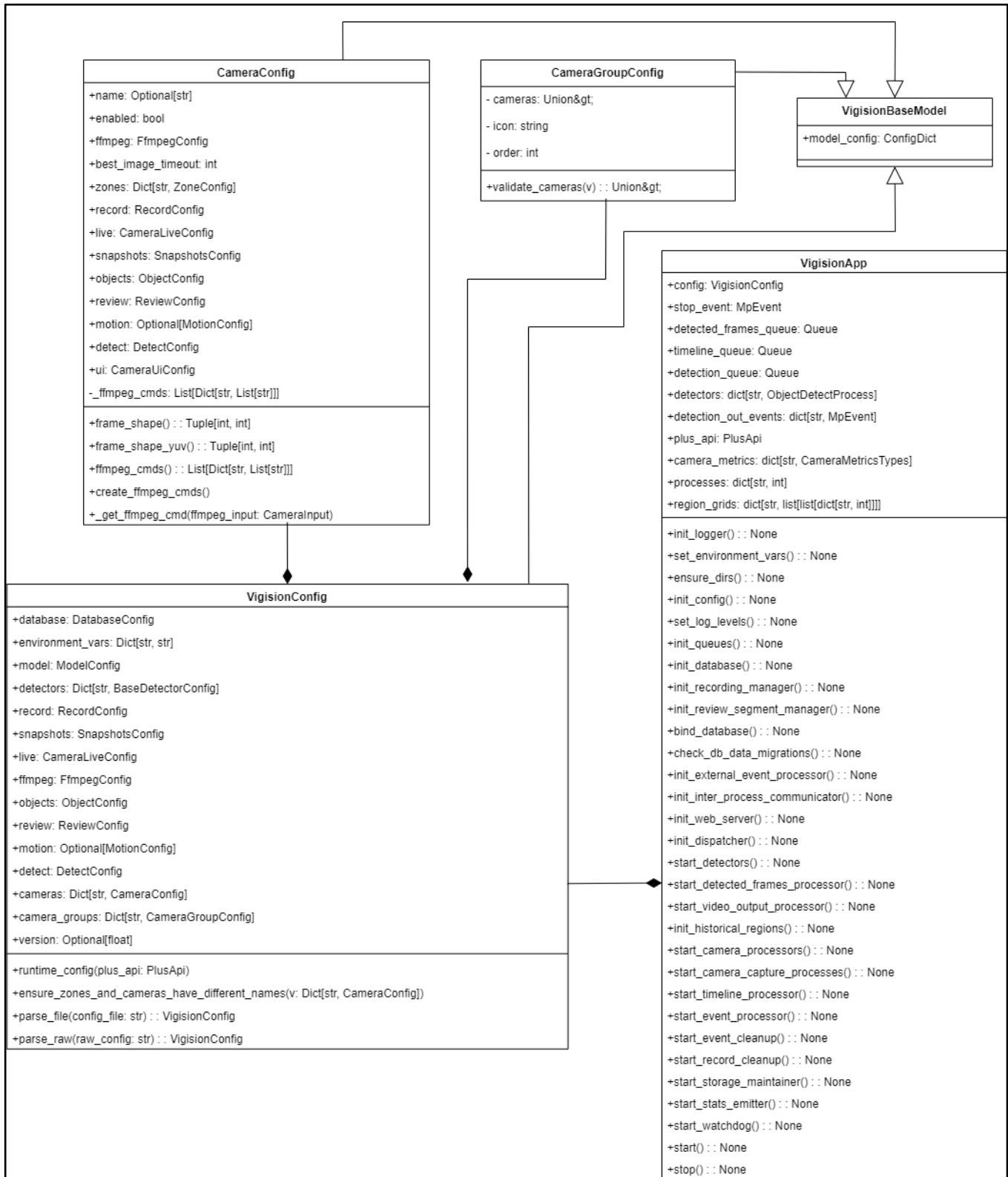


Figure 189. Class Diagram - Edit camera group layout

3.8.2 Sequence Diagram

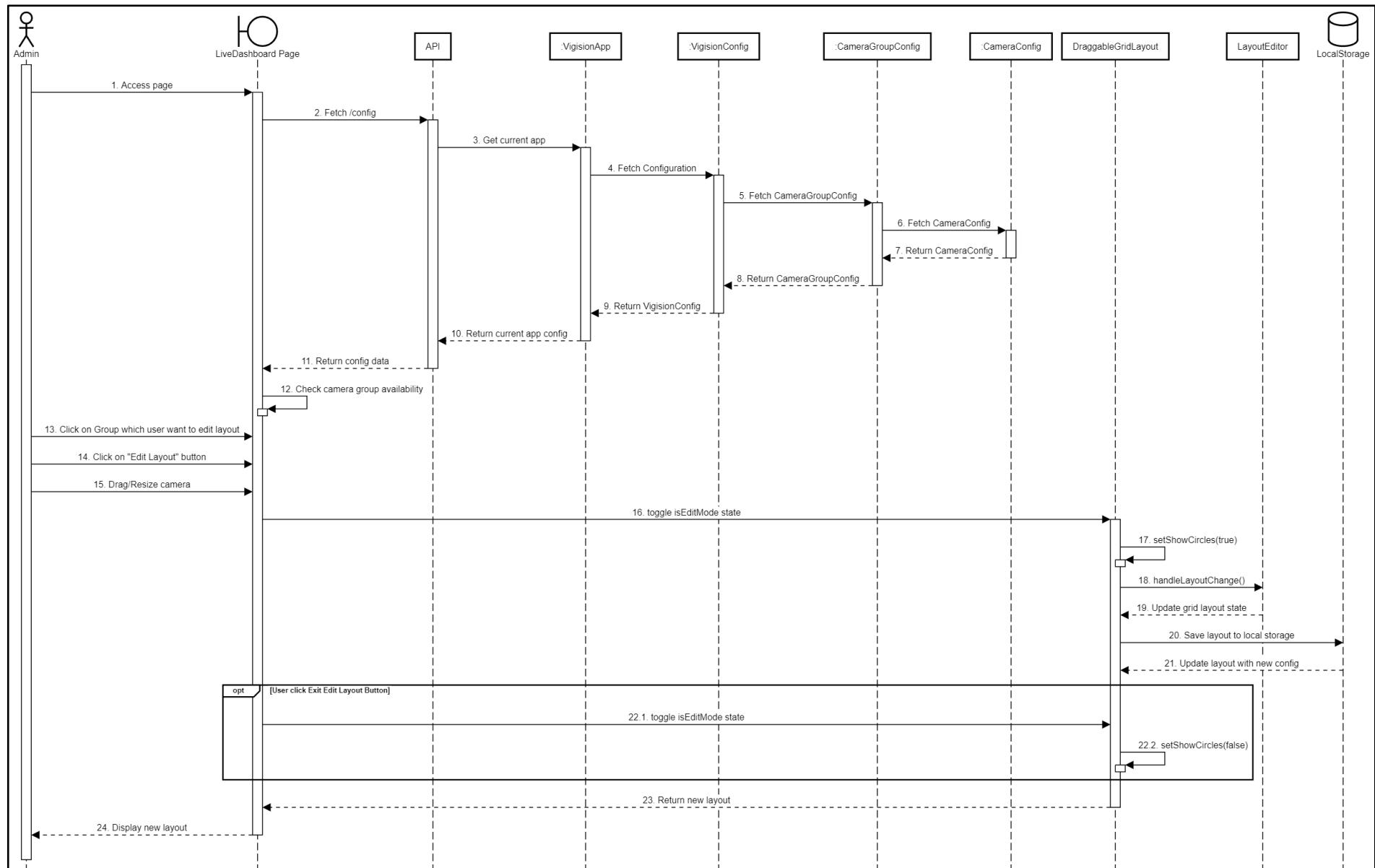


Figure 190. Sequence Diagram - Edit camera group layout

3.9 View in full screen

3.9.1 Class Diagram

This is frontend feature so don't have class diagram

3.9.2 Sequence Diagram

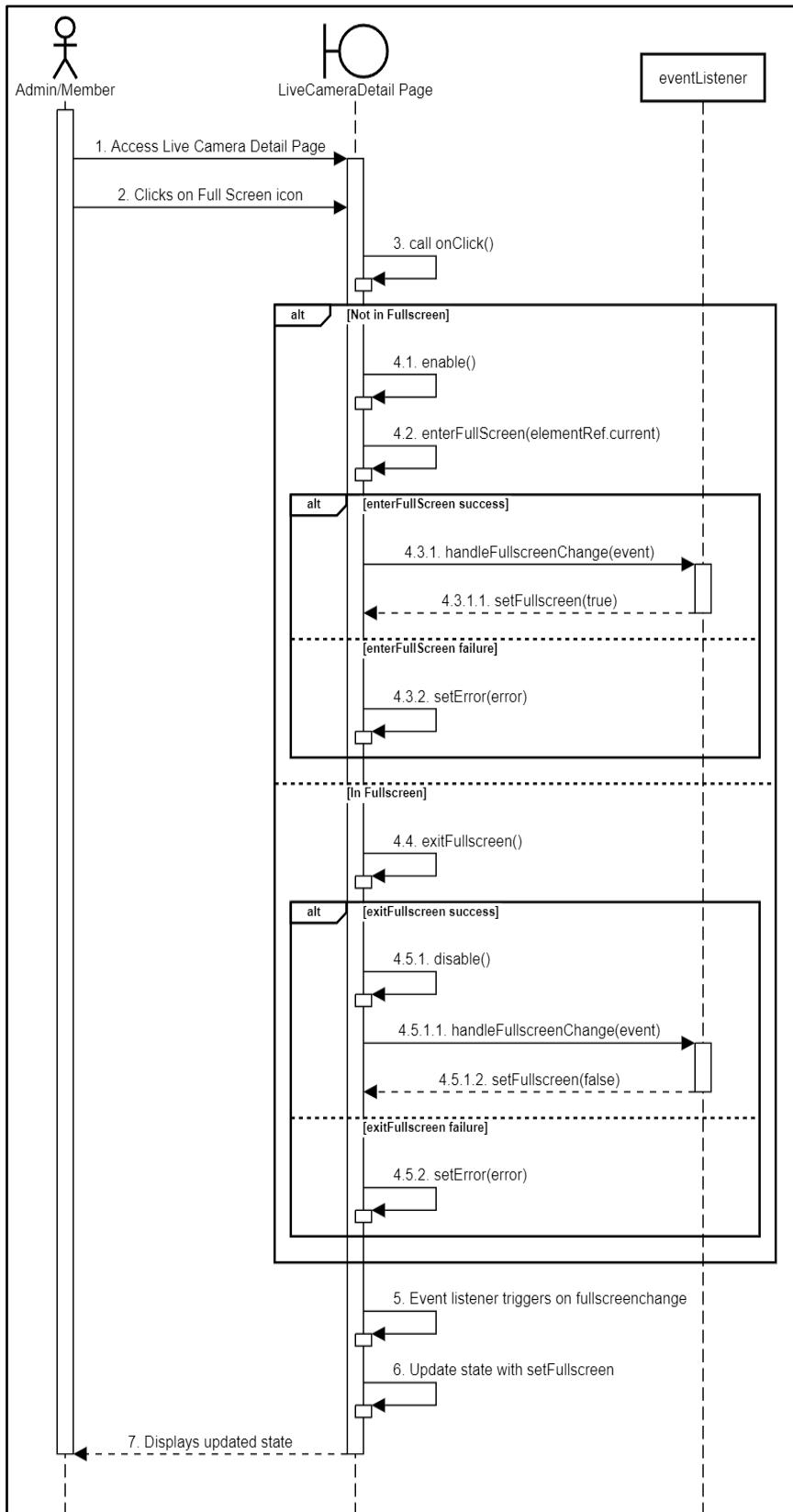


Figure 191. Sequence Diagram - View in full screen

3.10 View detail of live camera

3.10.1 Class Diagram

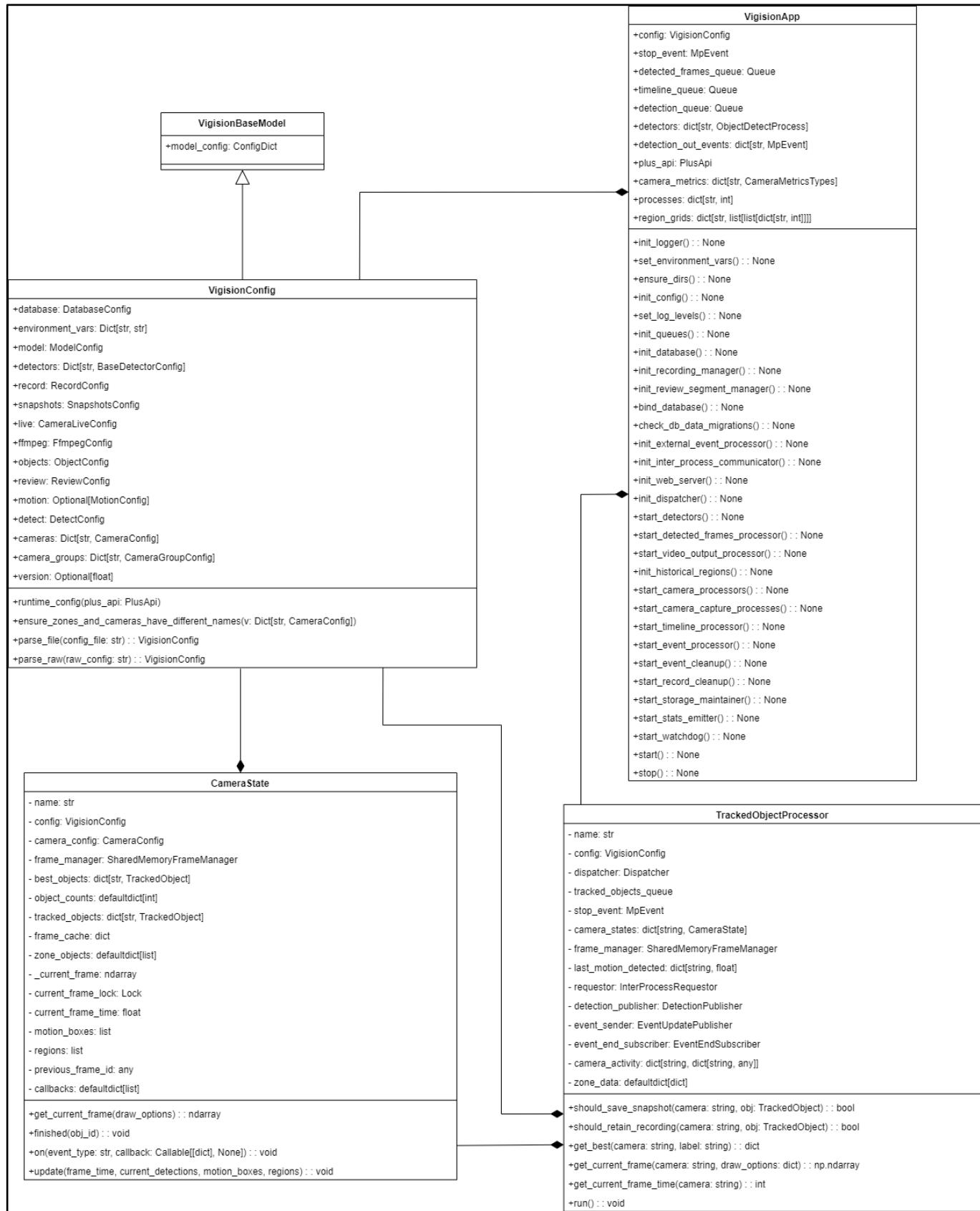
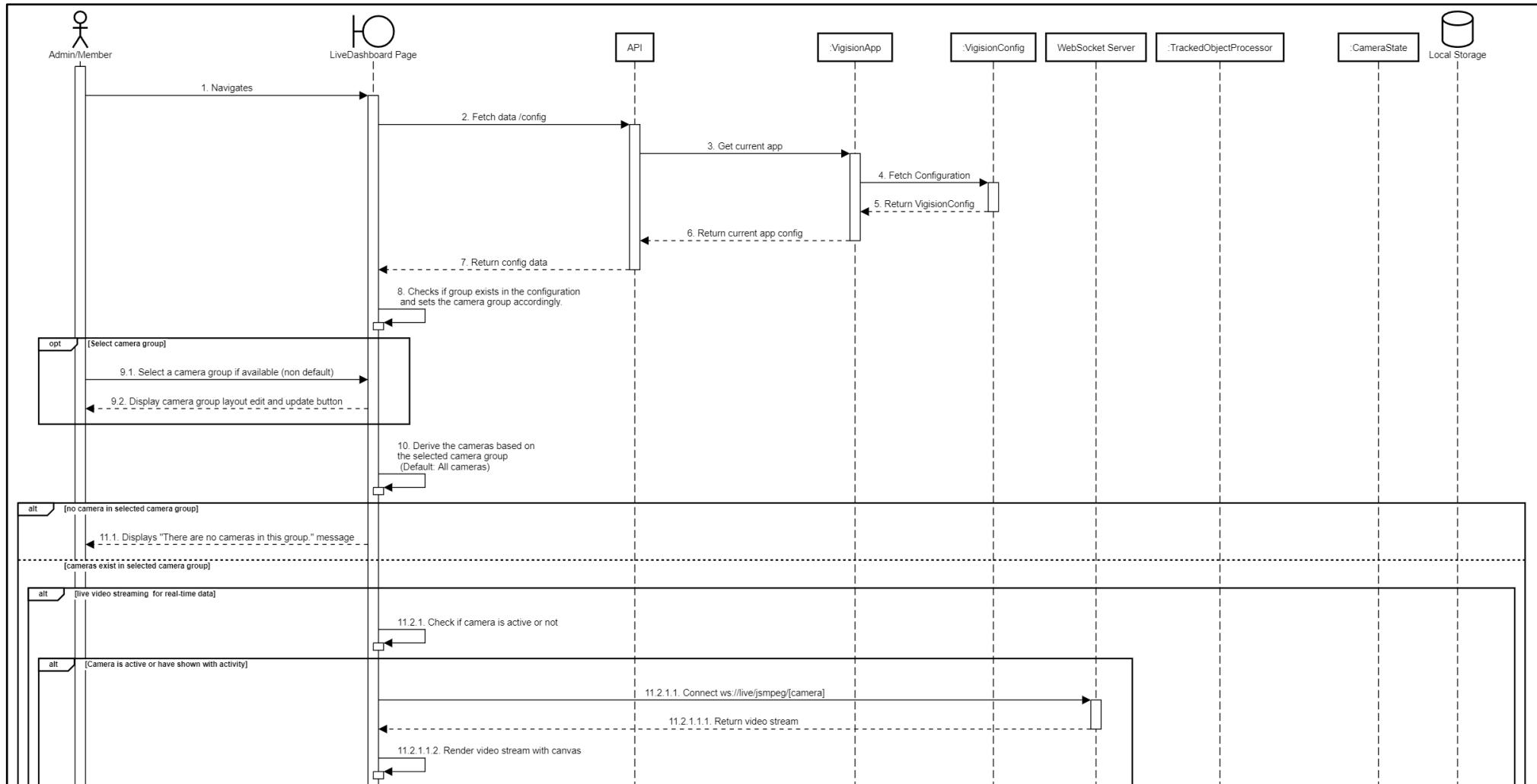


Figure 192. Class Diagram - View detail of live camera

3.10.2 Sequence Diagram



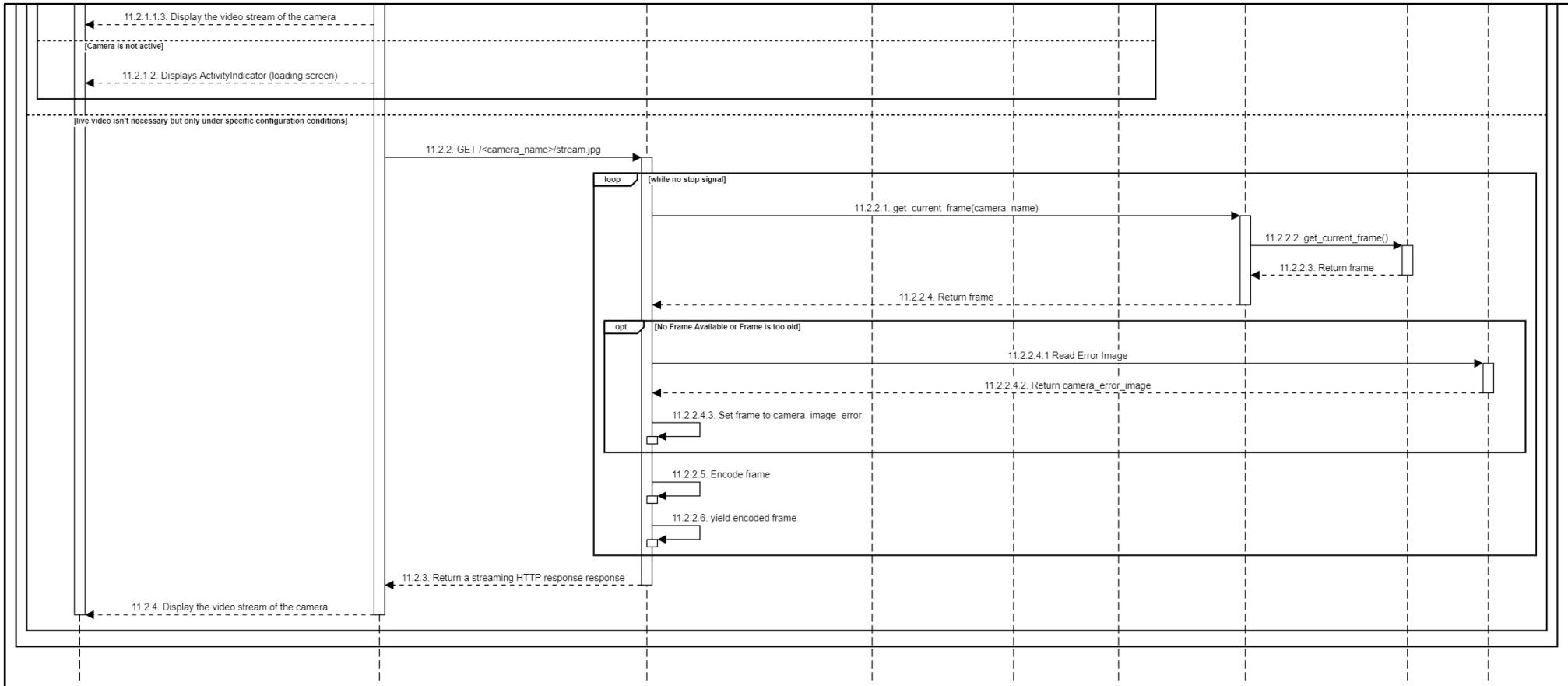


Figure 193. Sequence Diagram - View detail of live camera

3.11 Get shareable link

3.11.1 Class Diagram

This is a front end function. Only call API so that it does not have class diagram

3.11.2 Sequence Diagram

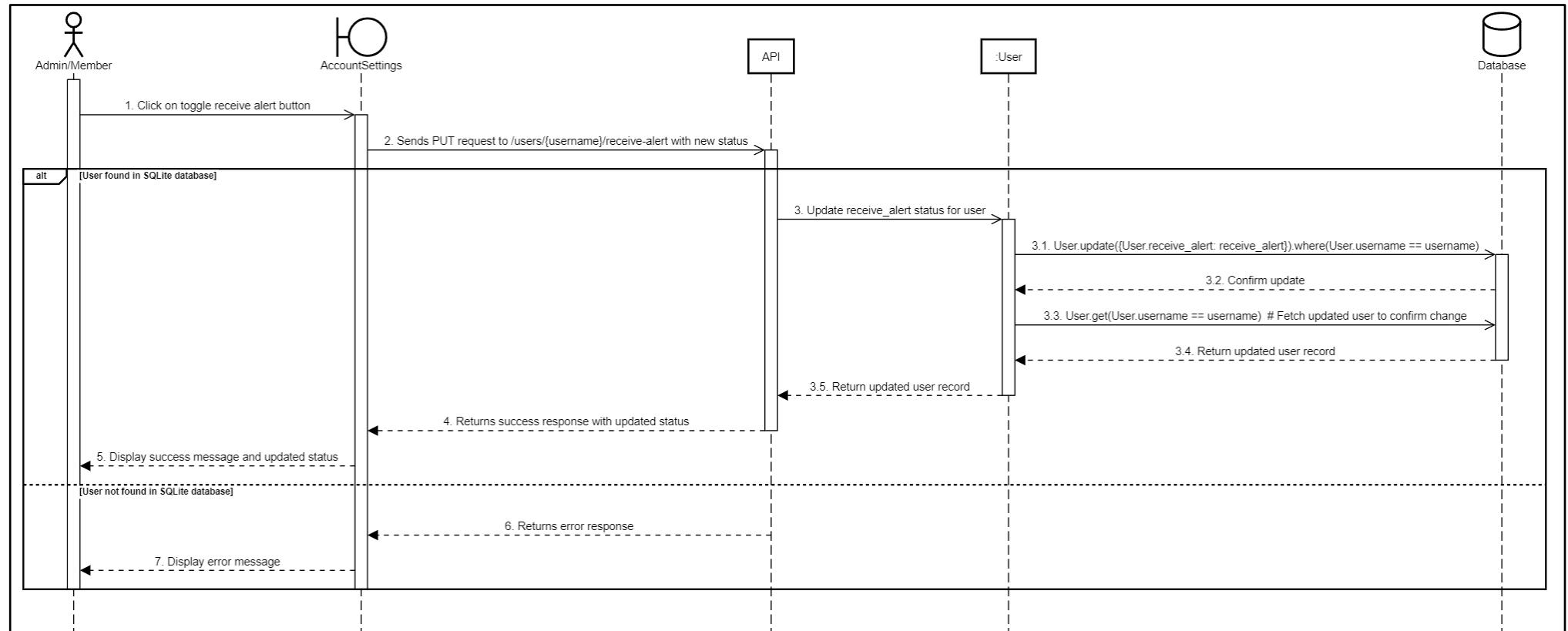


Figure 194. Sequence Diagram - Get shareable link

3.12 Enable/disable detection

3.12.1 Class Diagram

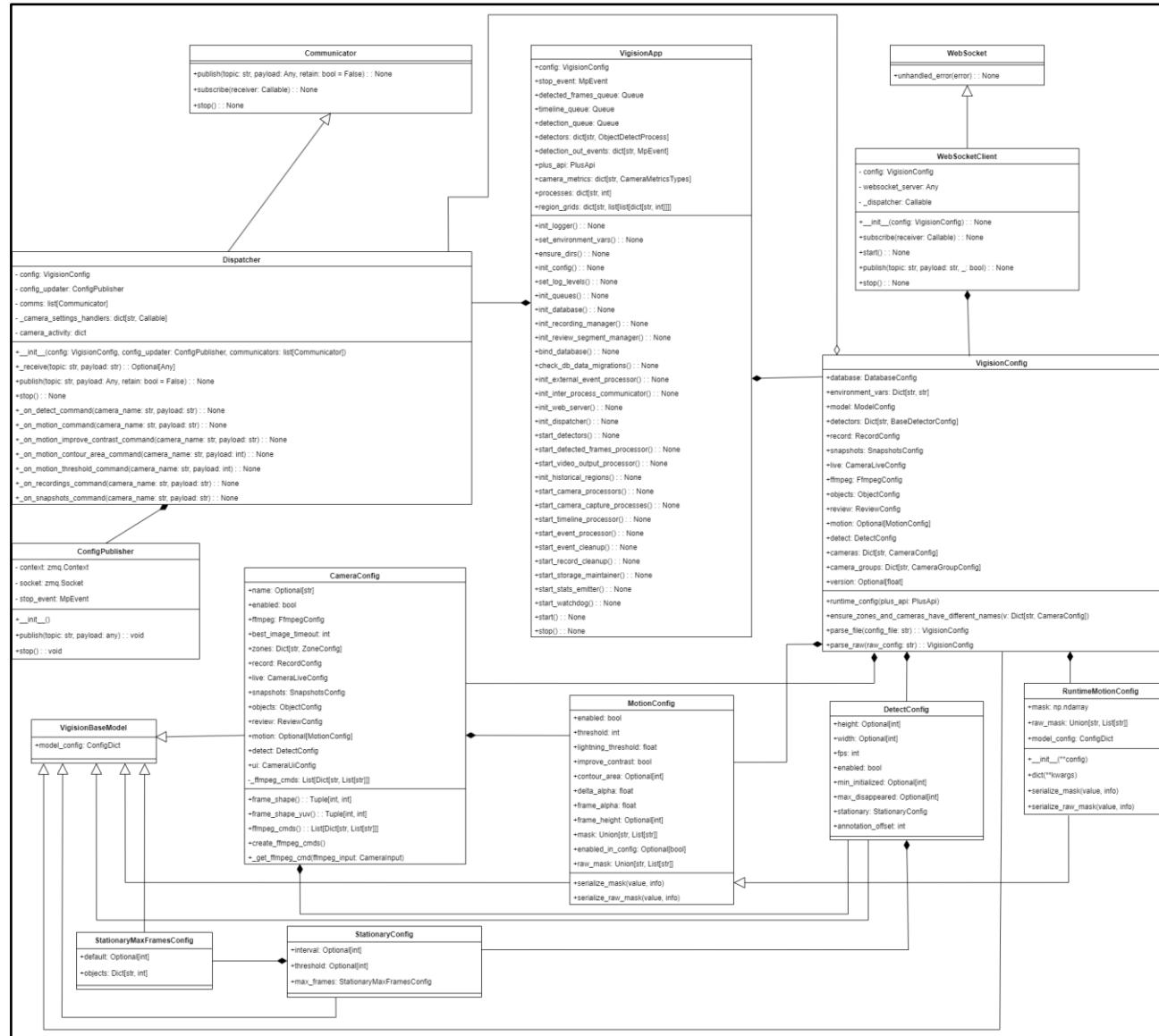


Figure 195. Class Diagram - Enable/disable detection

3.12.2 Sequence Diagram

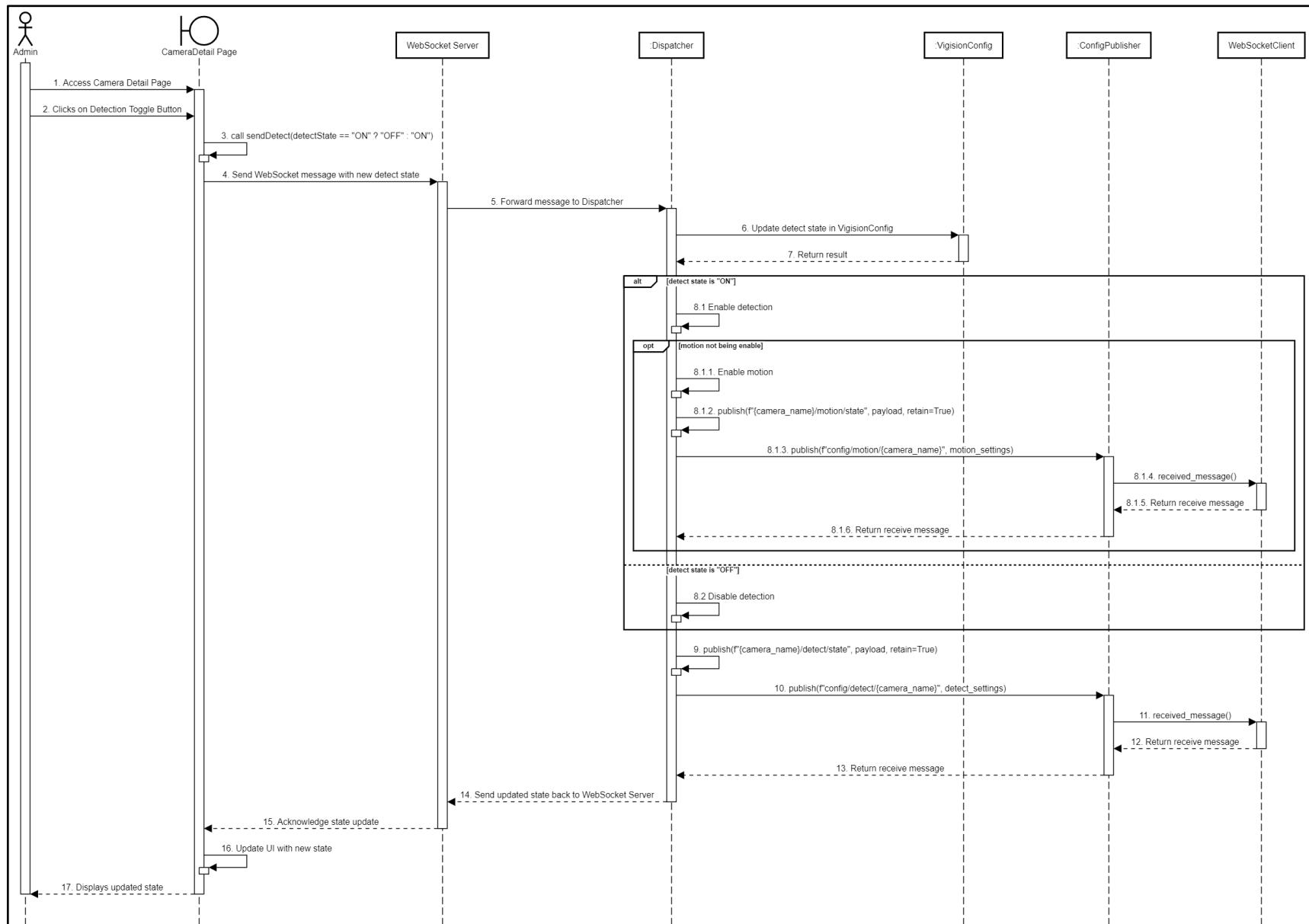


Figure 196. Sequence Diagram - Enable/disable detection

3.13 Enable/disable recording

3.13.1 Class Diagram

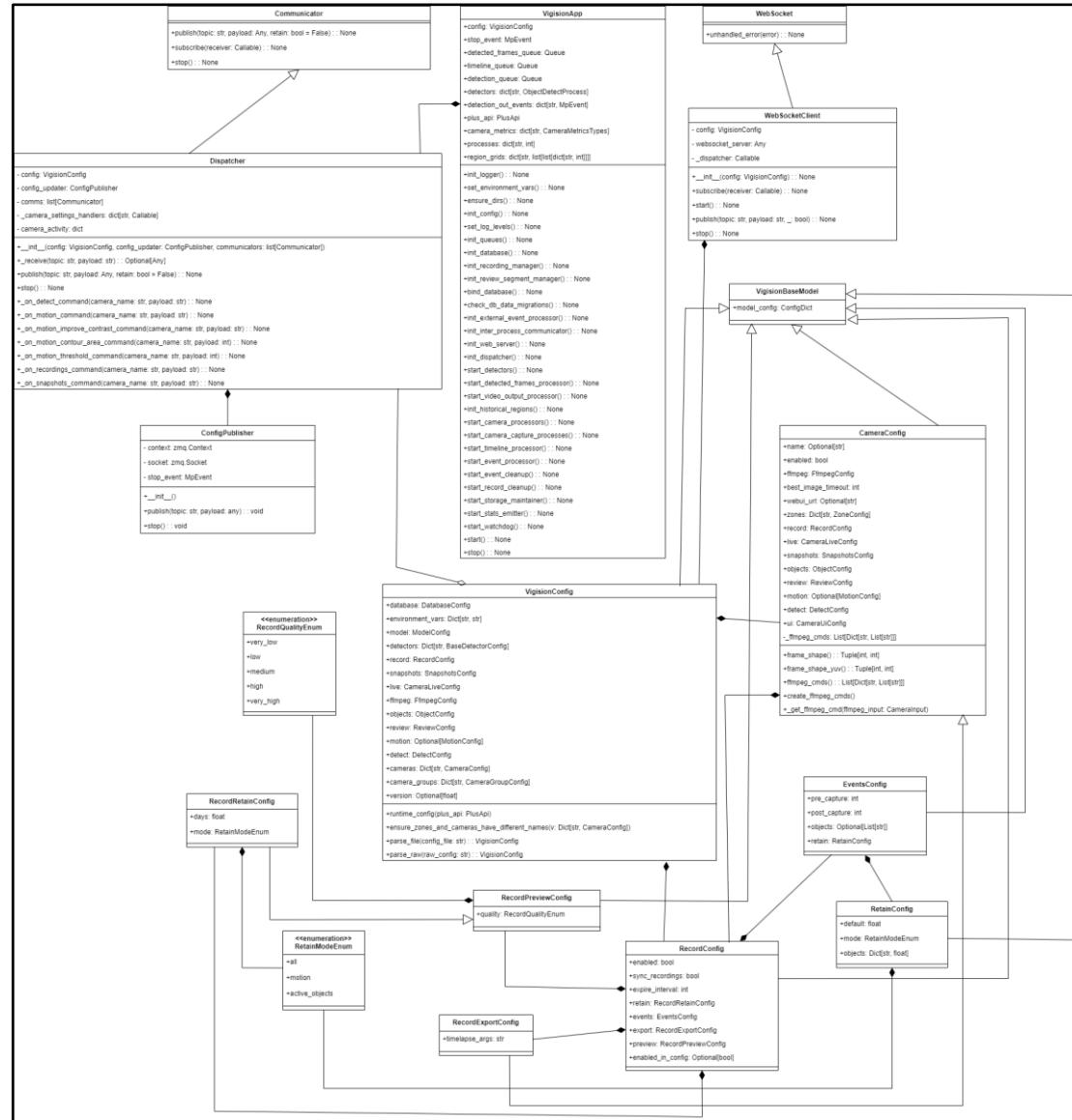


Figure 197. Class Diagram - Enable/disable recording

3.13.2 Sequence Diagram

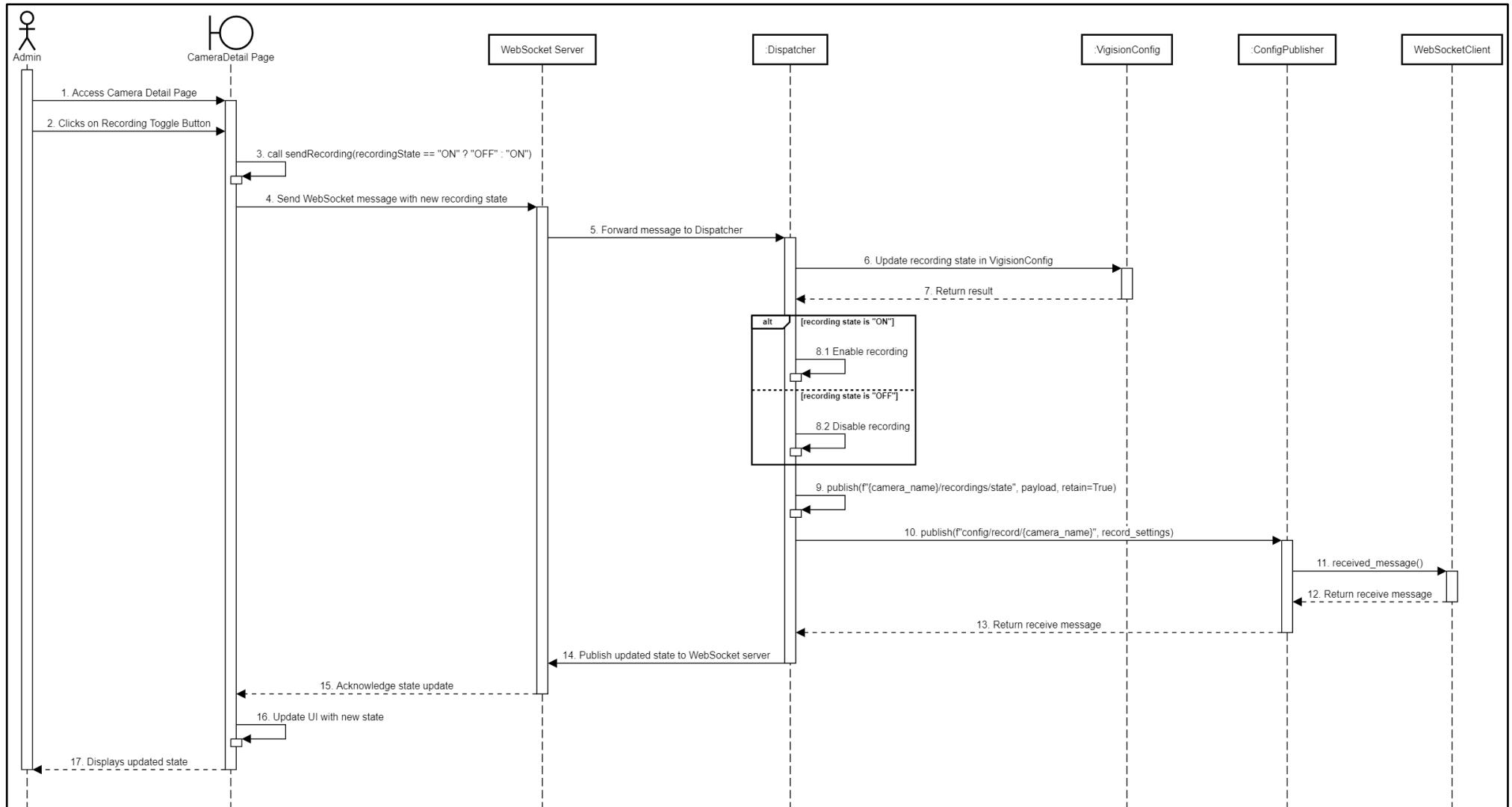


Figure 198. Sequence Diagram - Enable/disable recording

3.14 Enable/disable snapshots

3.14.1 Class Diagram

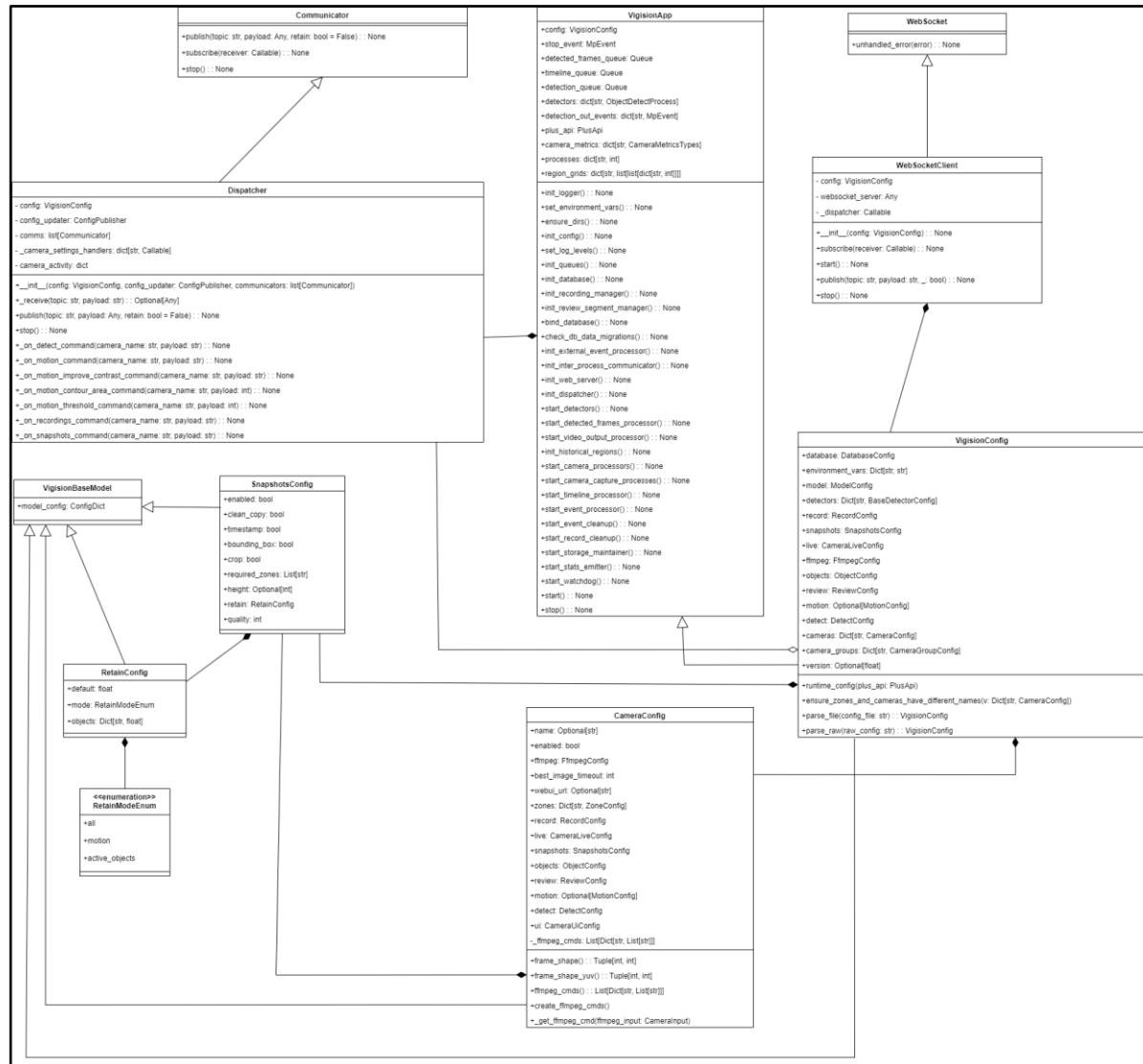


Figure 199. Class Diagram - Enable/disable snapshots

3.14.2 Sequence Diagram

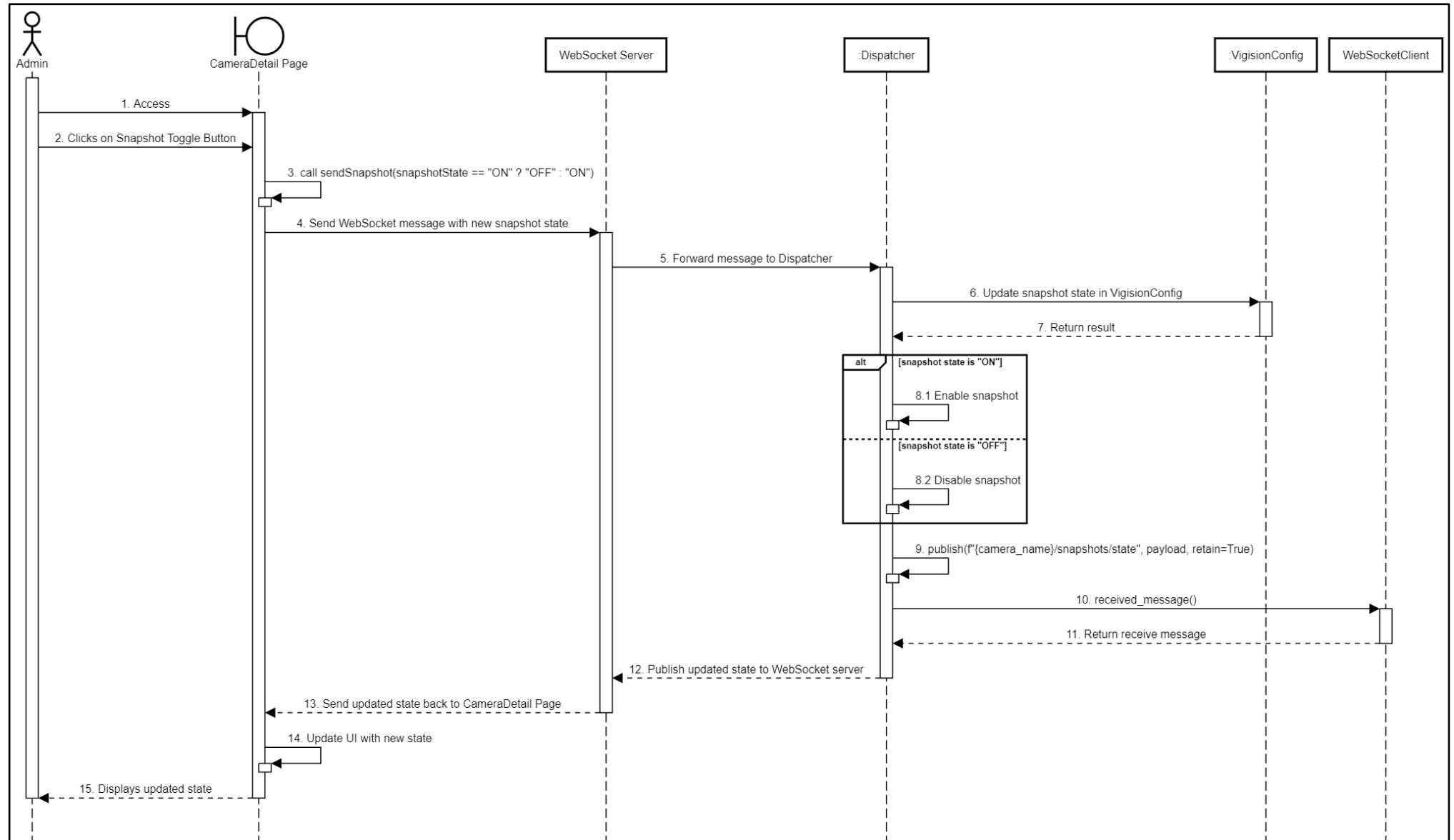


Figure 200. Sequence Diagram - Enable/disable snapshots

3.15 View defined masks and zones

3.15.1 Class Diagram

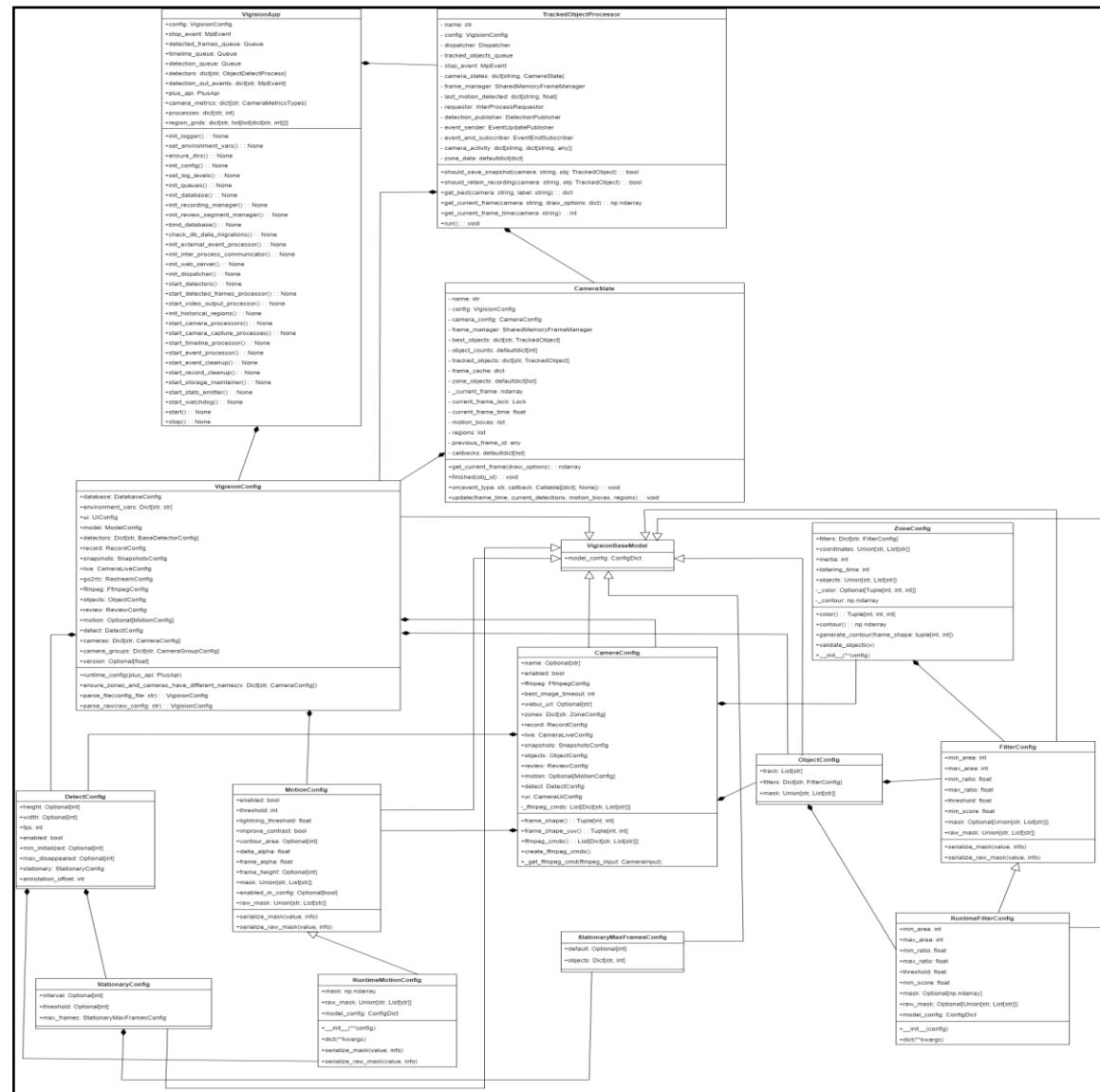
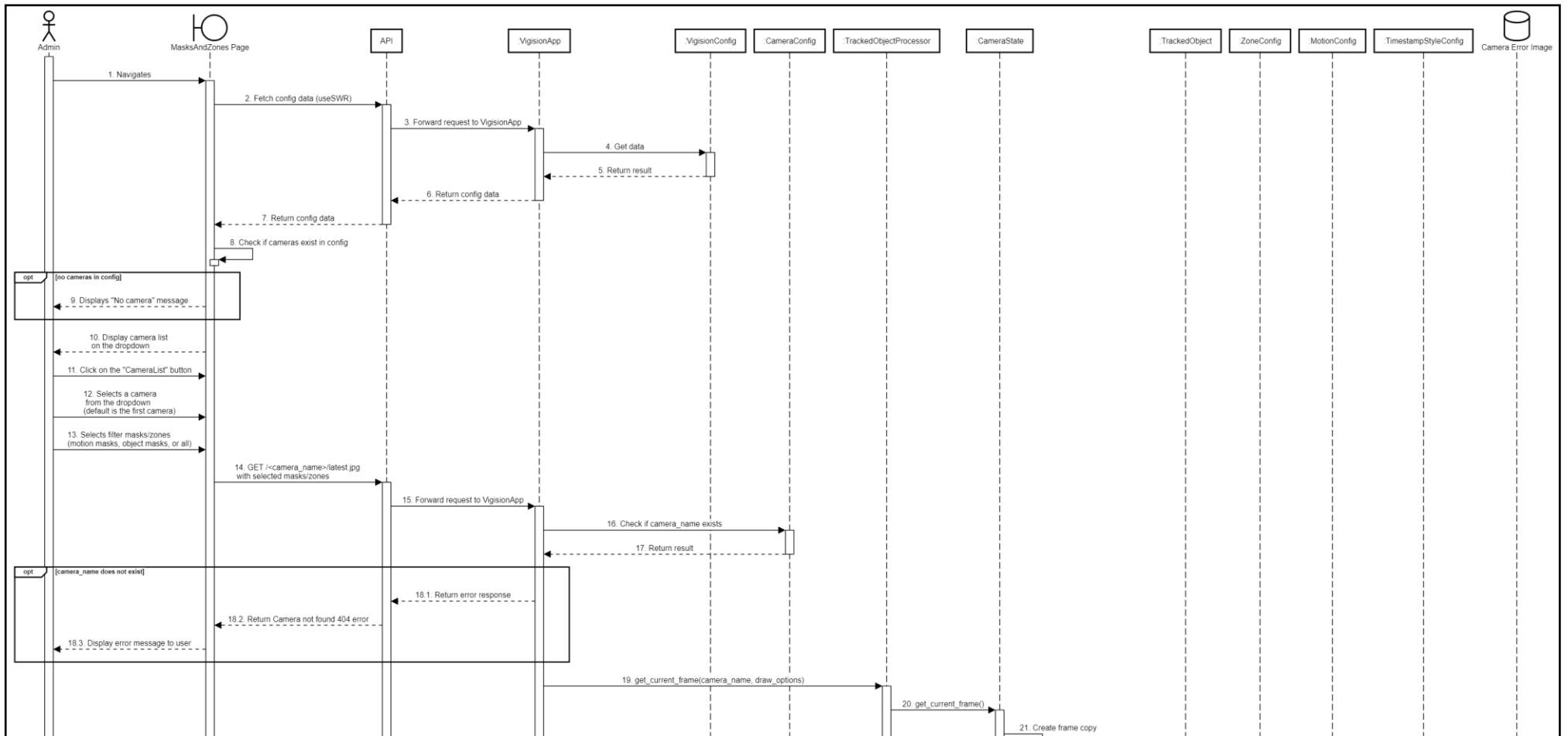
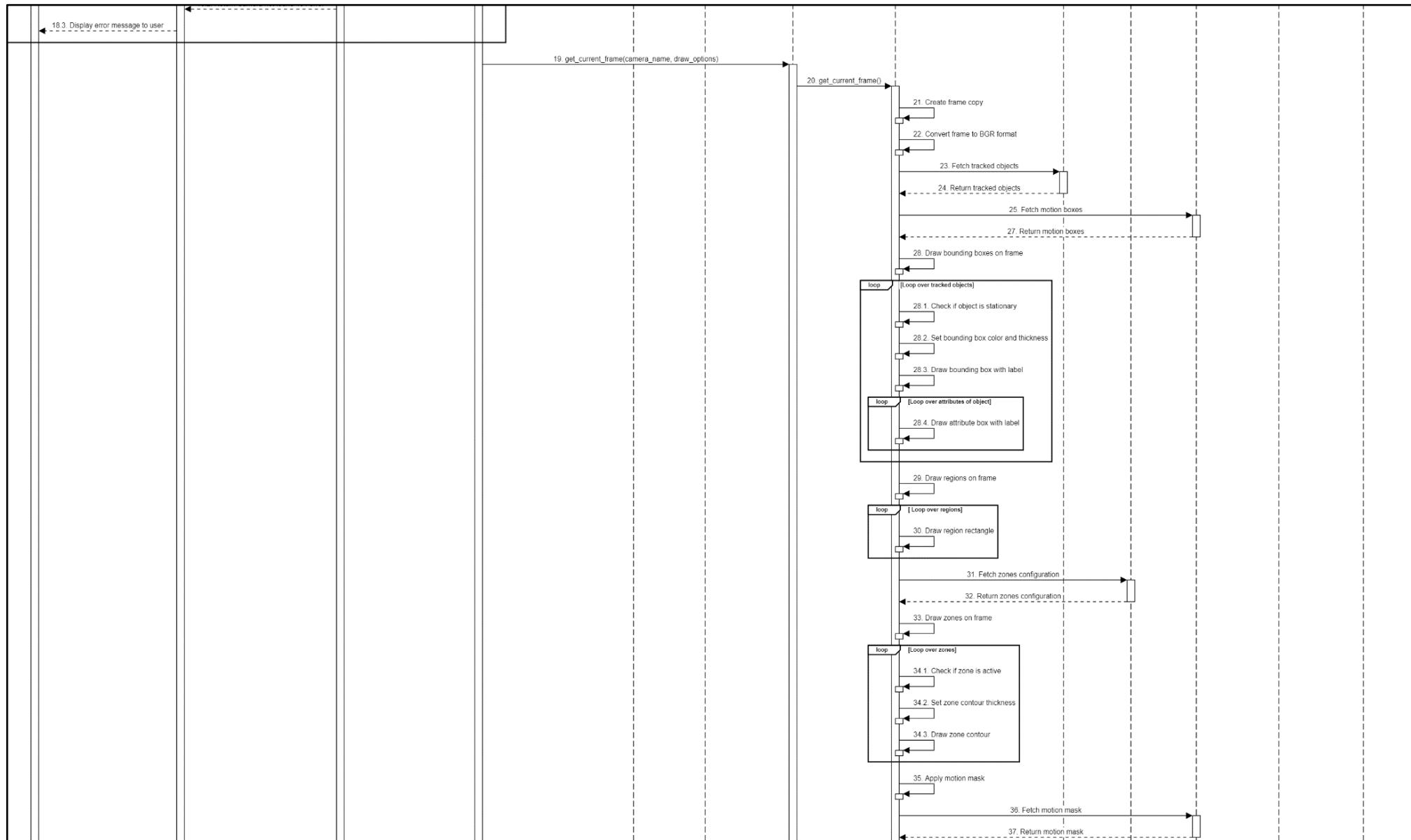


Figure 201. Class Diagram - View defined masks and zones

3.15.2 Sequence Diagram





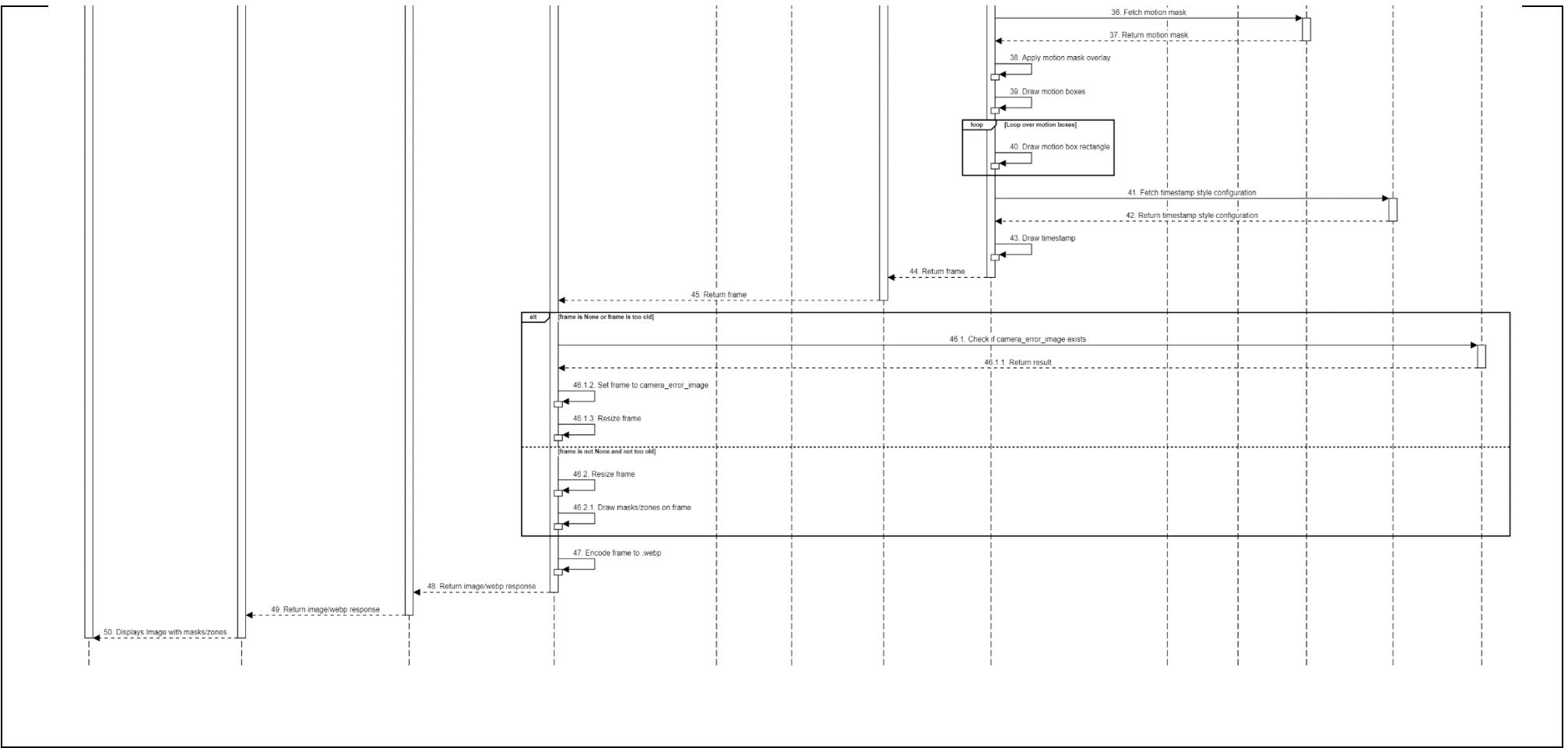


Figure 202. Sequence Diagram - View defined masks and zones

3.16 Manage motion masks

3.16.1 Class Diagram

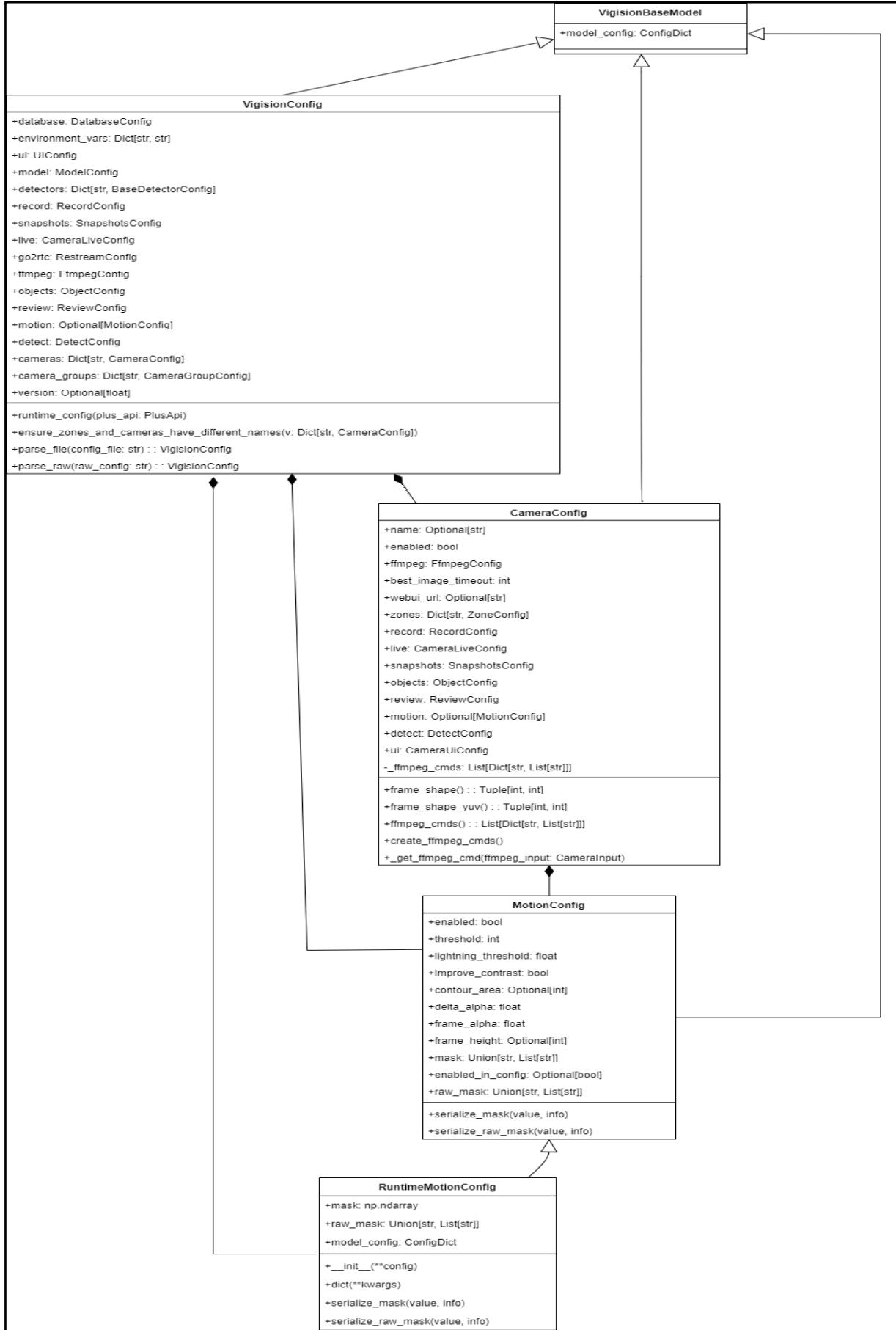
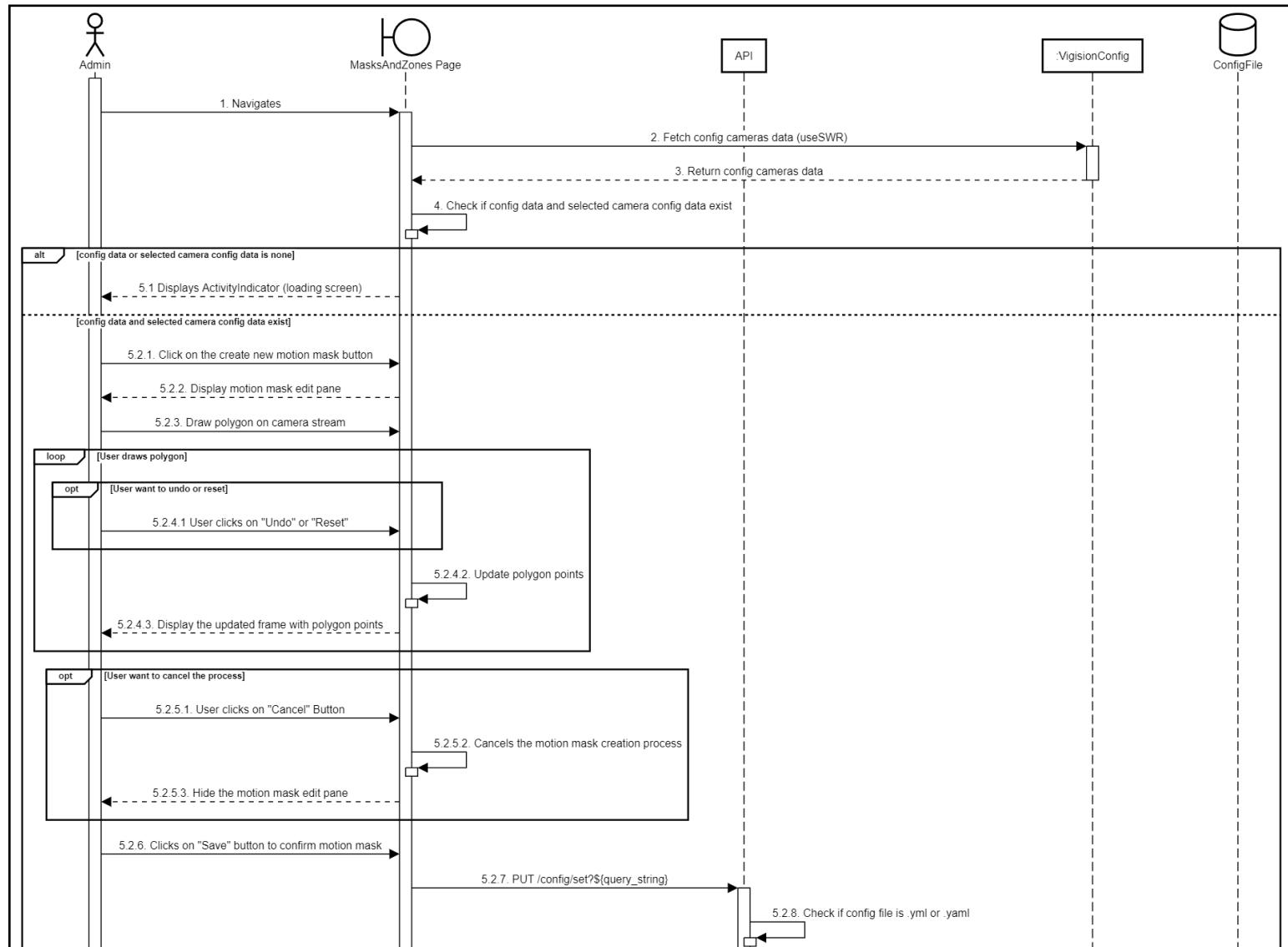


Figure 203. Class Diagram - Manage motion masks

3.16.2 Sequence Diagram

a. Create new motion mask



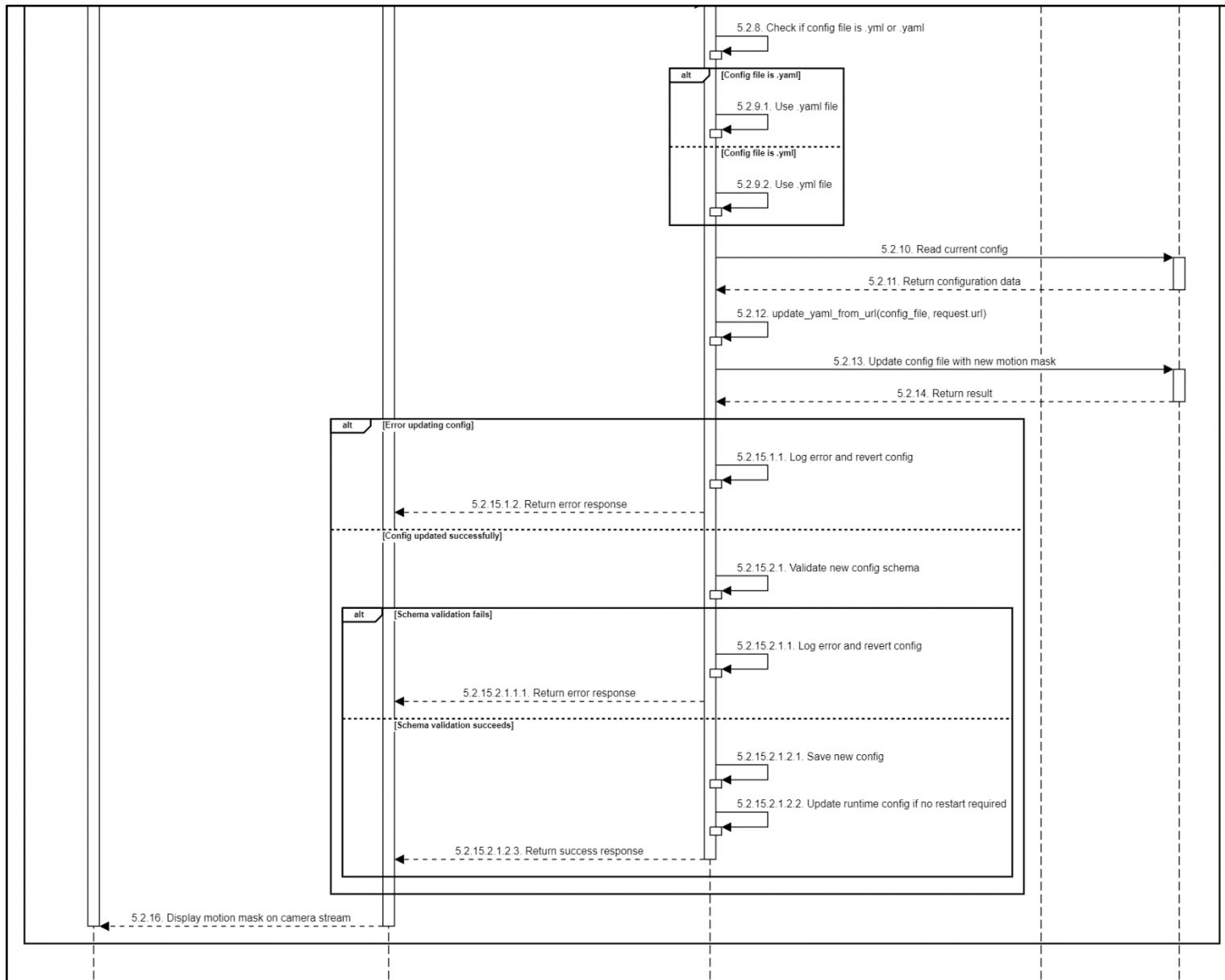
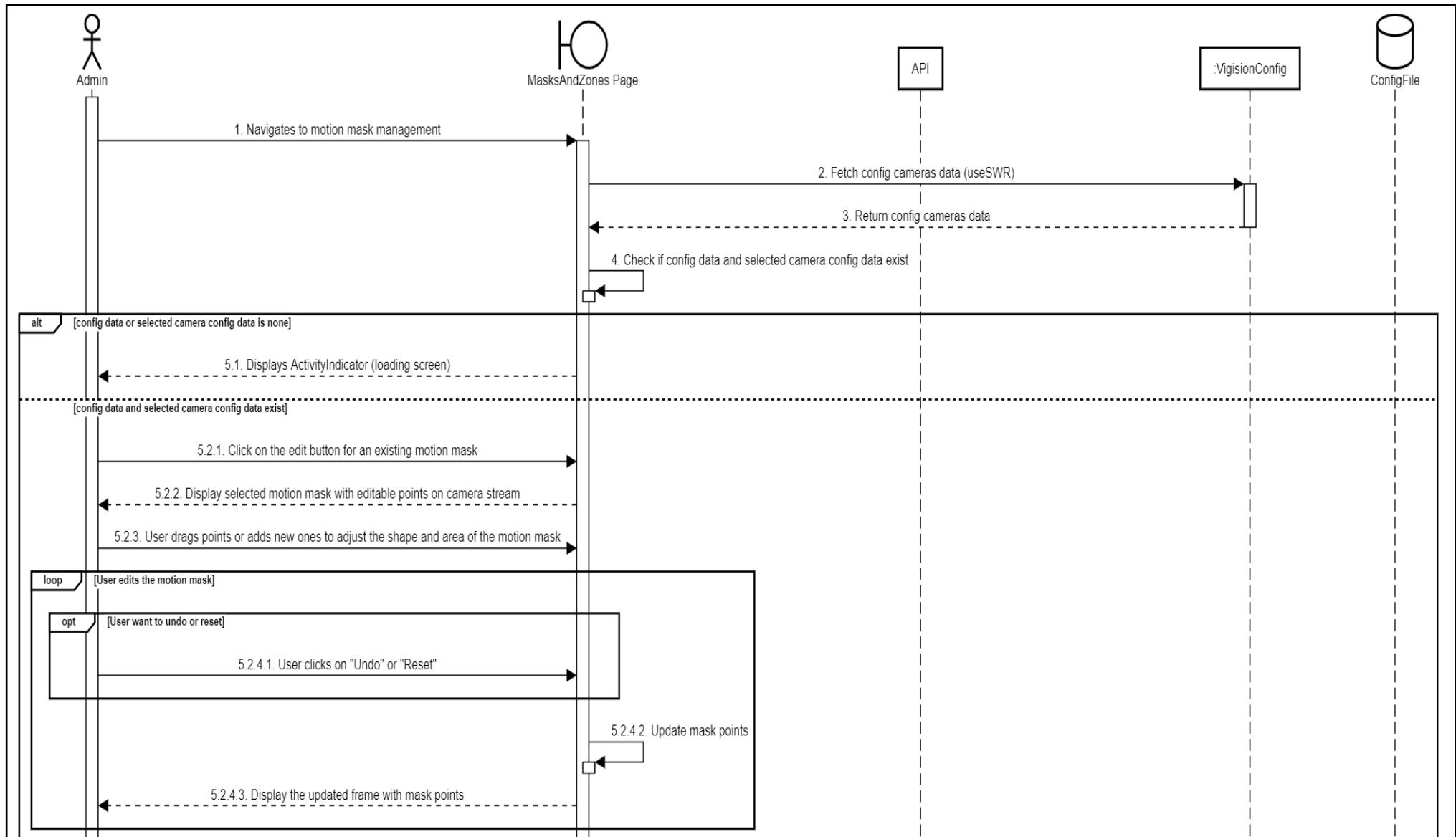
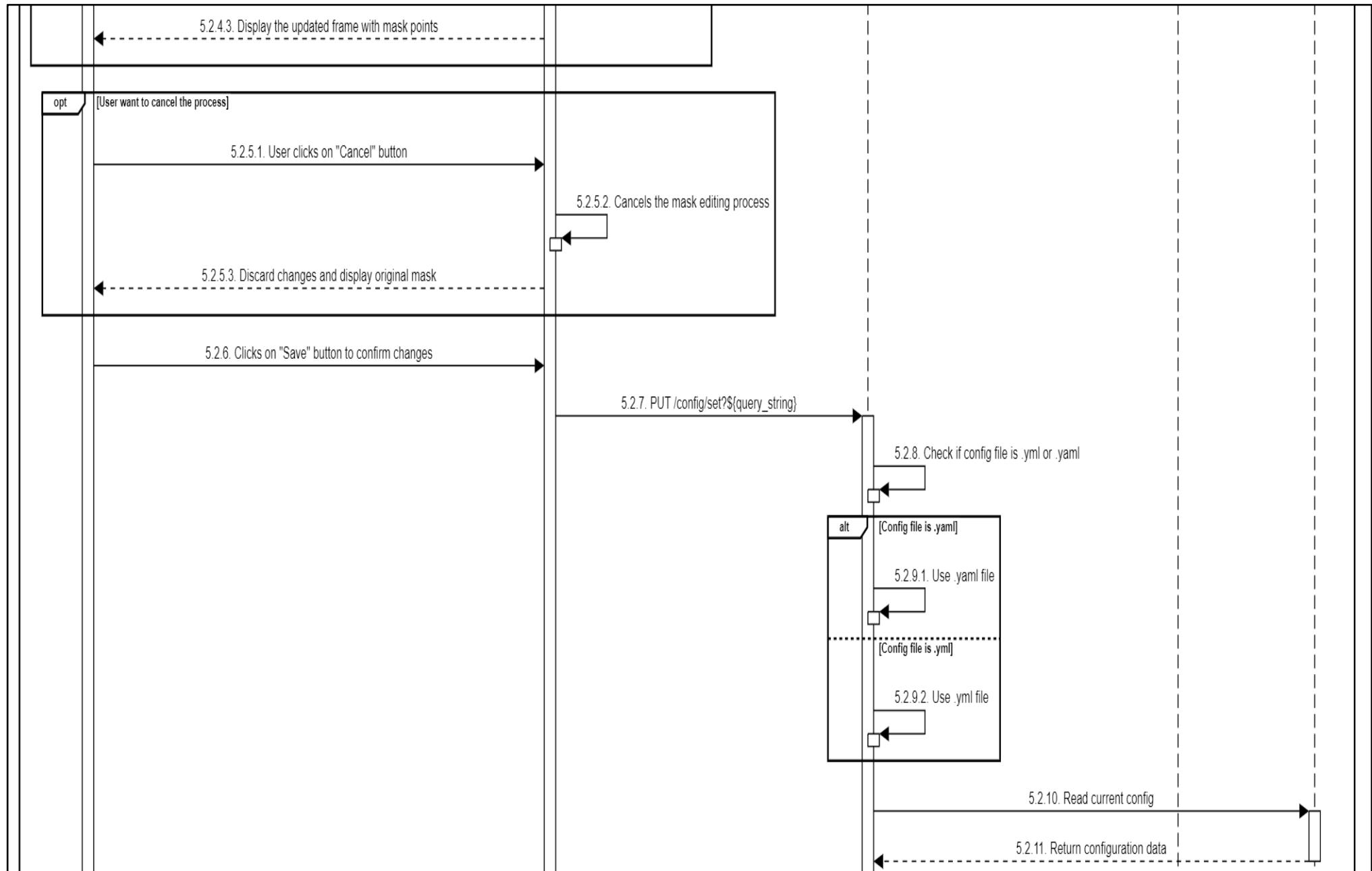


Figure 204. Sequence Diagram - Create new motion mask

b. Edit a motion mask





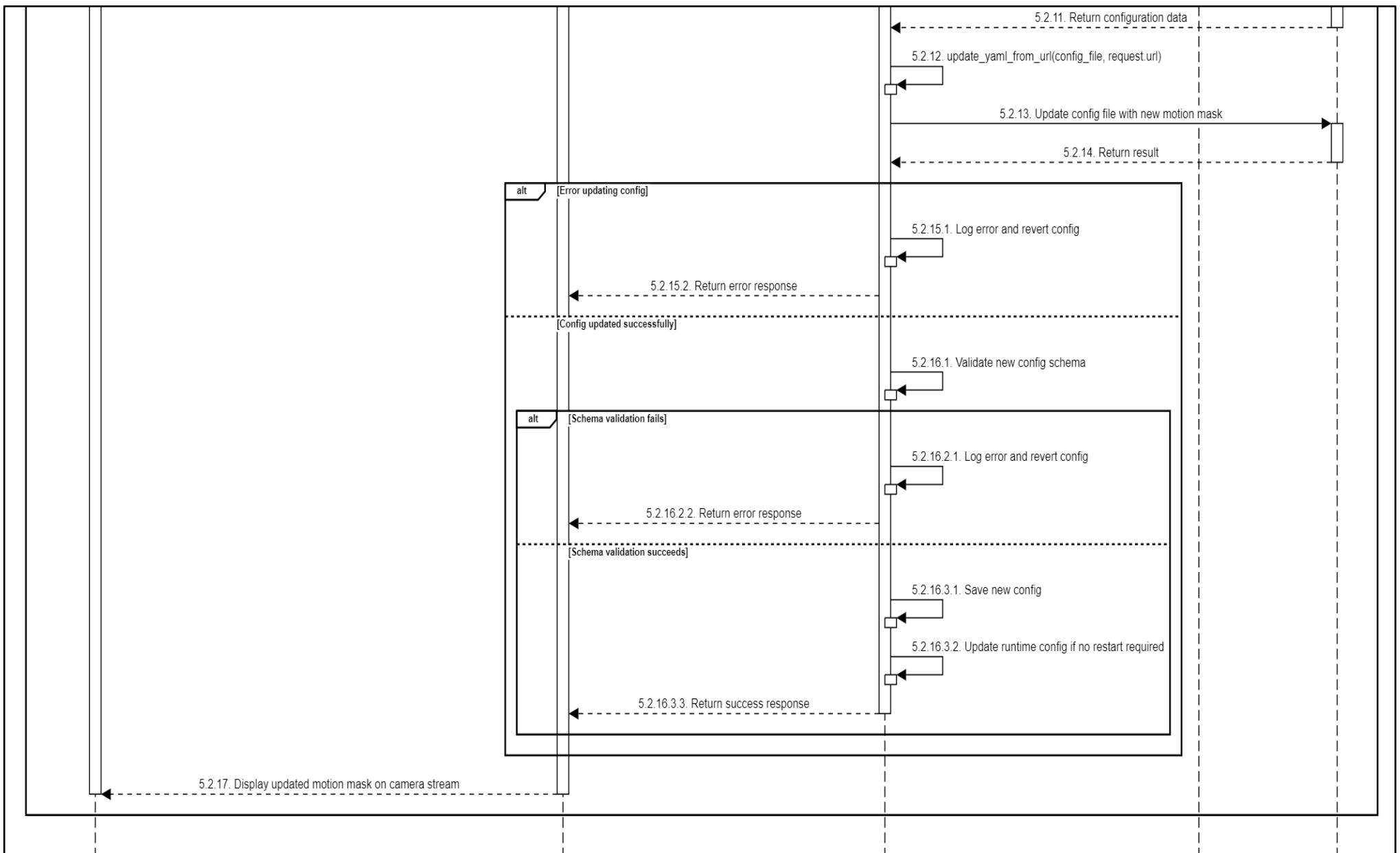
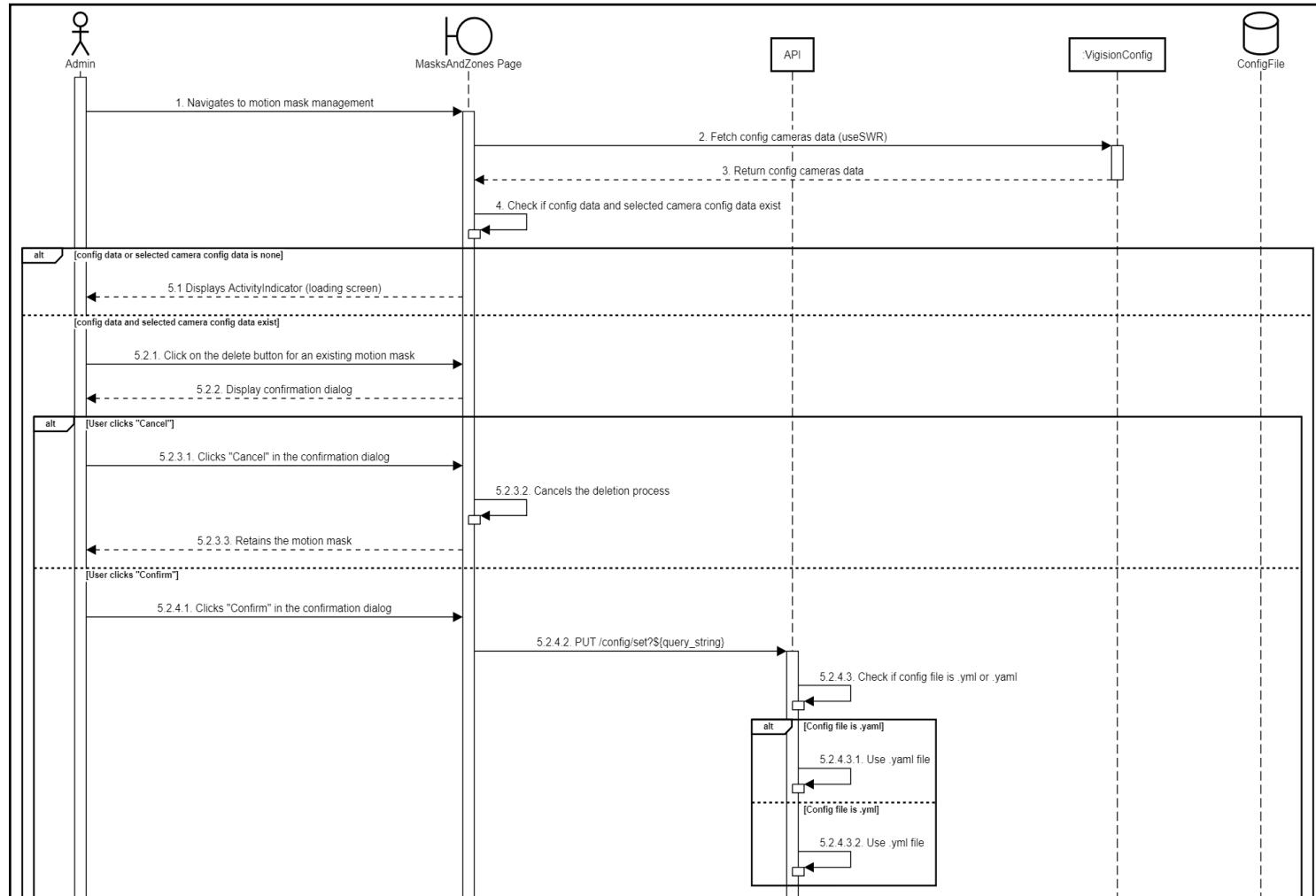


Figure 205. Sequence Diagram - Edit a motion mask

c. Delete a motion mask



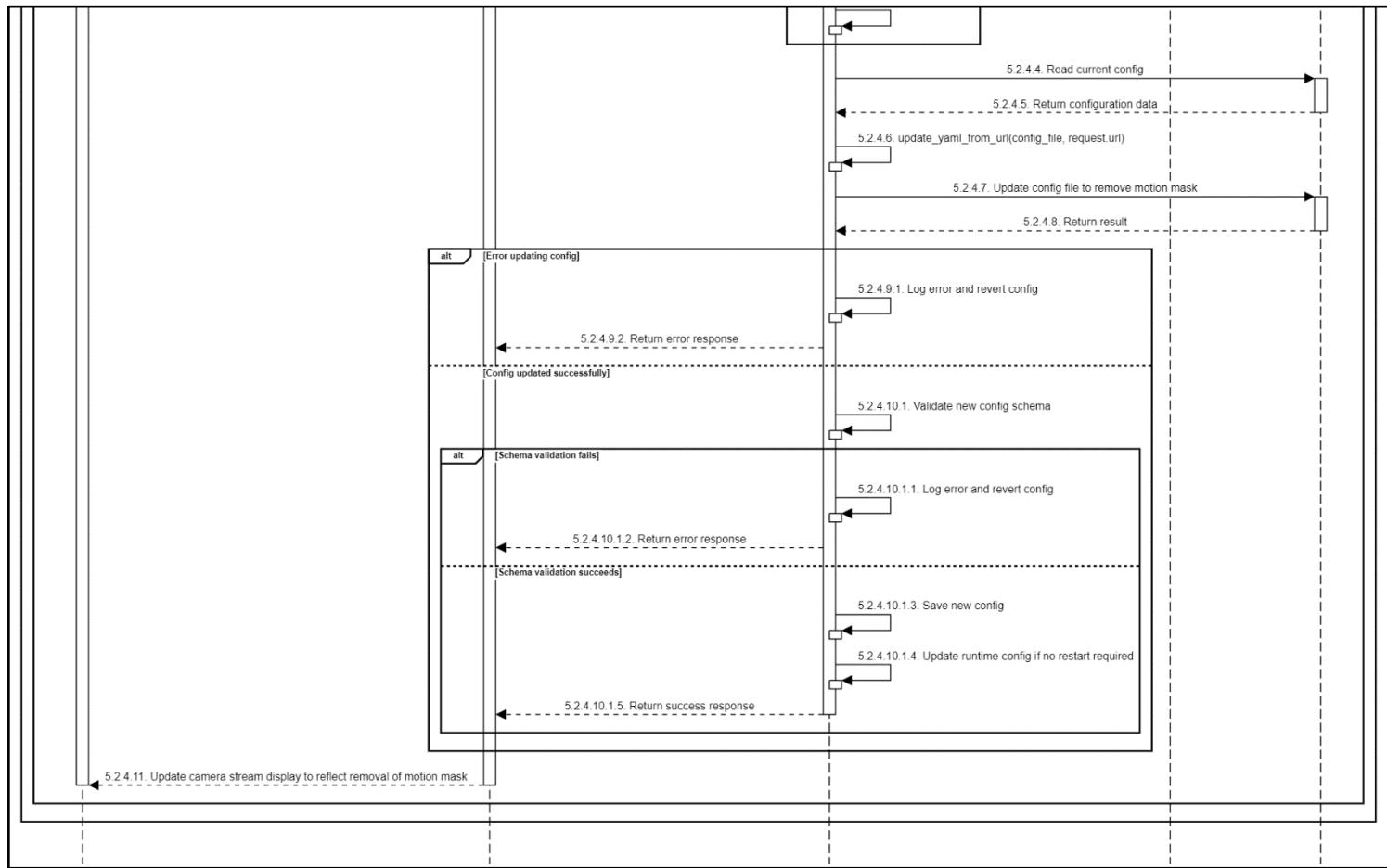


Figure 206. Sequence Diagram - Delete a motion mask

3.17 Manage zones

3.17.1 Class Diagram

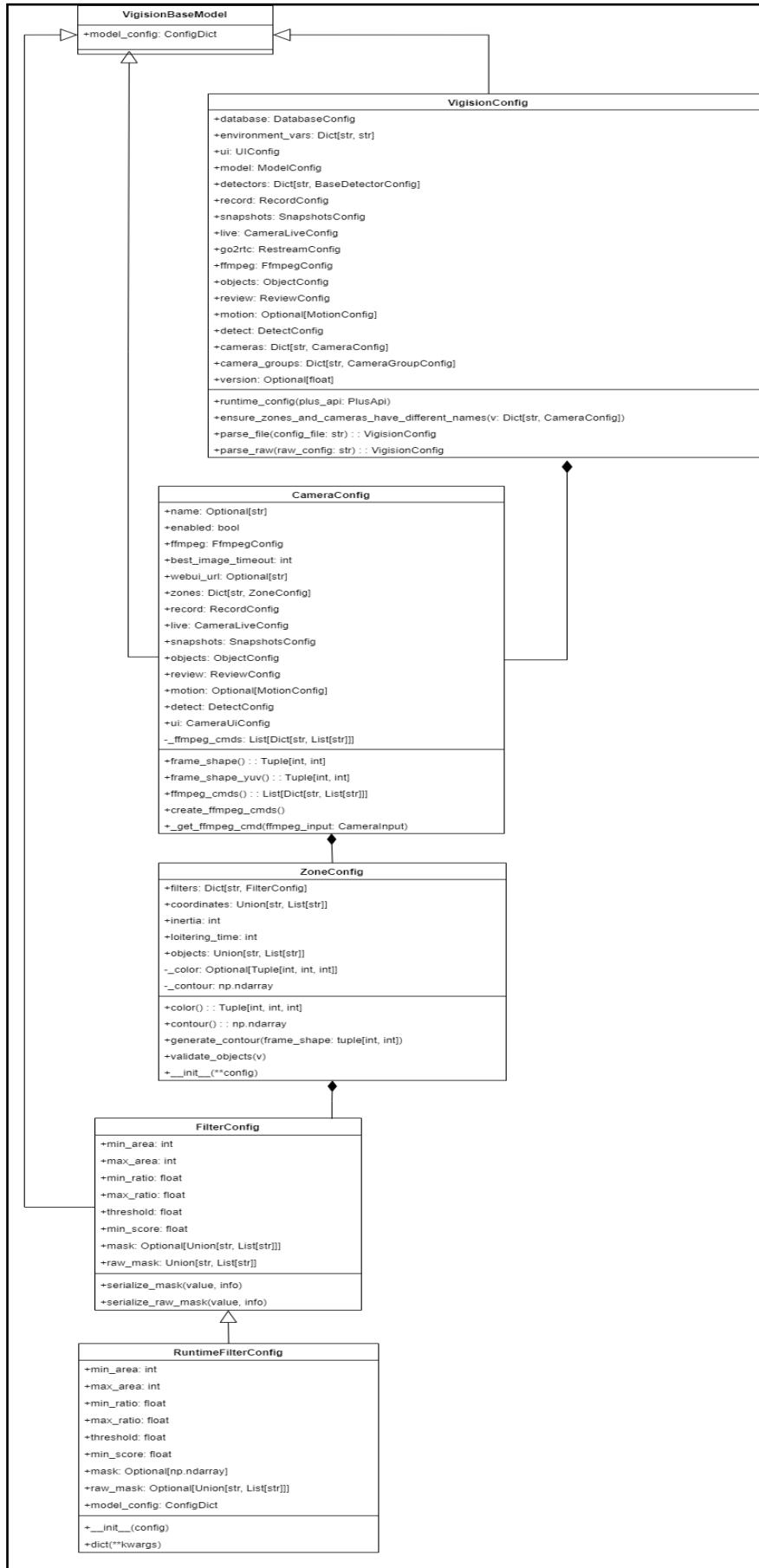
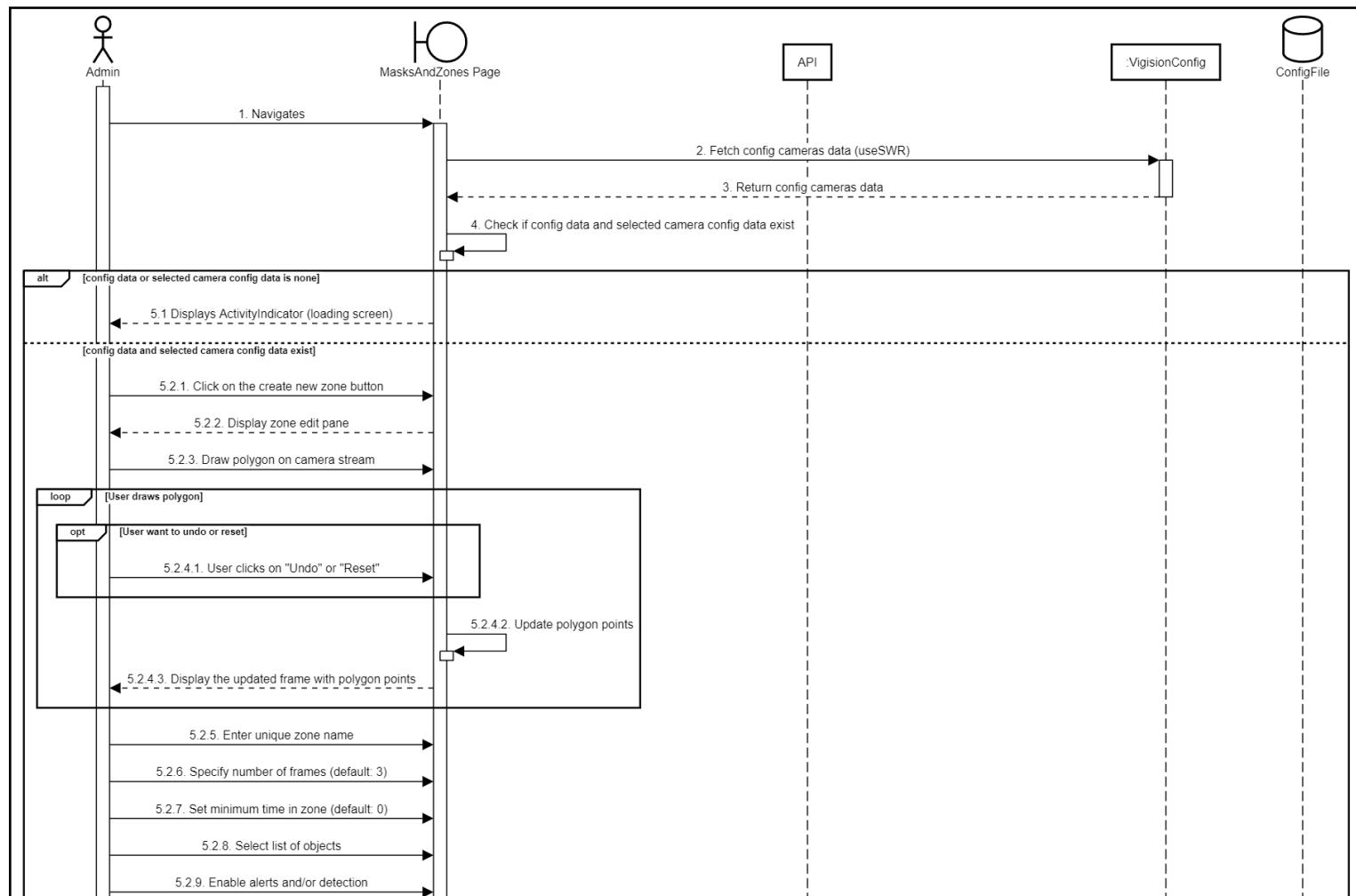
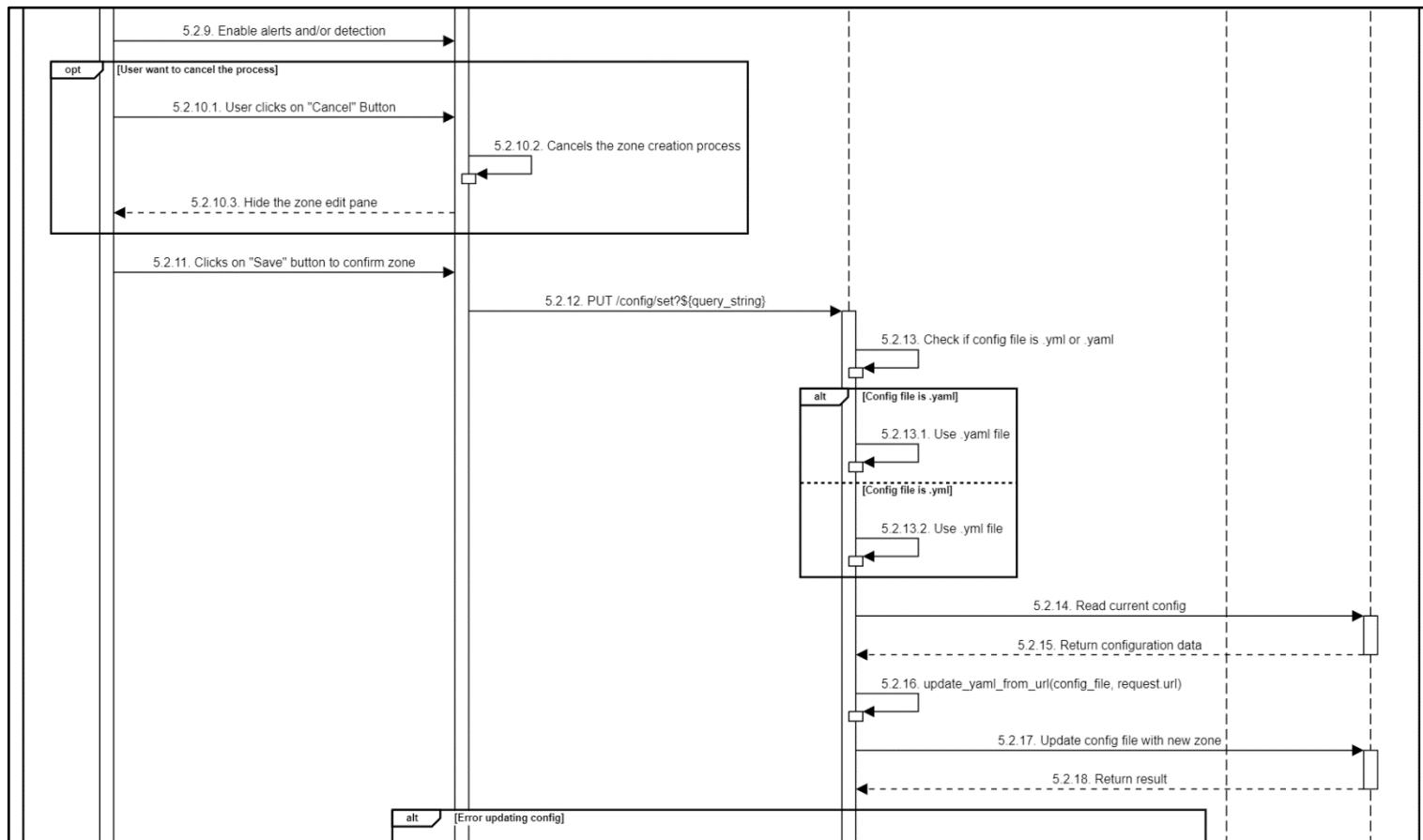


Figure 207. Class Diagram - Manage zones

3.17.2 Sequence Diagram

a. Create new zone





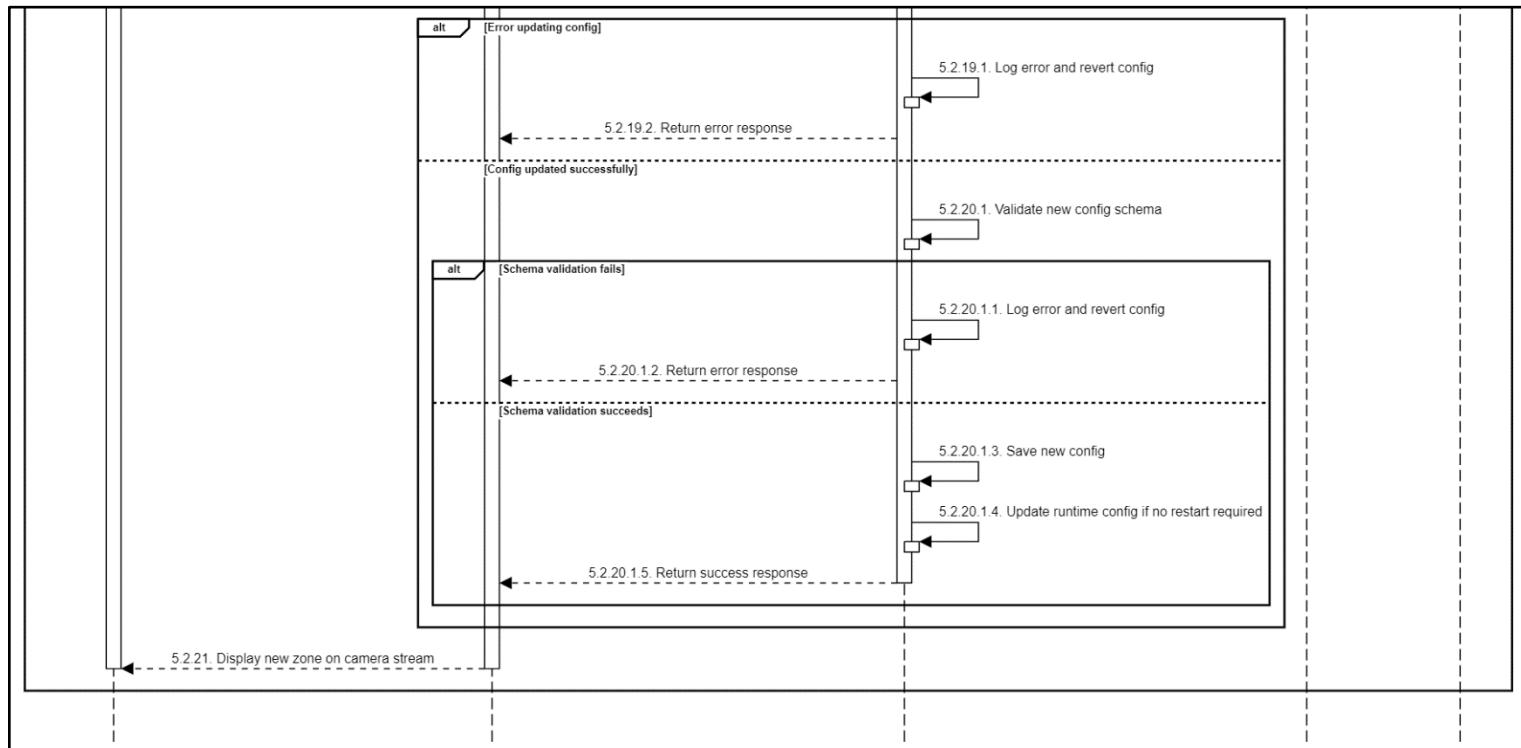
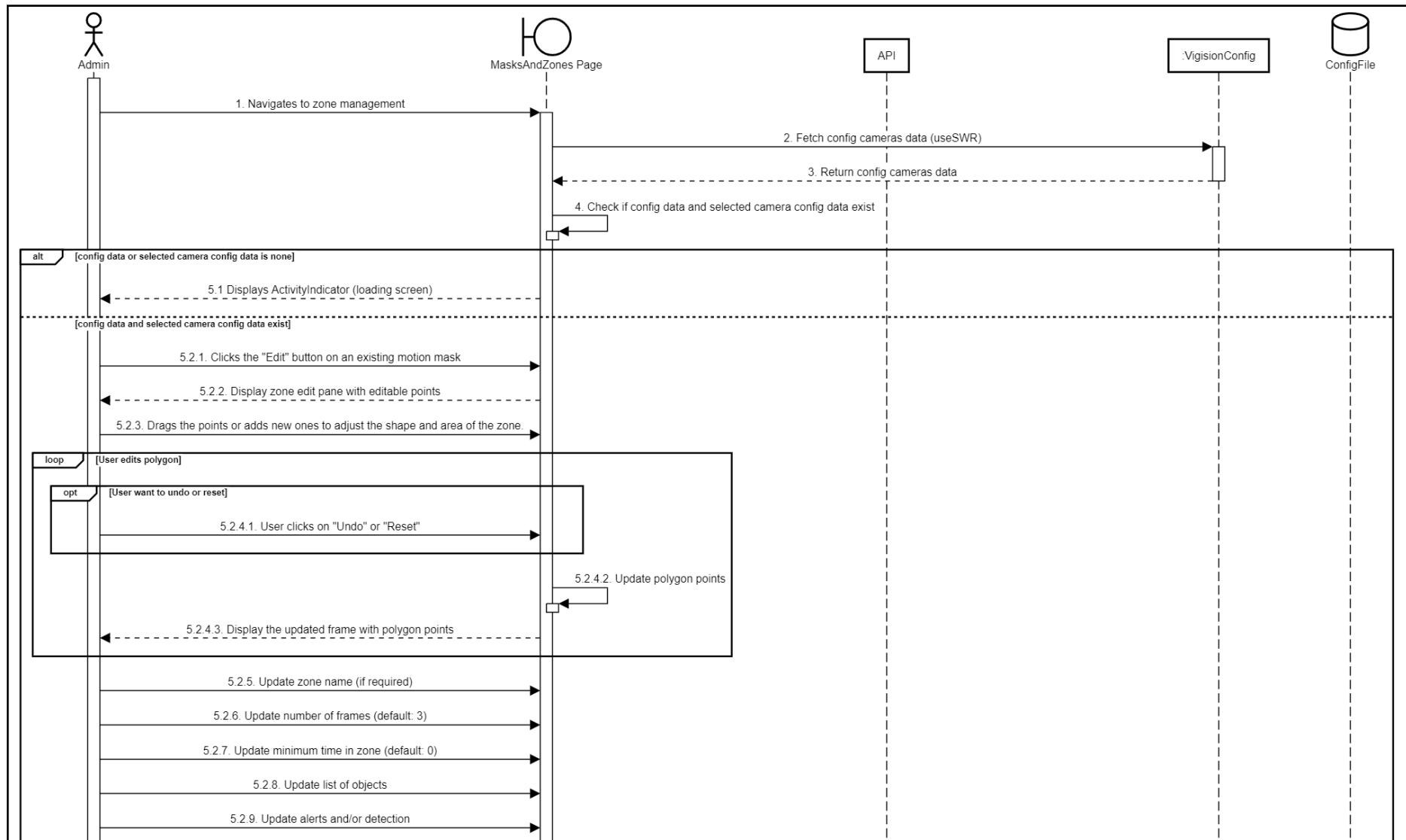
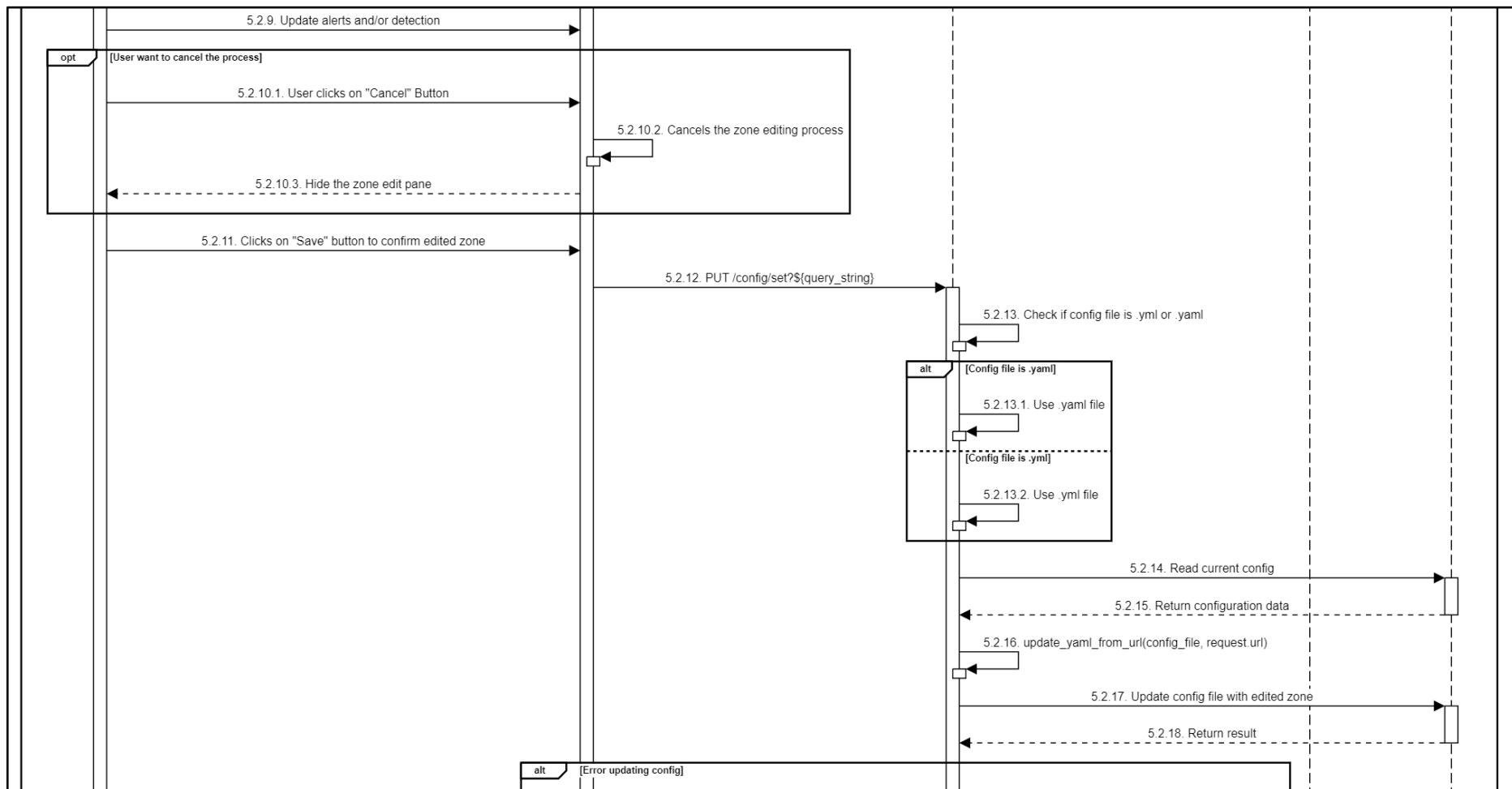


Figure 208. Sequence Diagram - Create new zone

b. Edit a zone





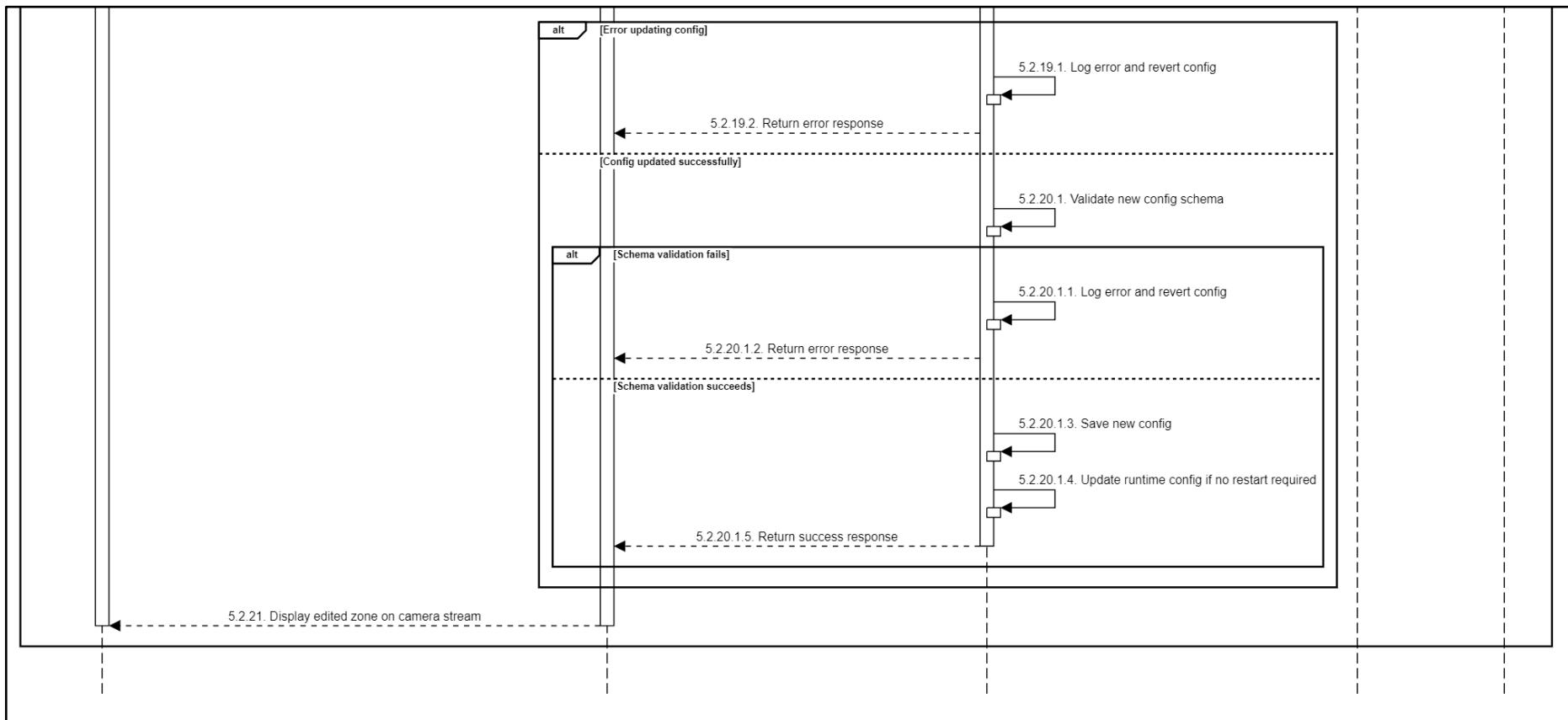
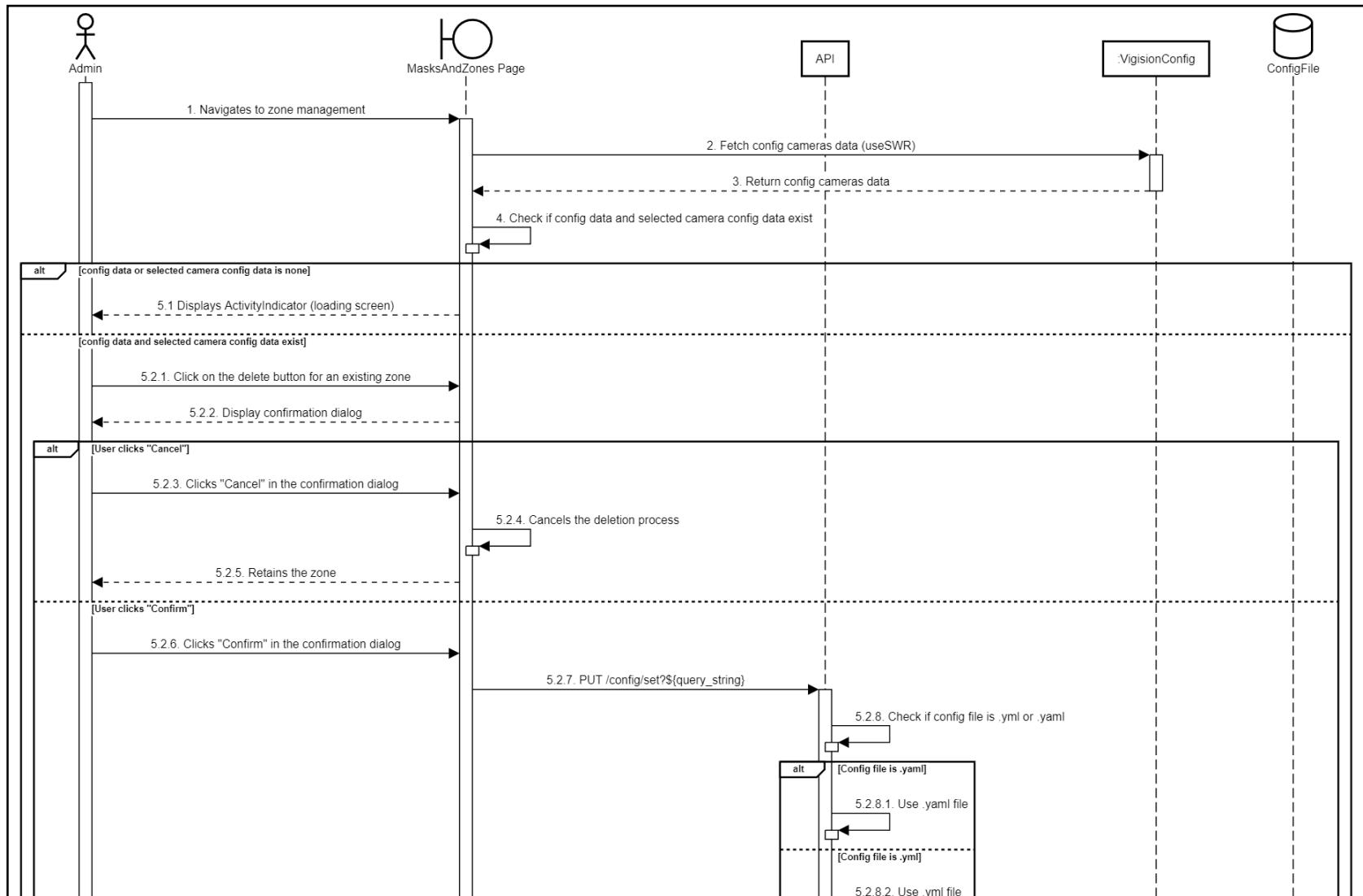


Figure 209. Sequence Diagram - Edit a zone

c. Delete a zone



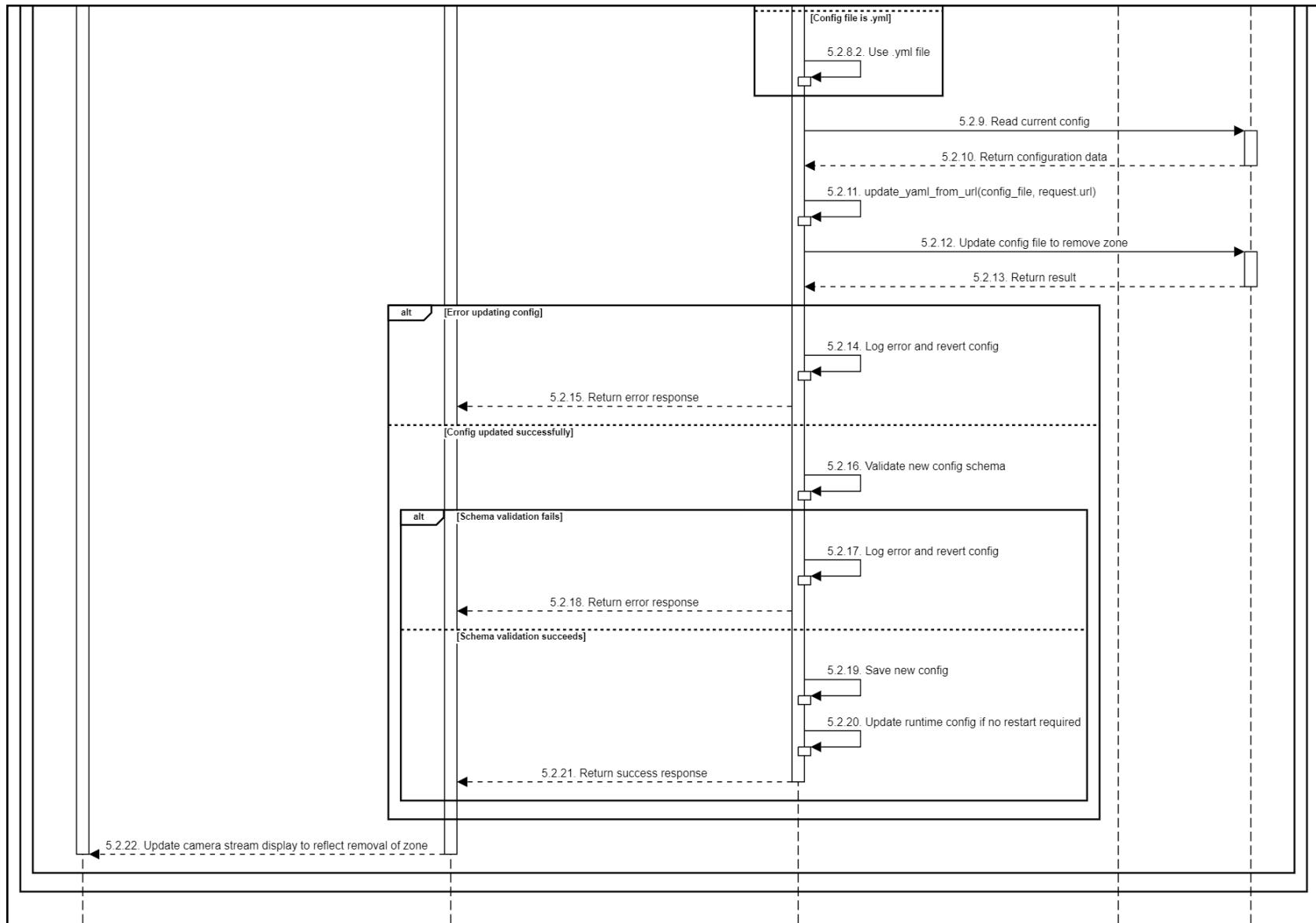


Figure 210. Sequence Diagram - Delete a zone

3.18 Manage object masks

3.18.1 Class Diagram

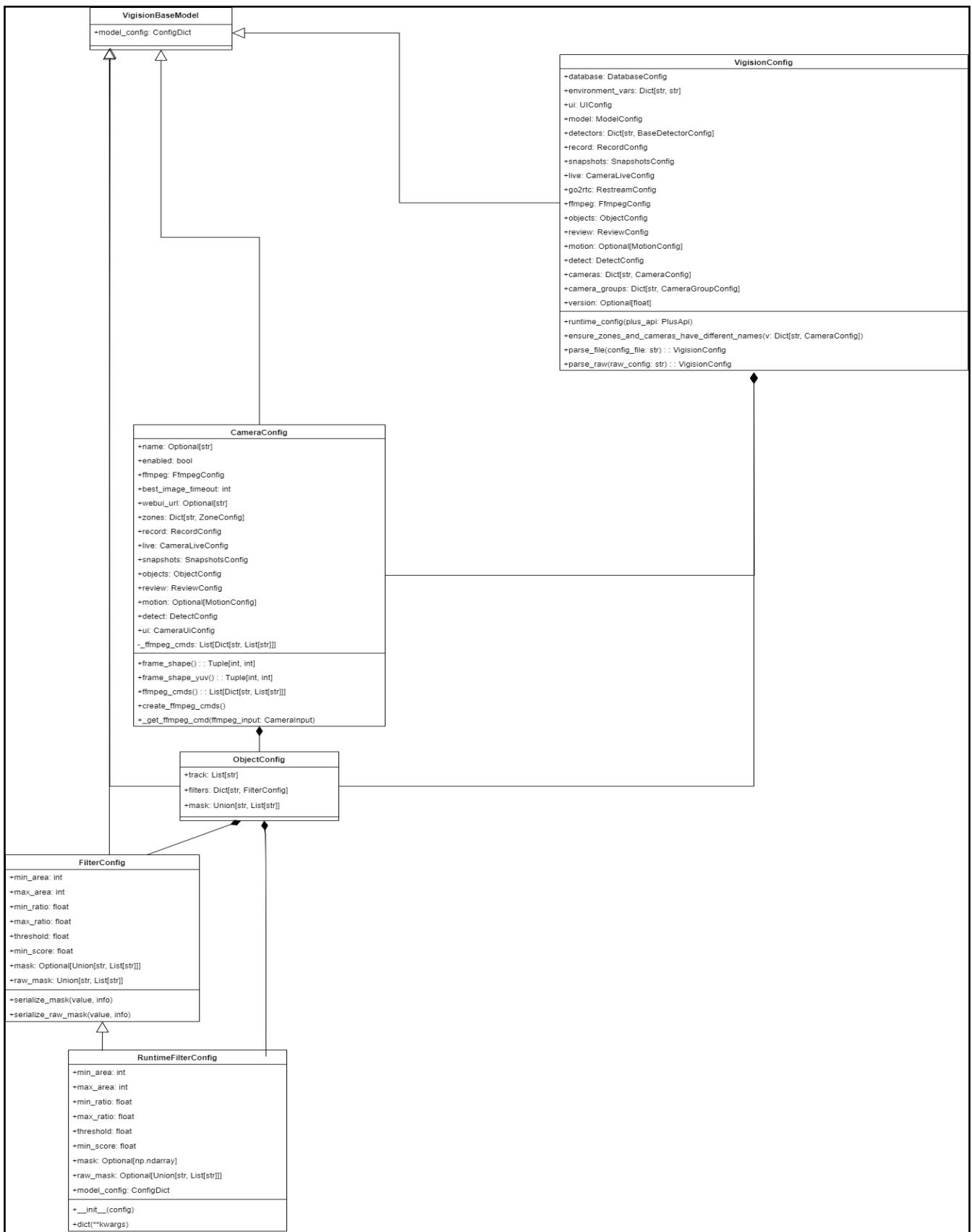
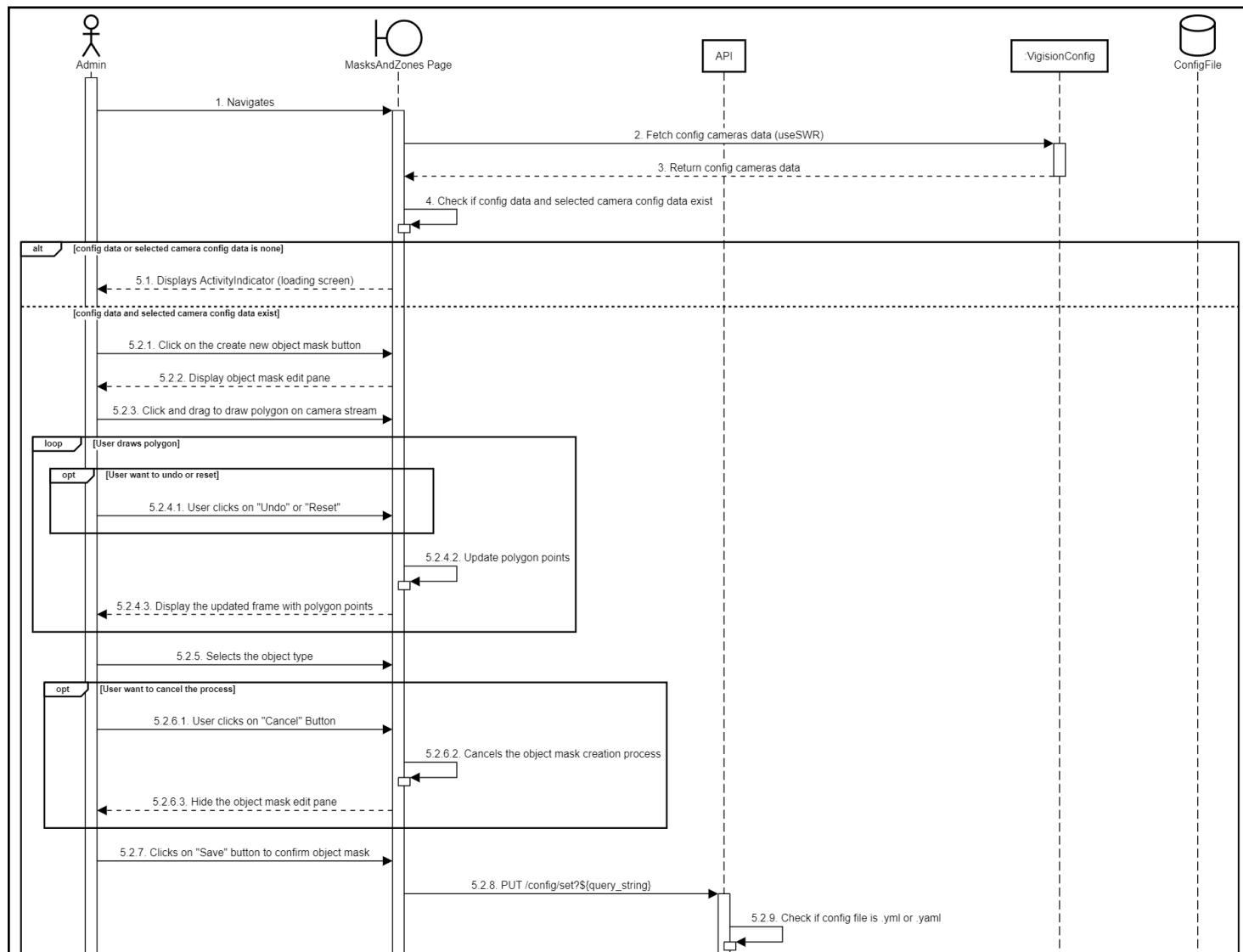


Figure 211. Class Diagram - Manage object mask

3.18.2 Sequence Diagram

a. Create new object mask



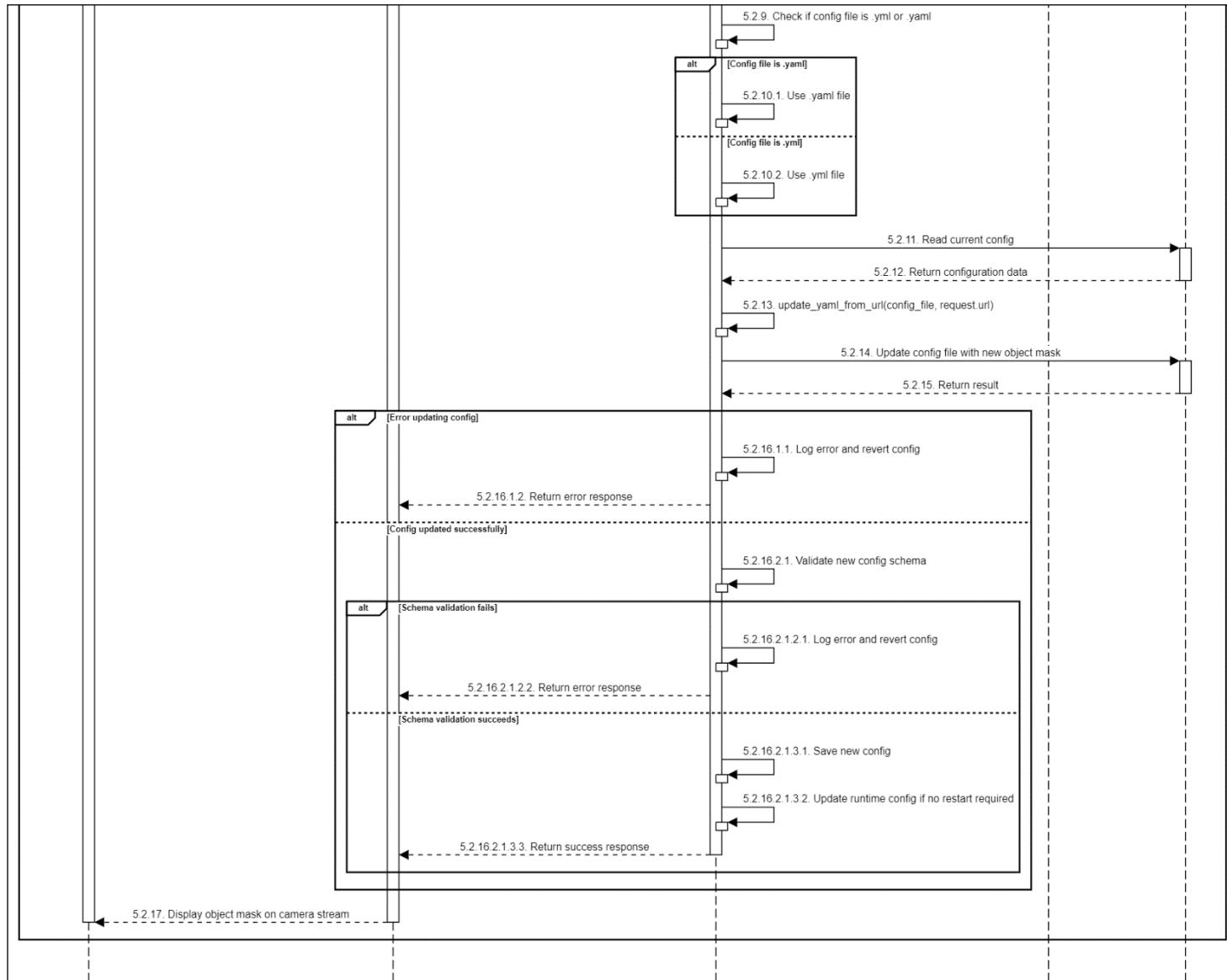
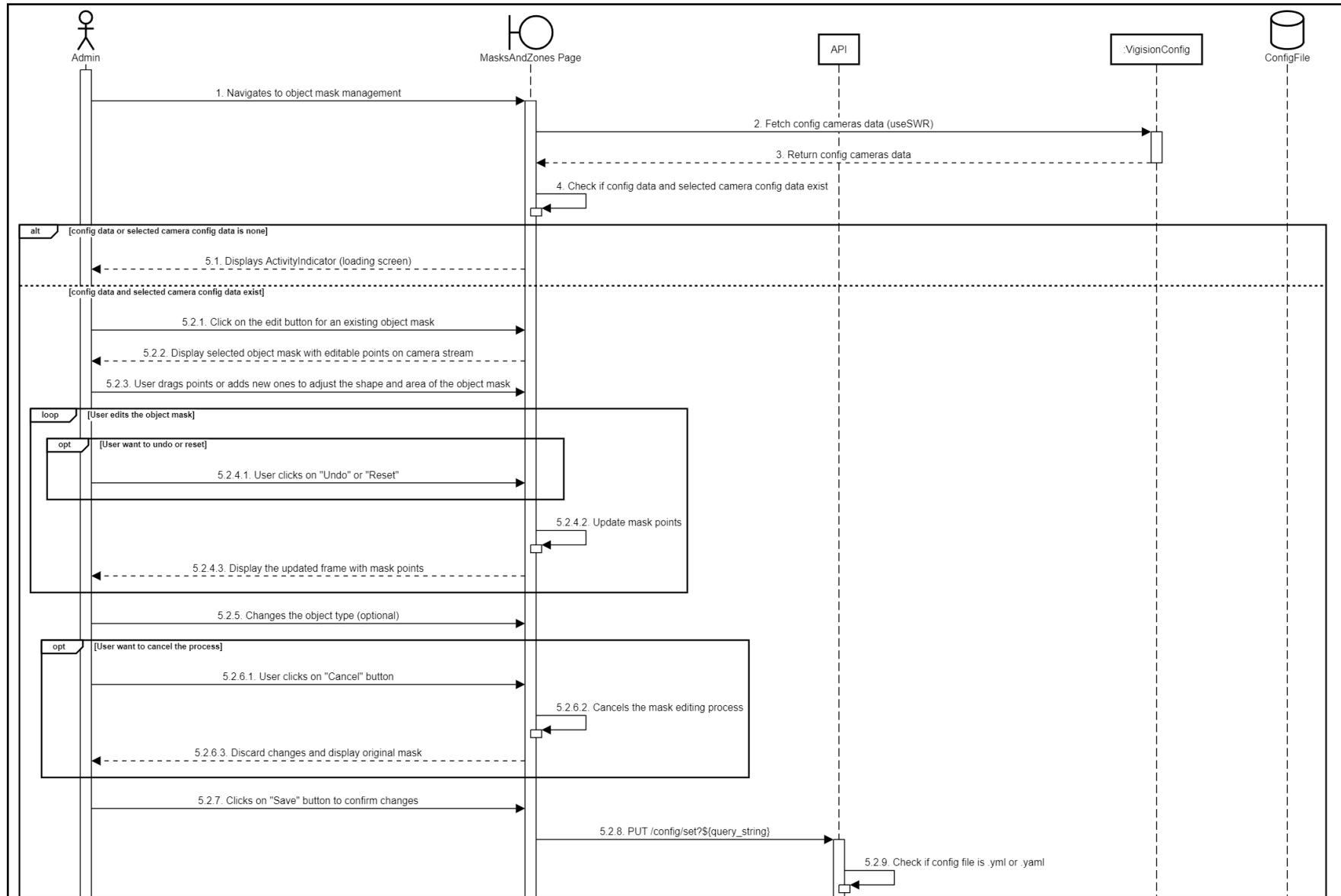


Figure 212. Sequence Diagram - Create new object mask

b. Edit an object mask



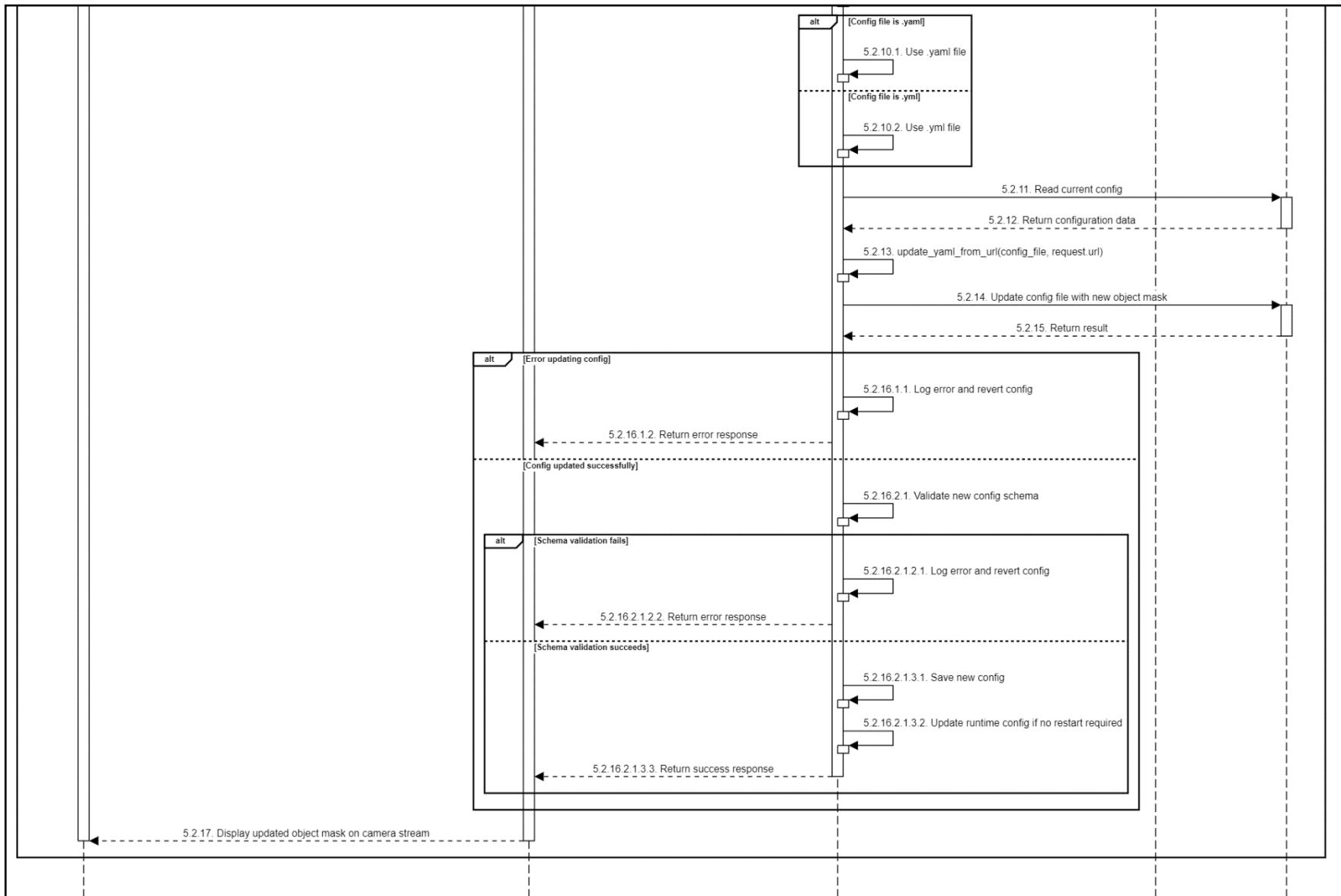
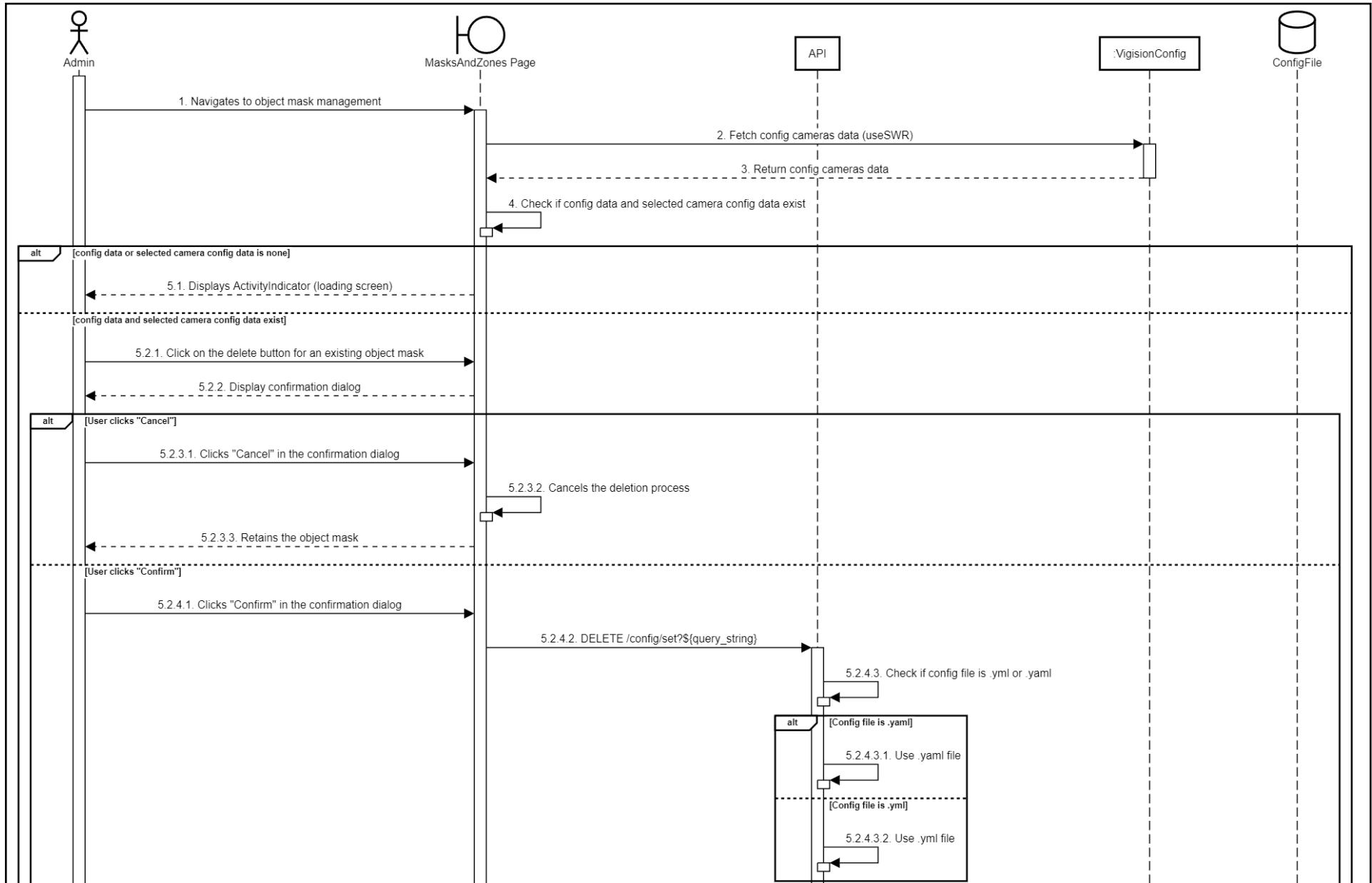


Figure 213. Sequence Diagram - Edit an object mask

c. Delete an object mask



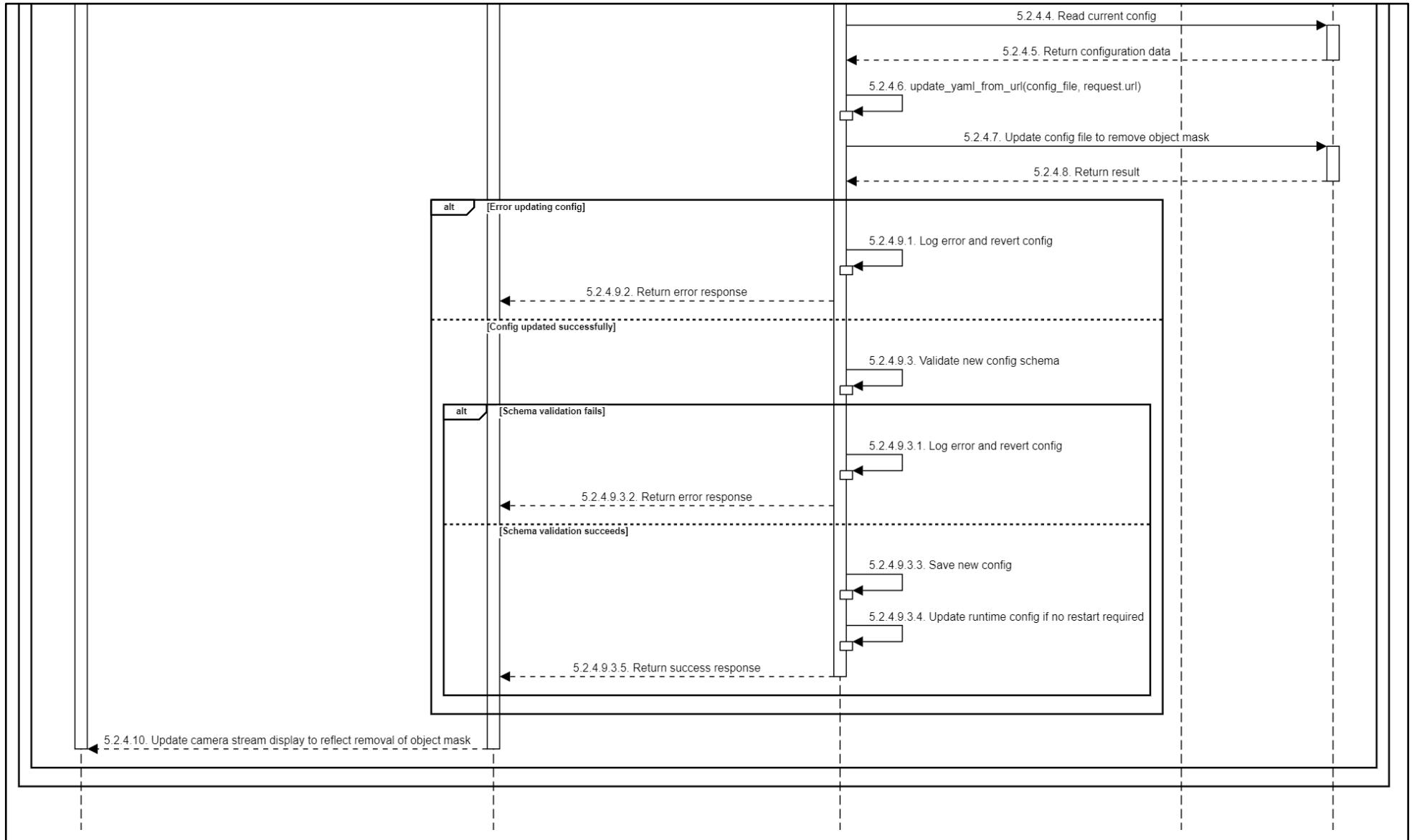


Figure 214. Sequence Diagram - Delete an object mask

3.19 Fine-tune motion detection

3.19.1 Class Diagram

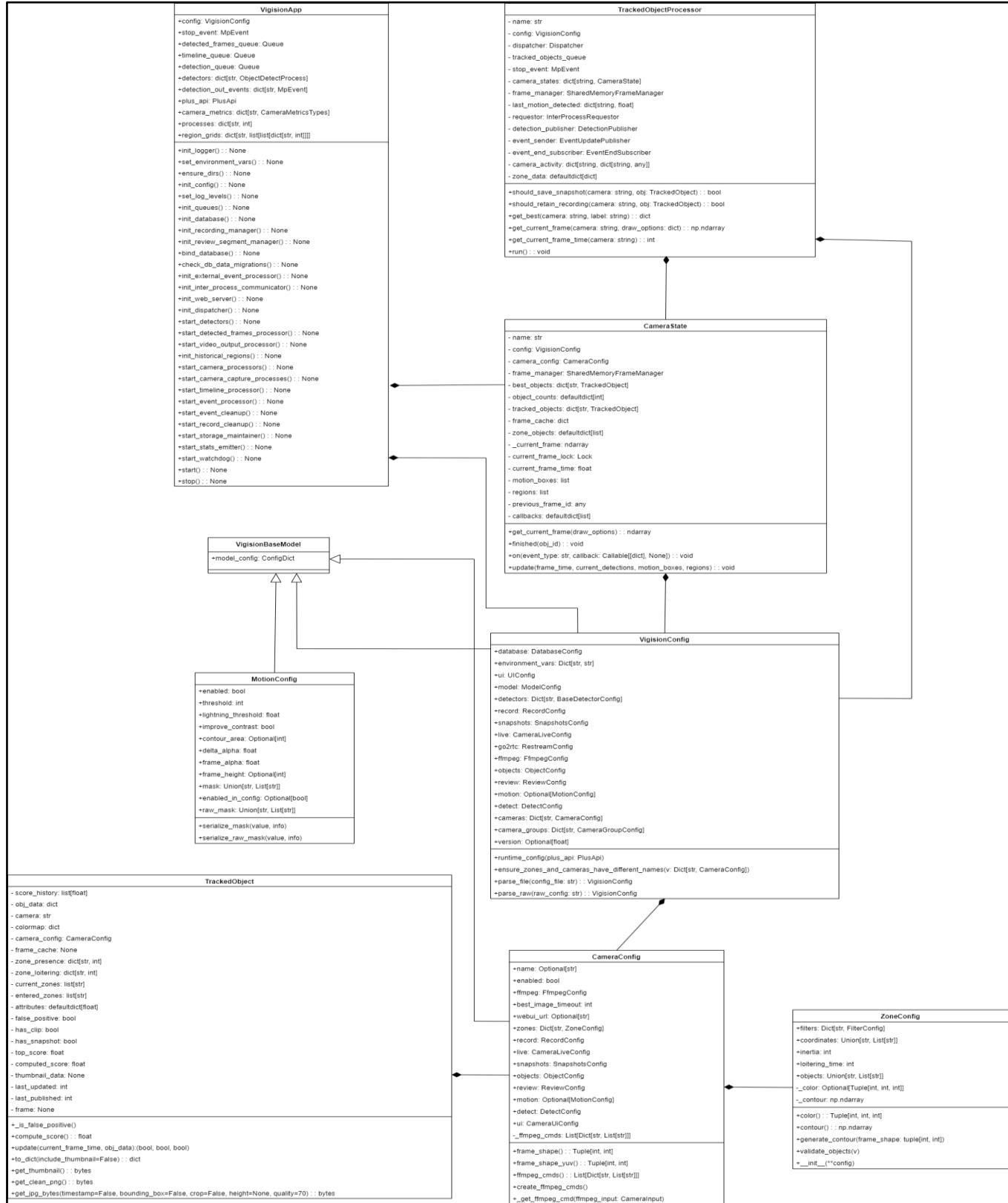
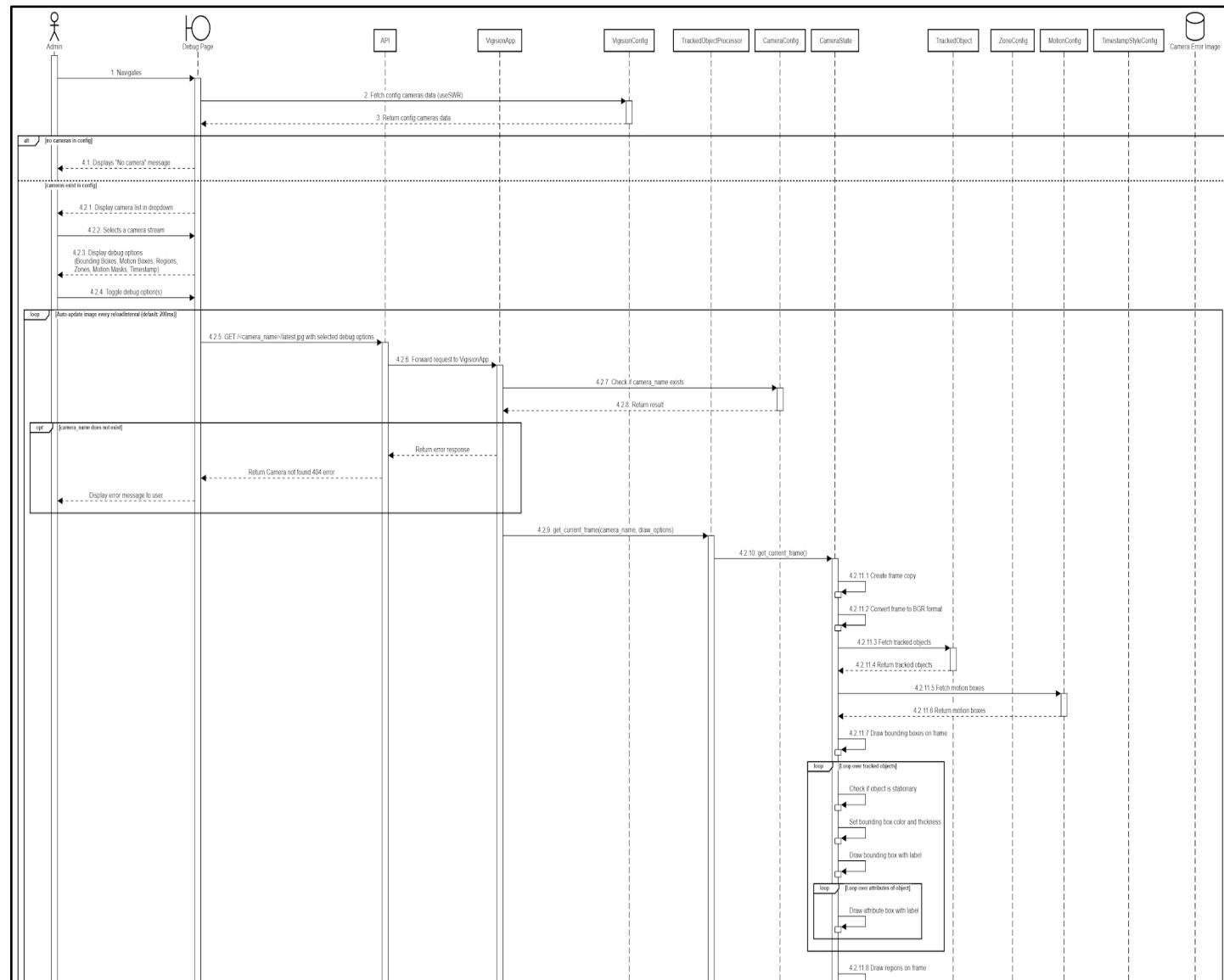


Figure 215. Class Diagram - Fine-tune motion detection

3.19.2 Sequence Diagram



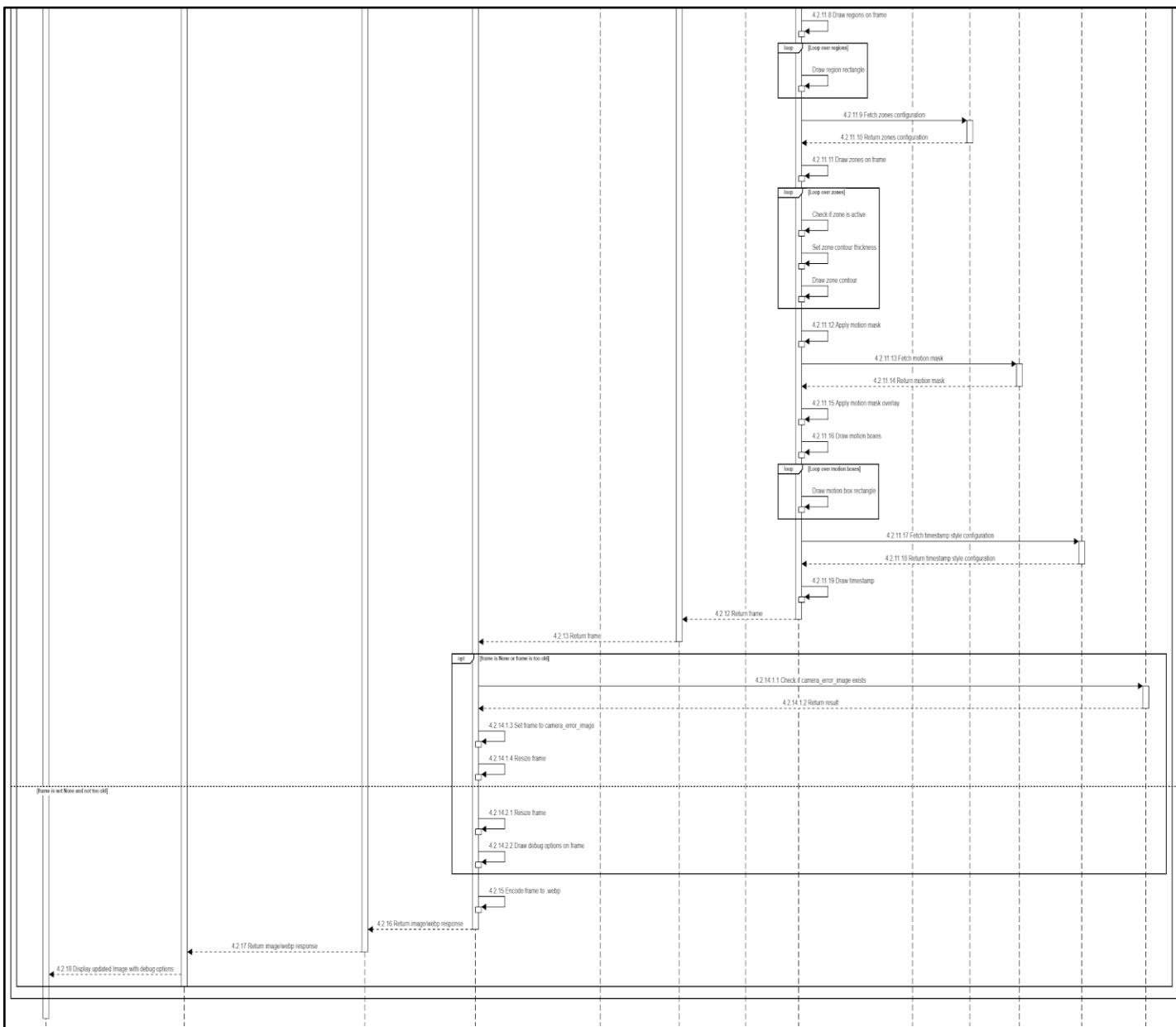


Figure 216. Sequence Diagram - Fine-tune motion detection

3.20 Debug

3.21 Class Diagram

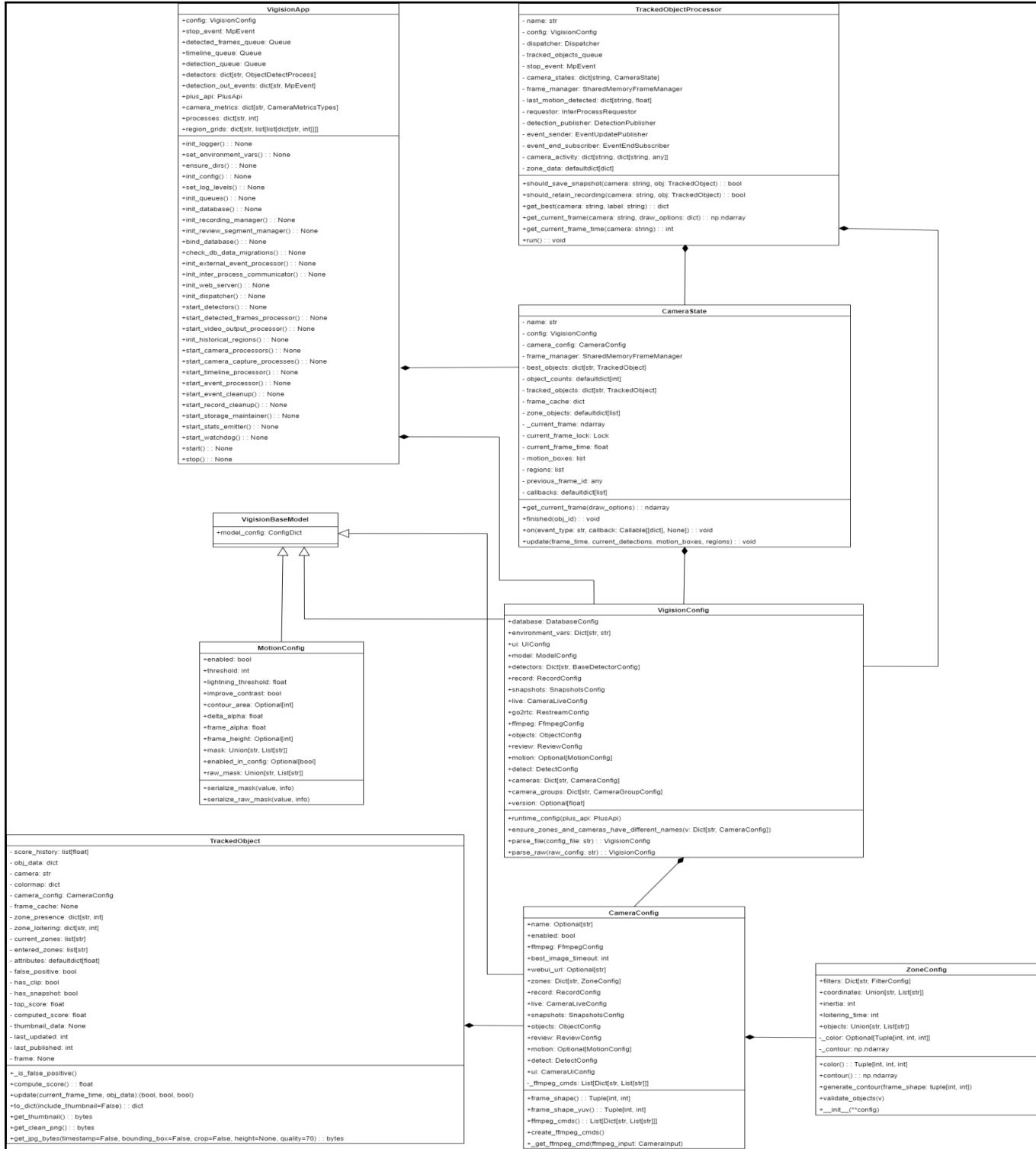
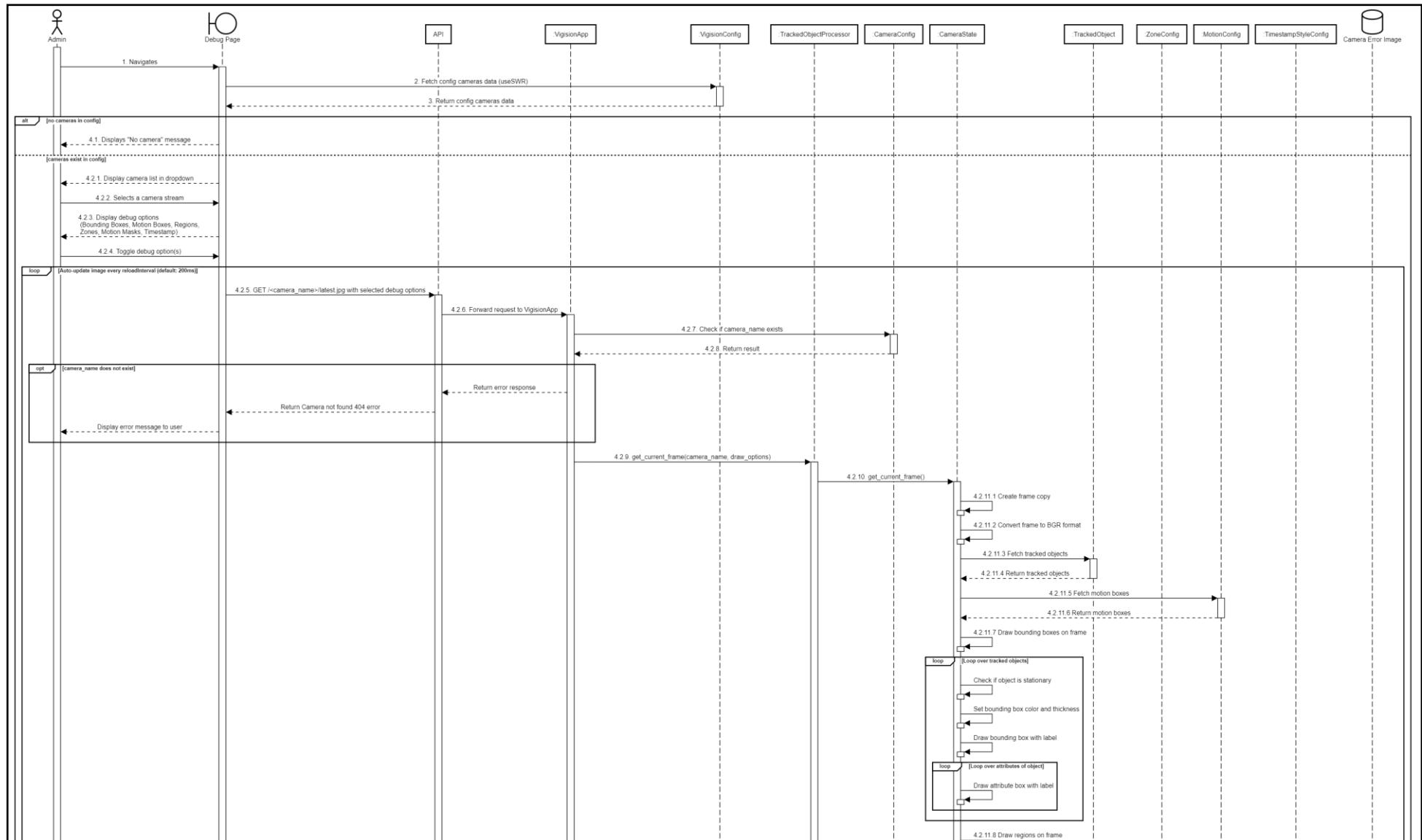


Figure 217. Class Diagram - Debug

3.22 Sequence Diagram



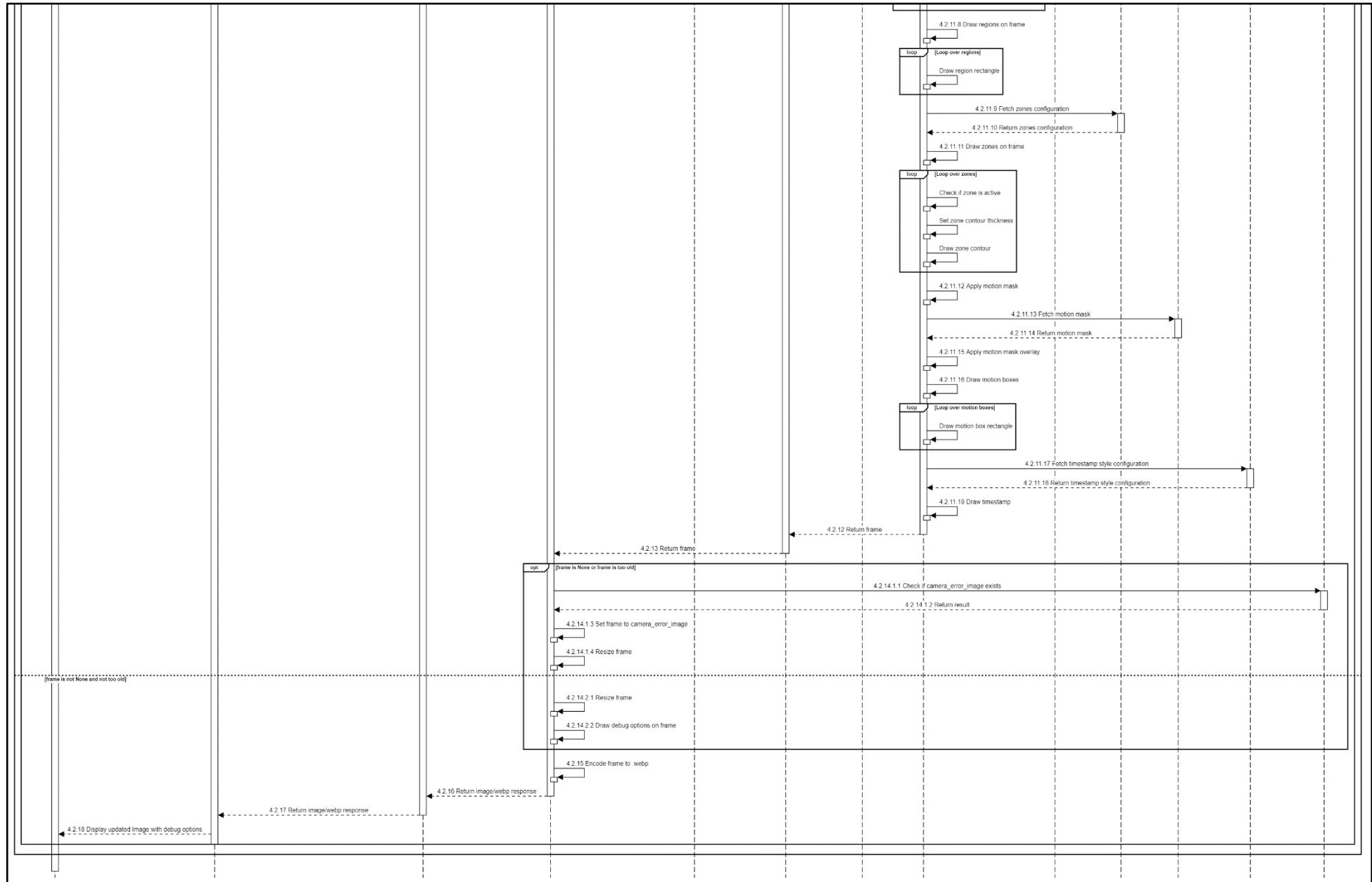


Figure 218. Sequence Diagram - Debug

3.21 View alerts/detections

3.21.1 Class Diagram

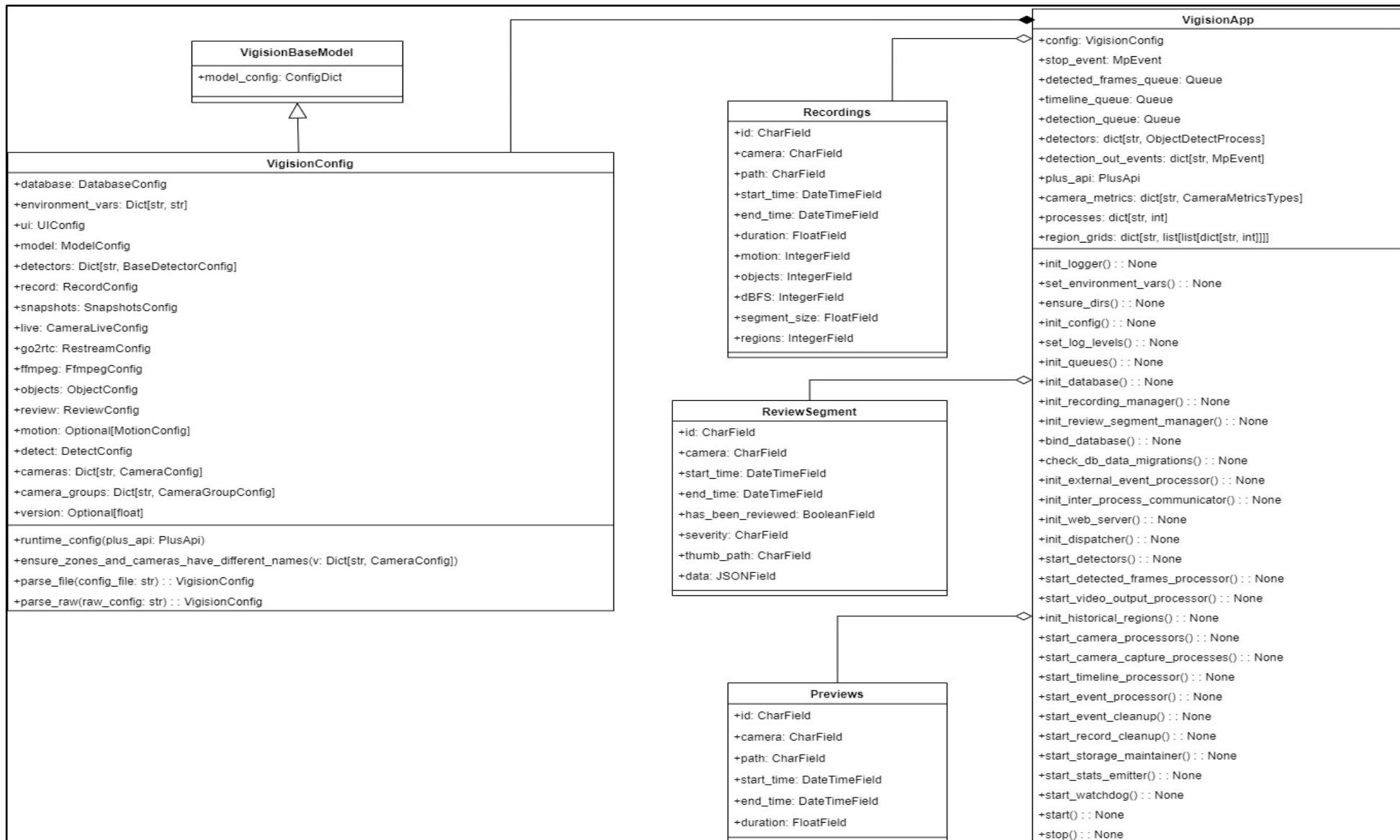
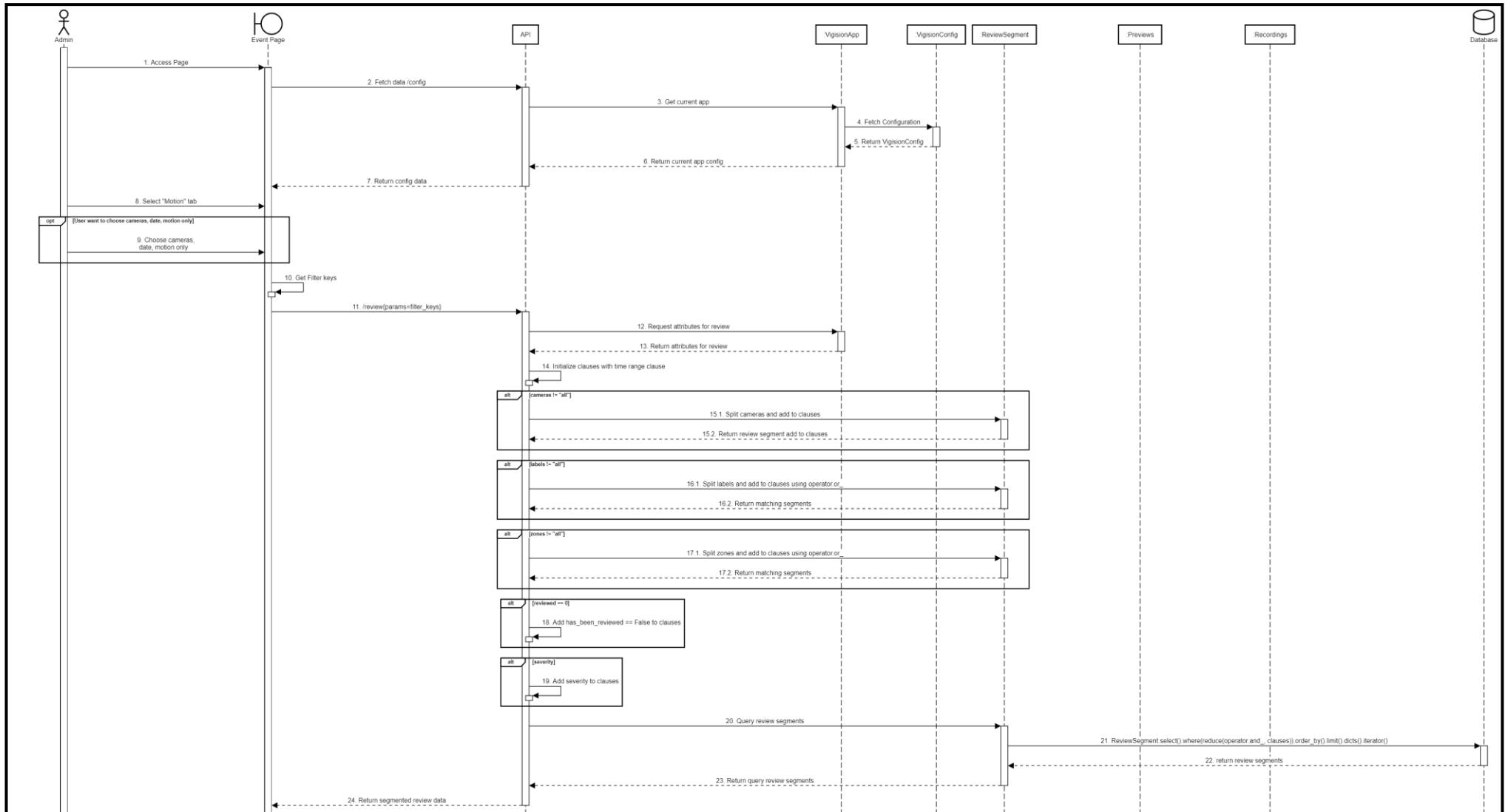
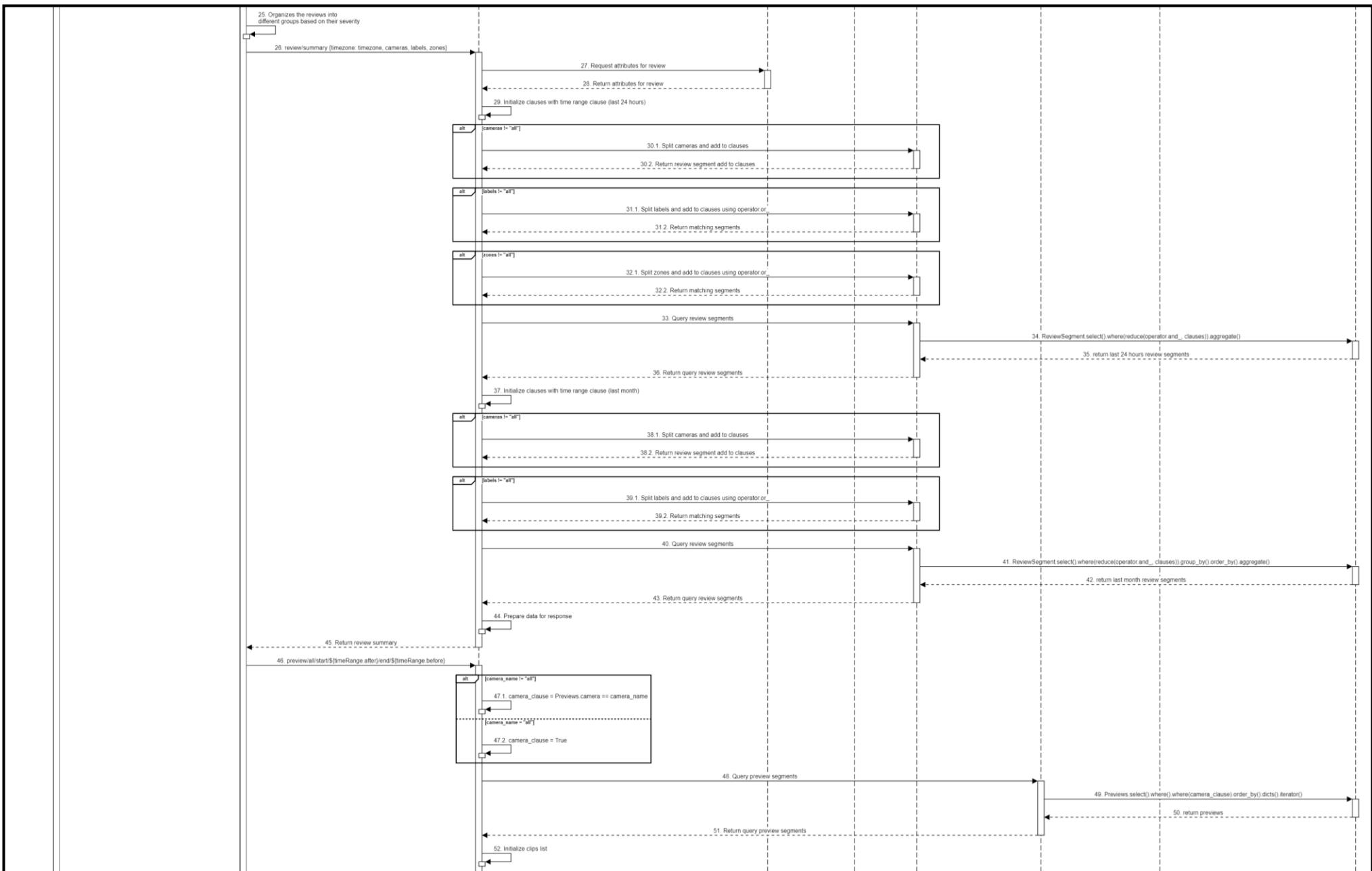


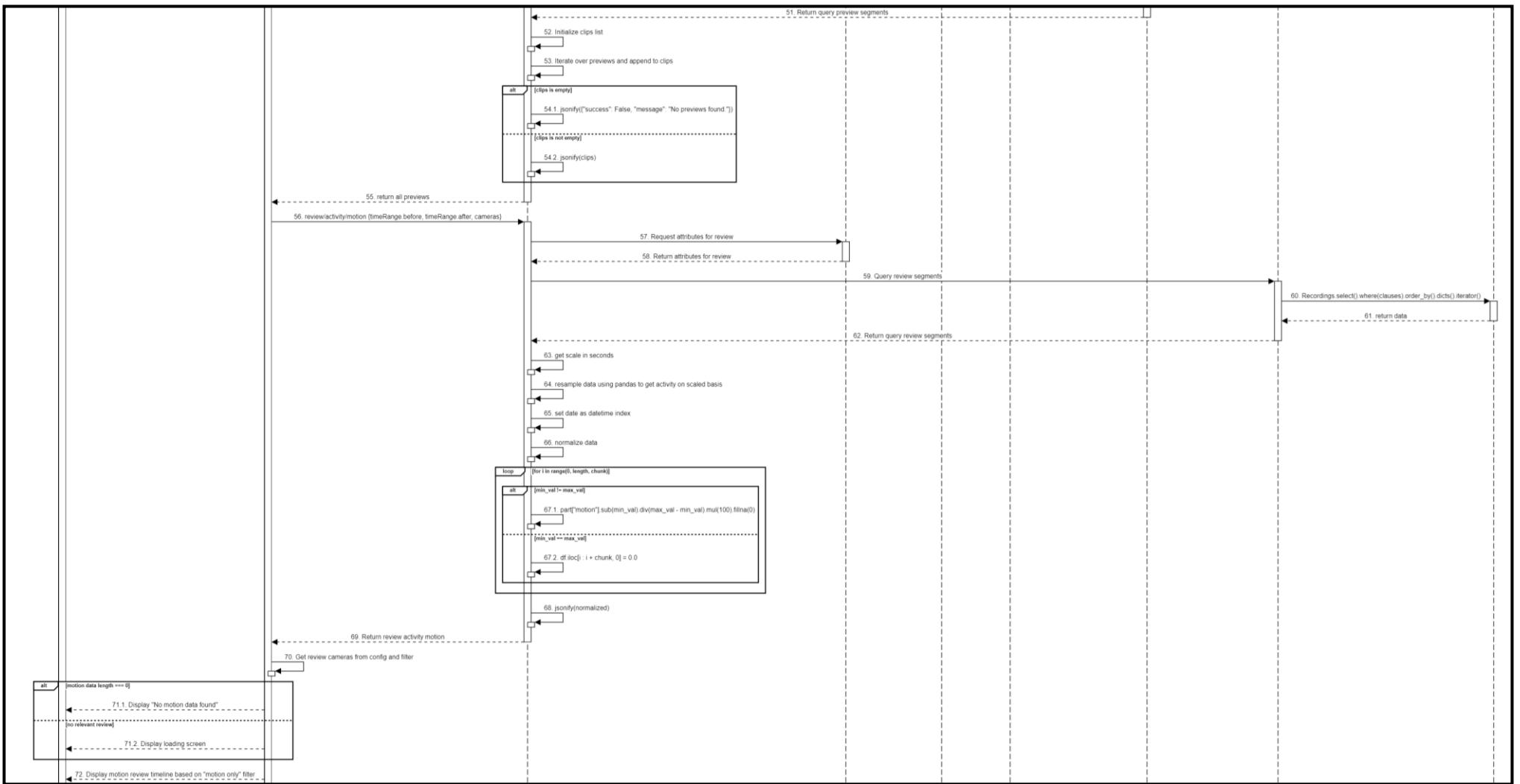
Figure 219. Class Diagram - View alerts/detections

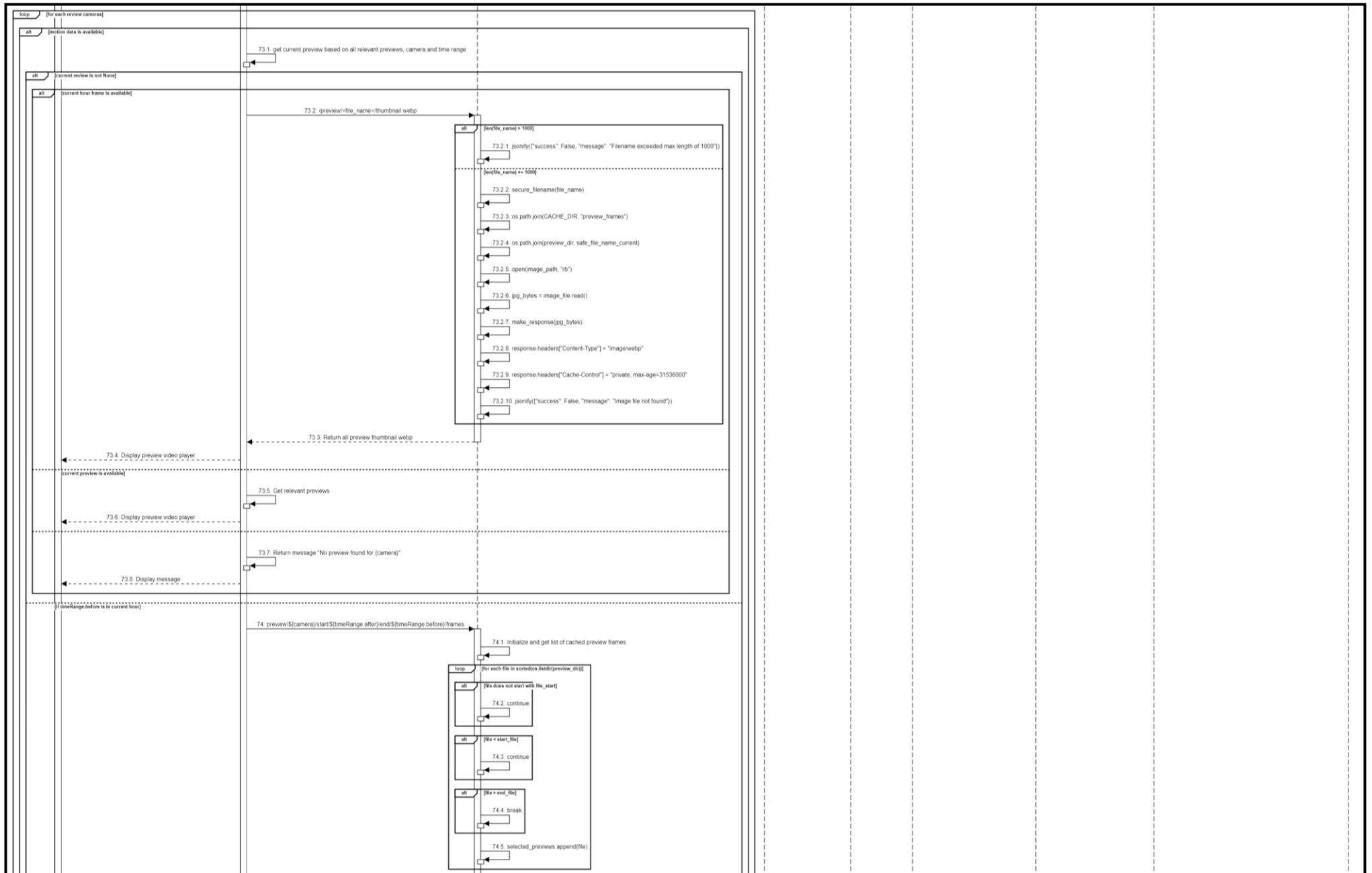
3.21.2 Sequence Diagram

a. View motion events









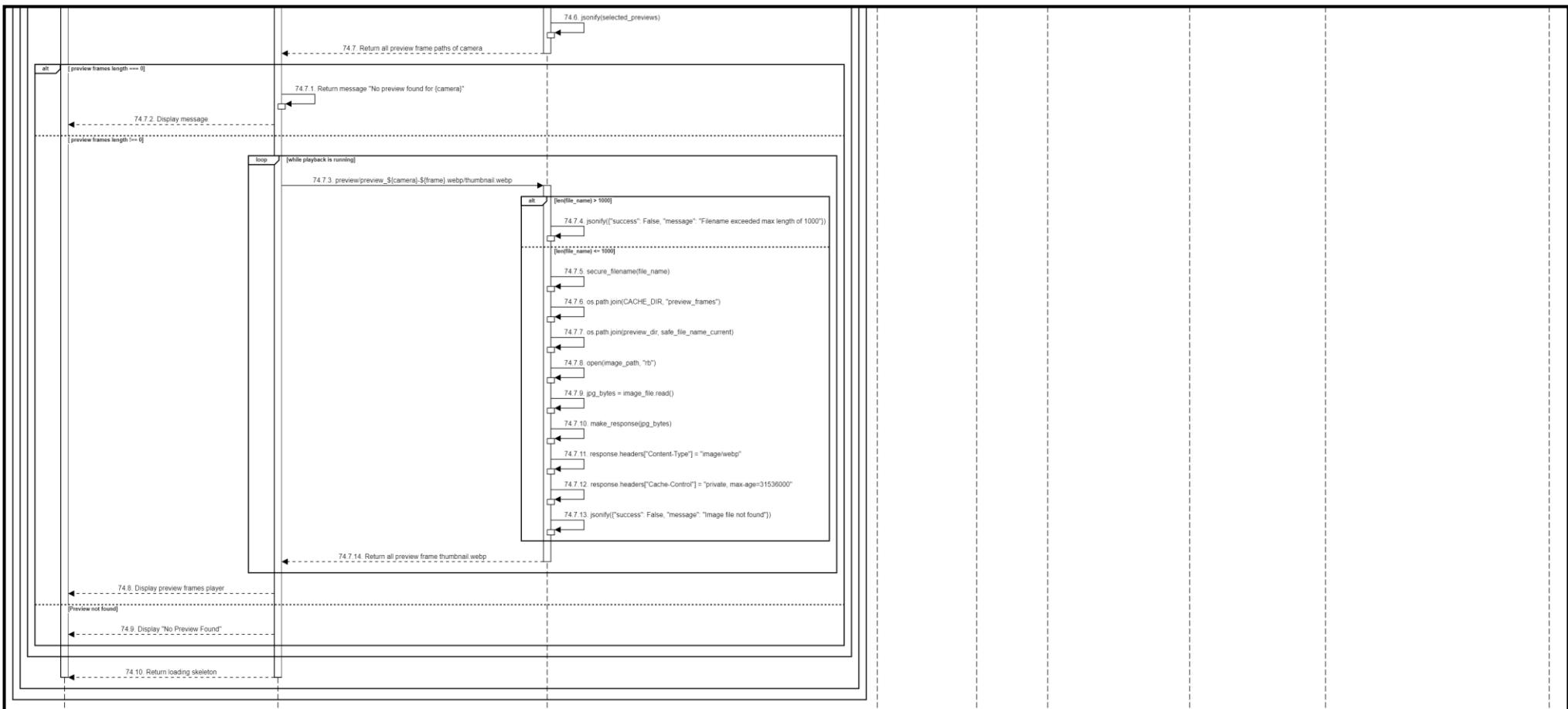
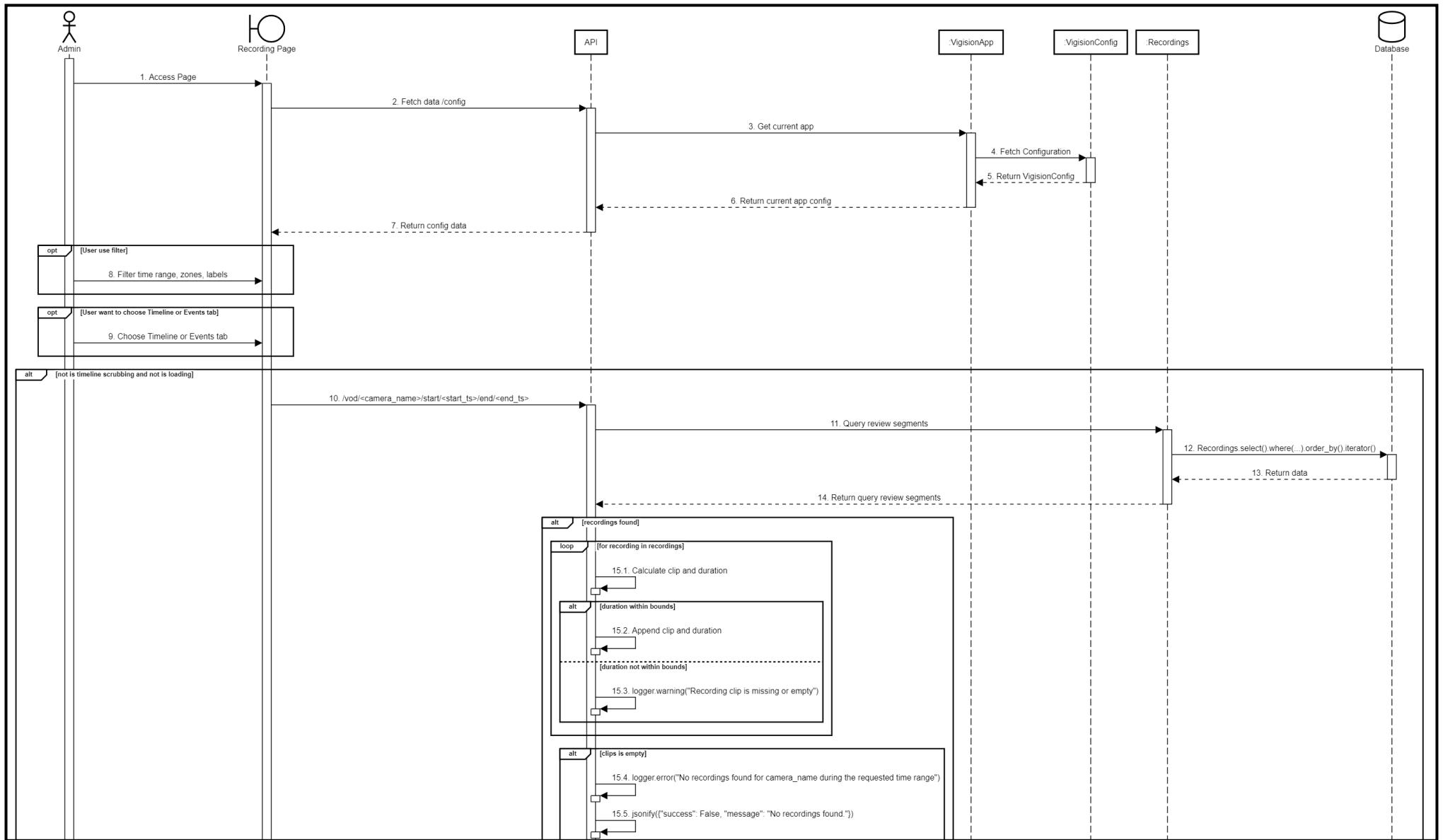
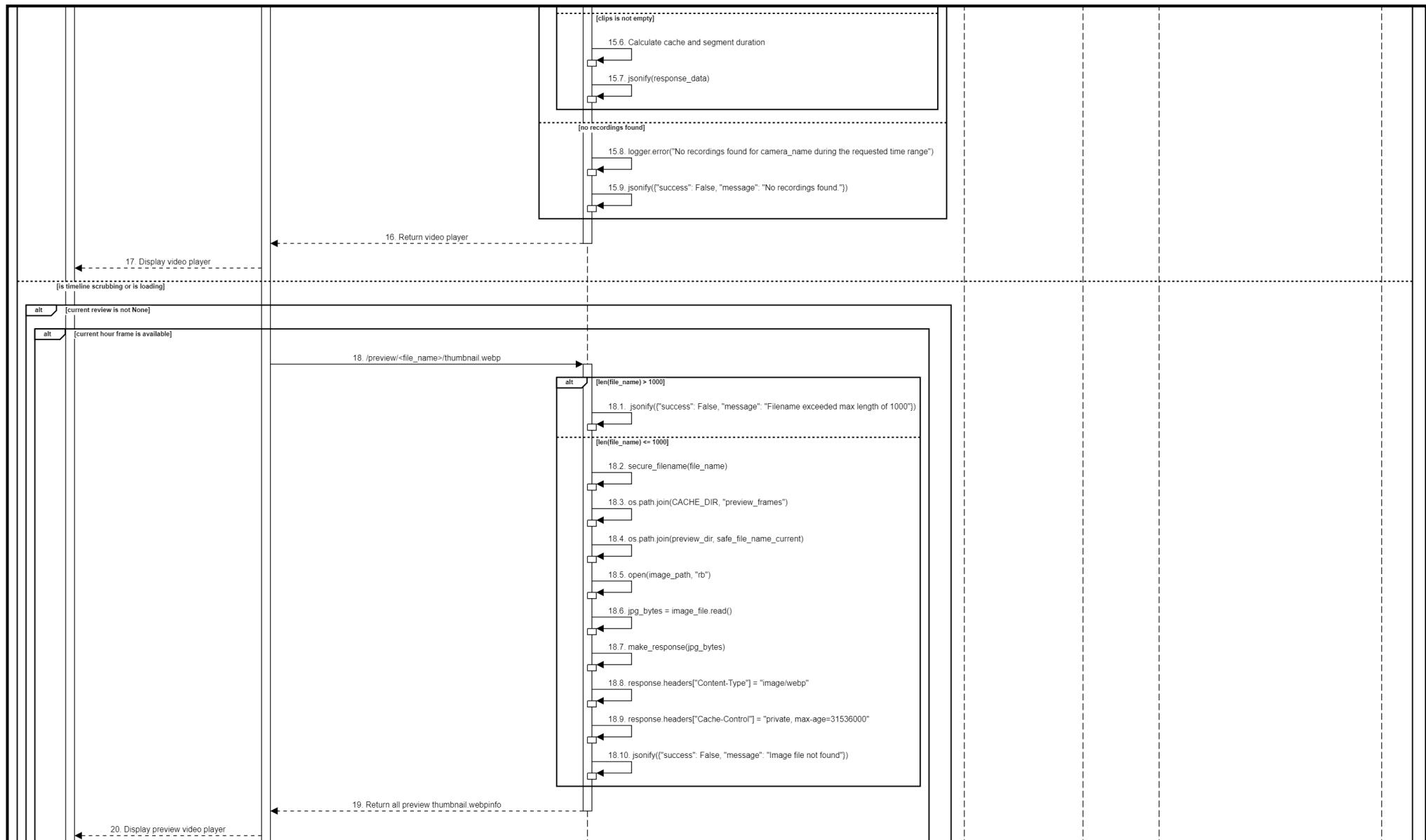
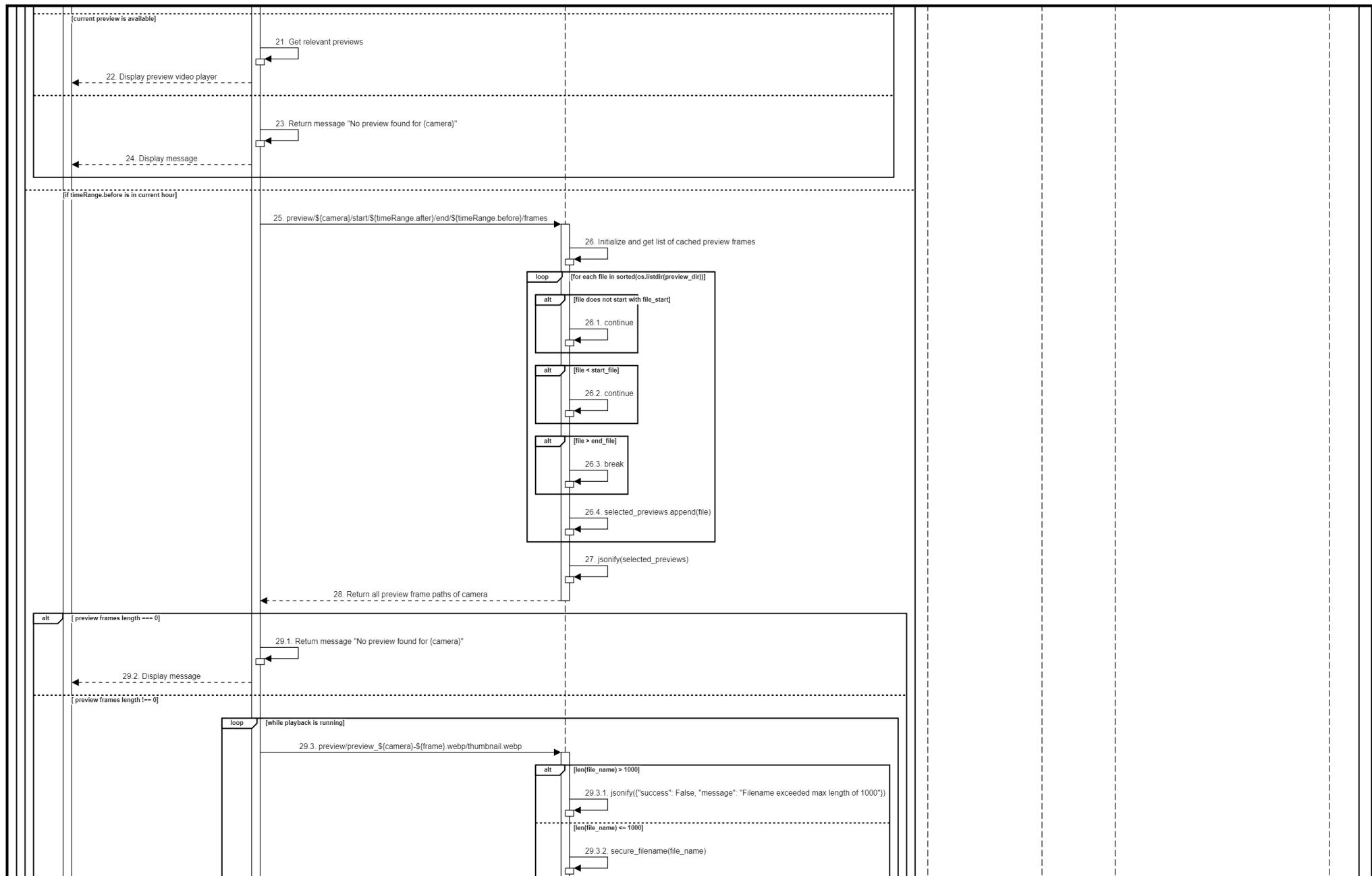


Figure 220. Sequence Diagram - View motion events

b. View event recording







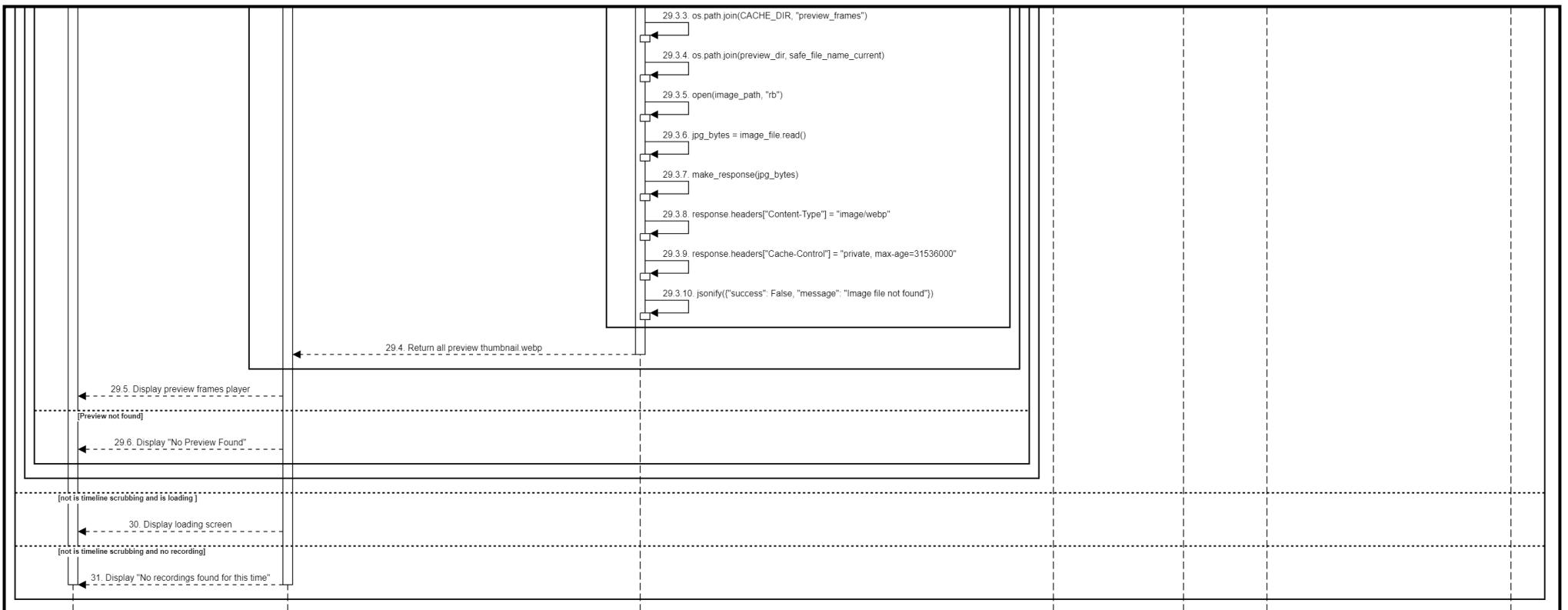


Figure 221. Sequence Diagram - View event recording

c. Mark events as reviewed

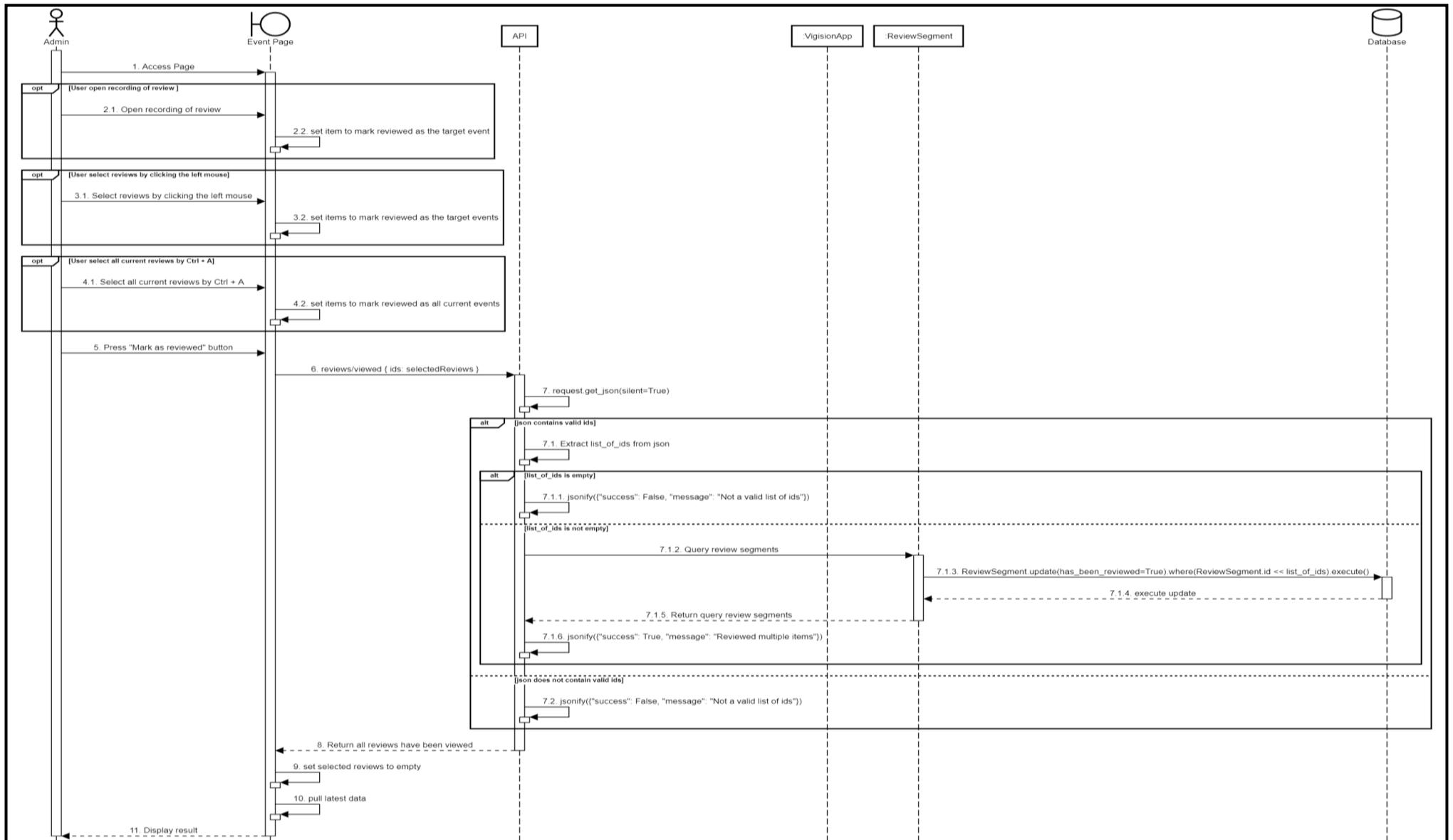
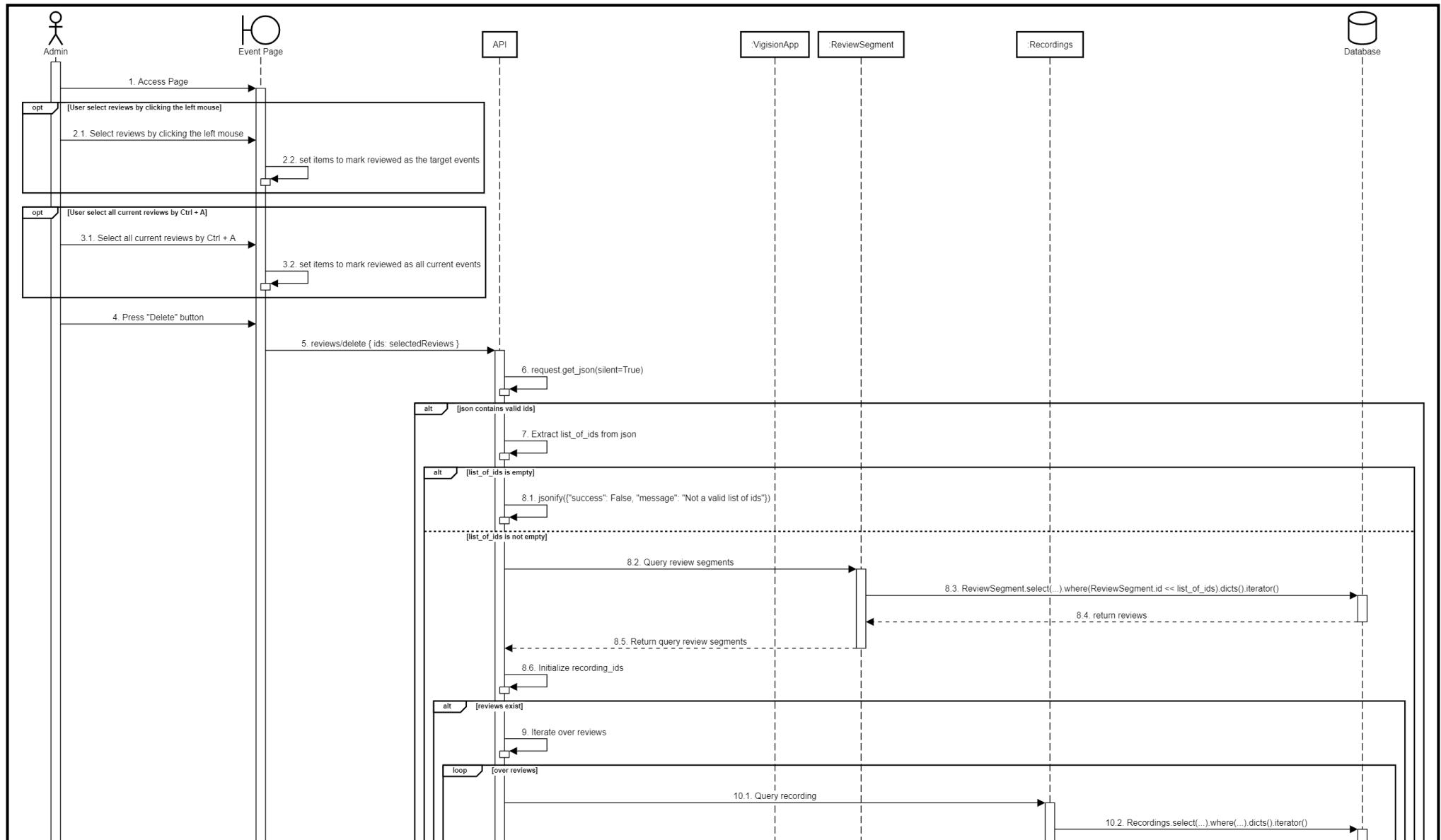


Figure 222. Sequence Diagram - Mark events as reviewed

d. Delete events



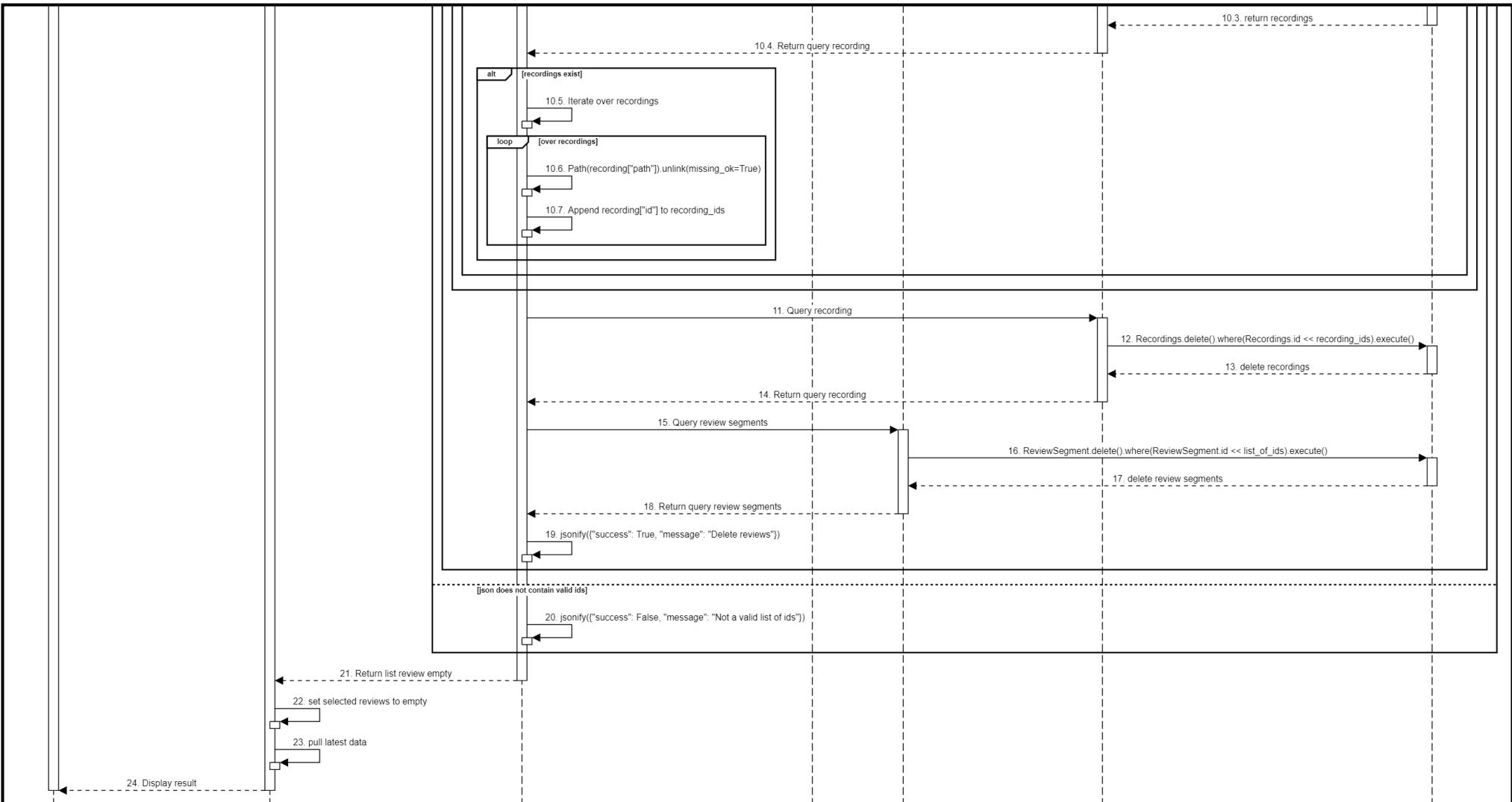


Figure 223. Sequence Diagram - Delete events

3.22 Manage exports

3.22.1 Class Diagram

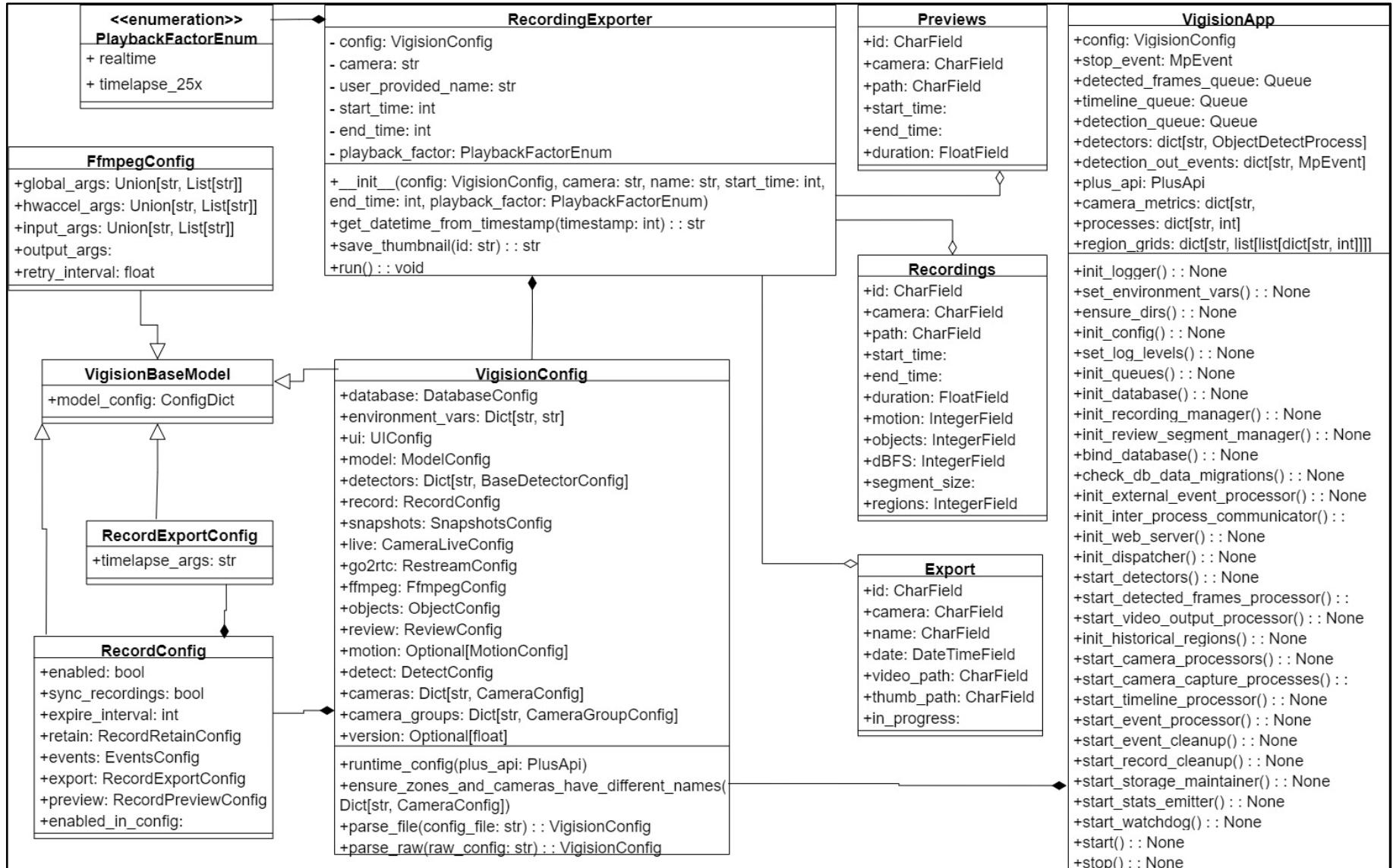


Figure 224. Class Diagram - Manage exports

3.22.2 Sequence Diagram

a. View export detail

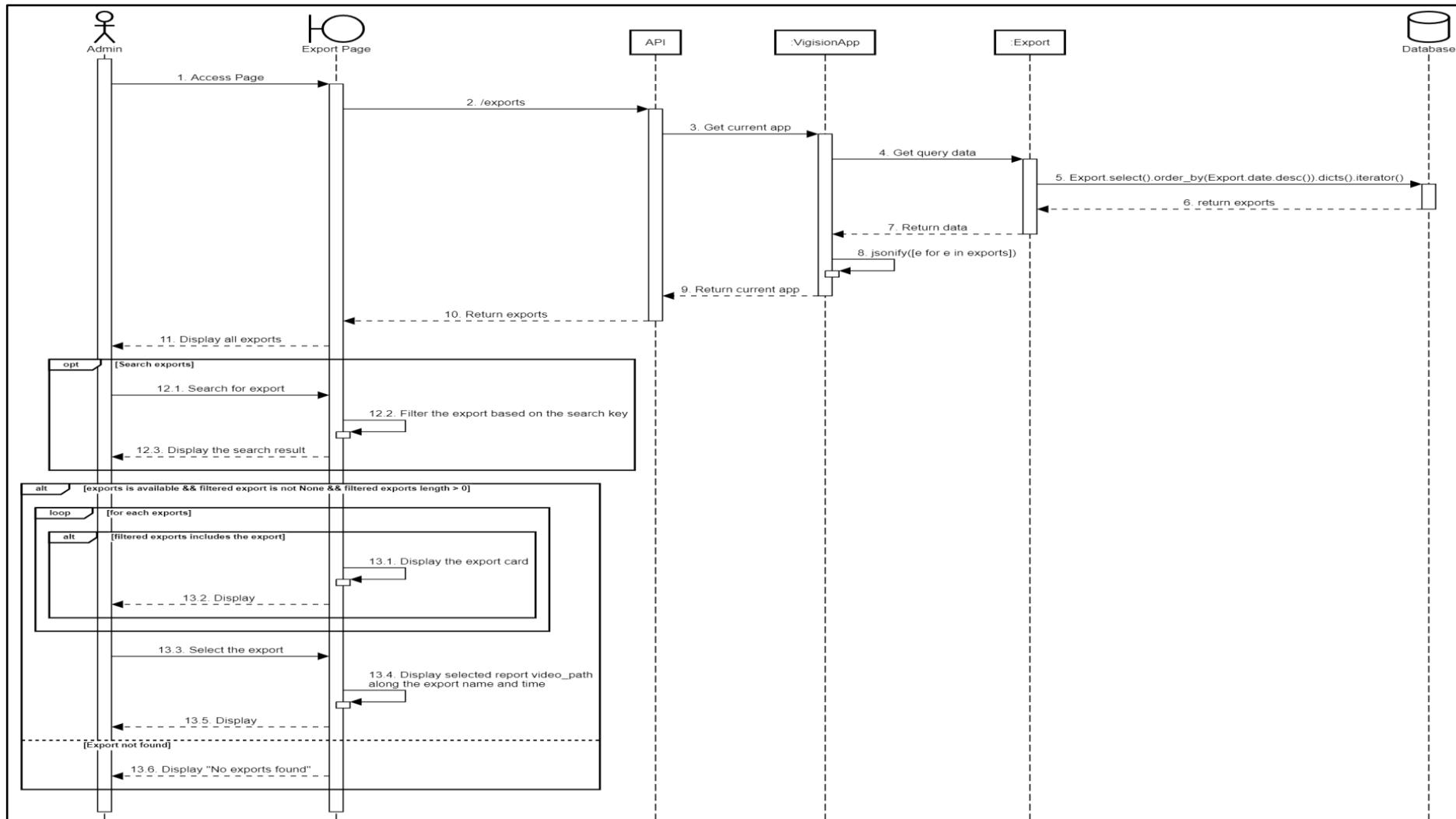
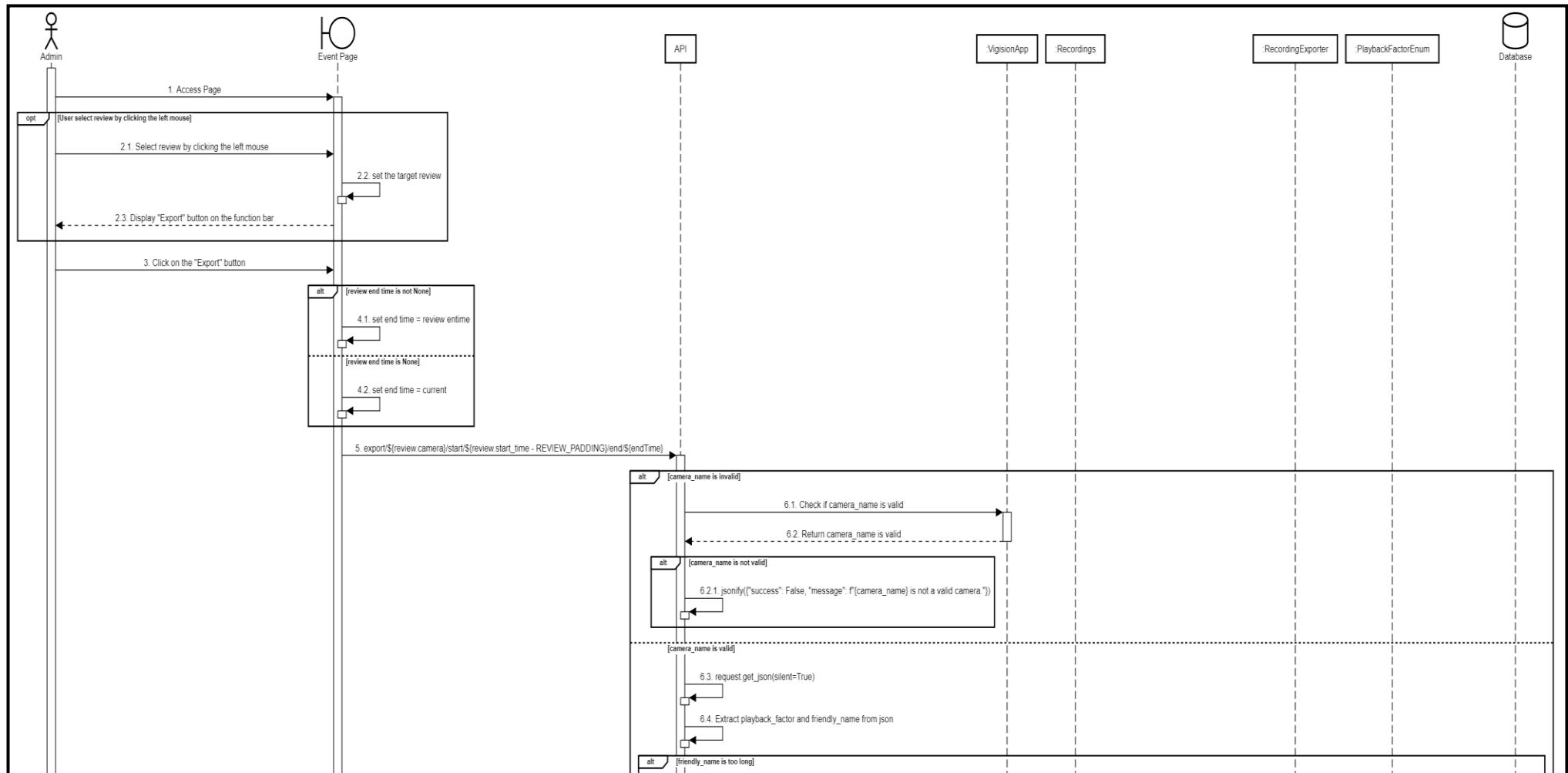


Figure 225. Sequence Diagram - View export detail

b. Export event recording



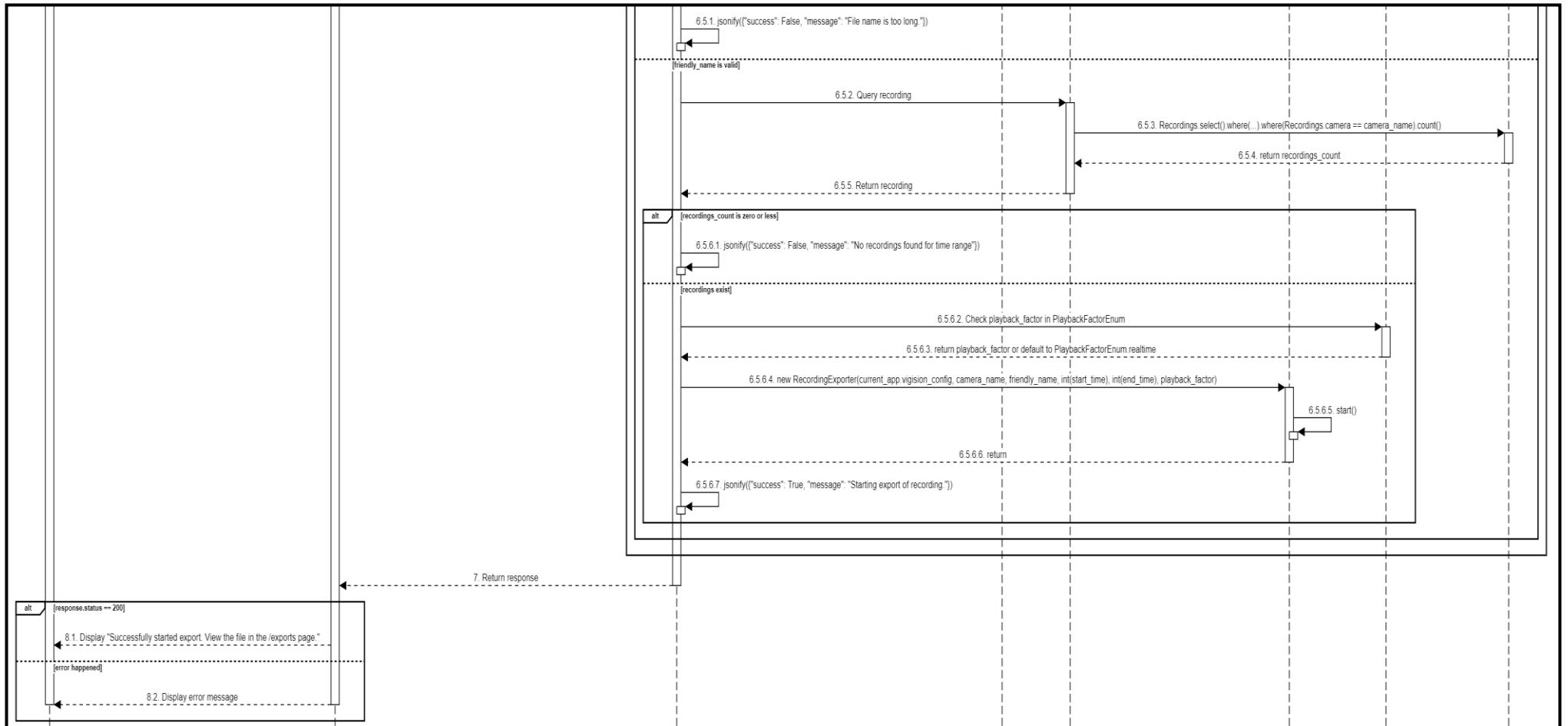
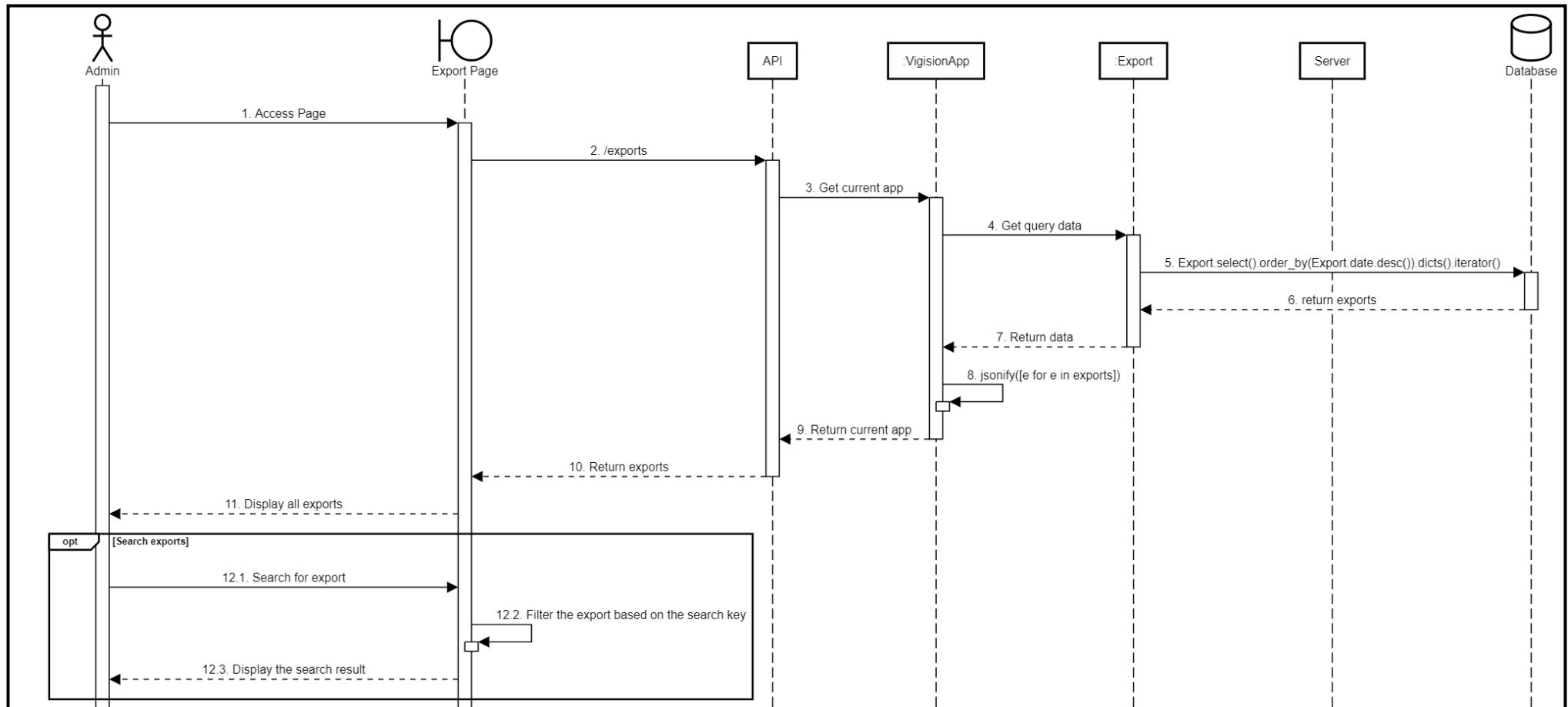


Figure 226. Sequence Diagram - Export event recording

c. Download export



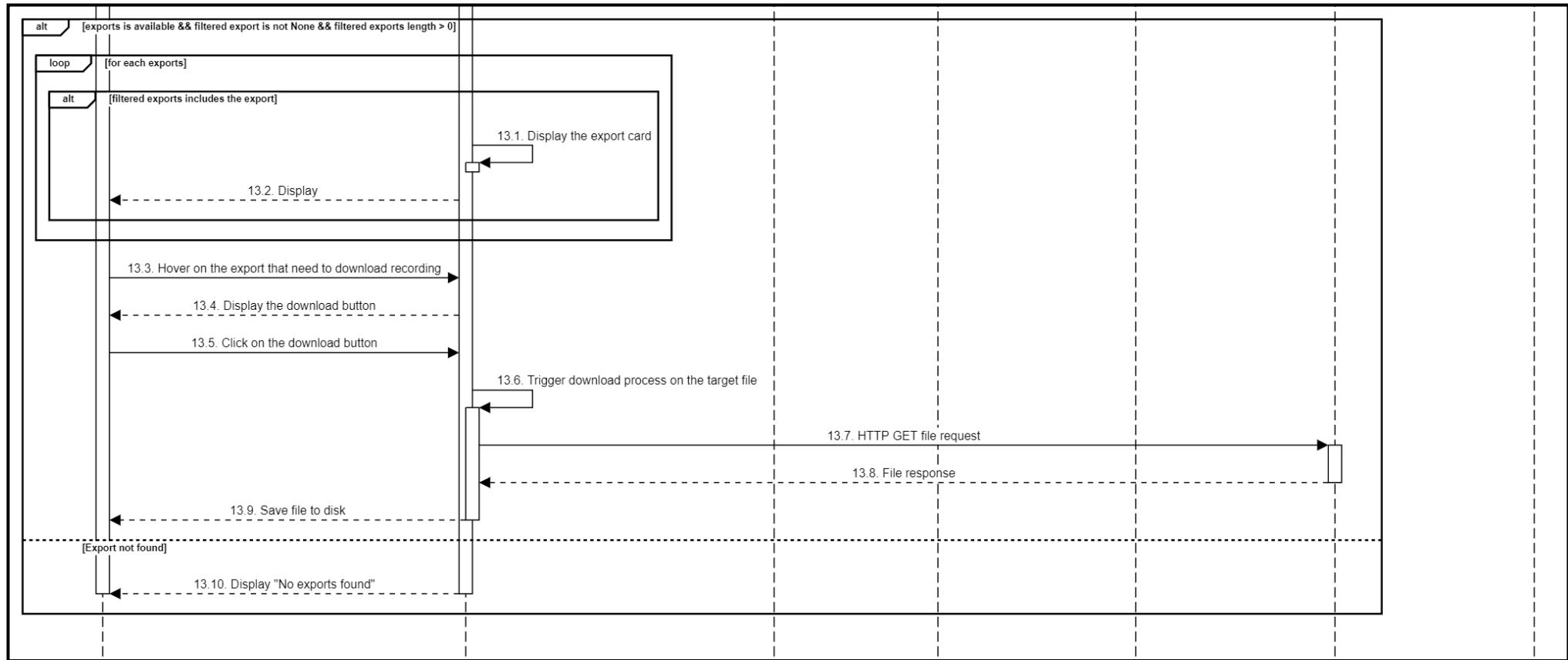
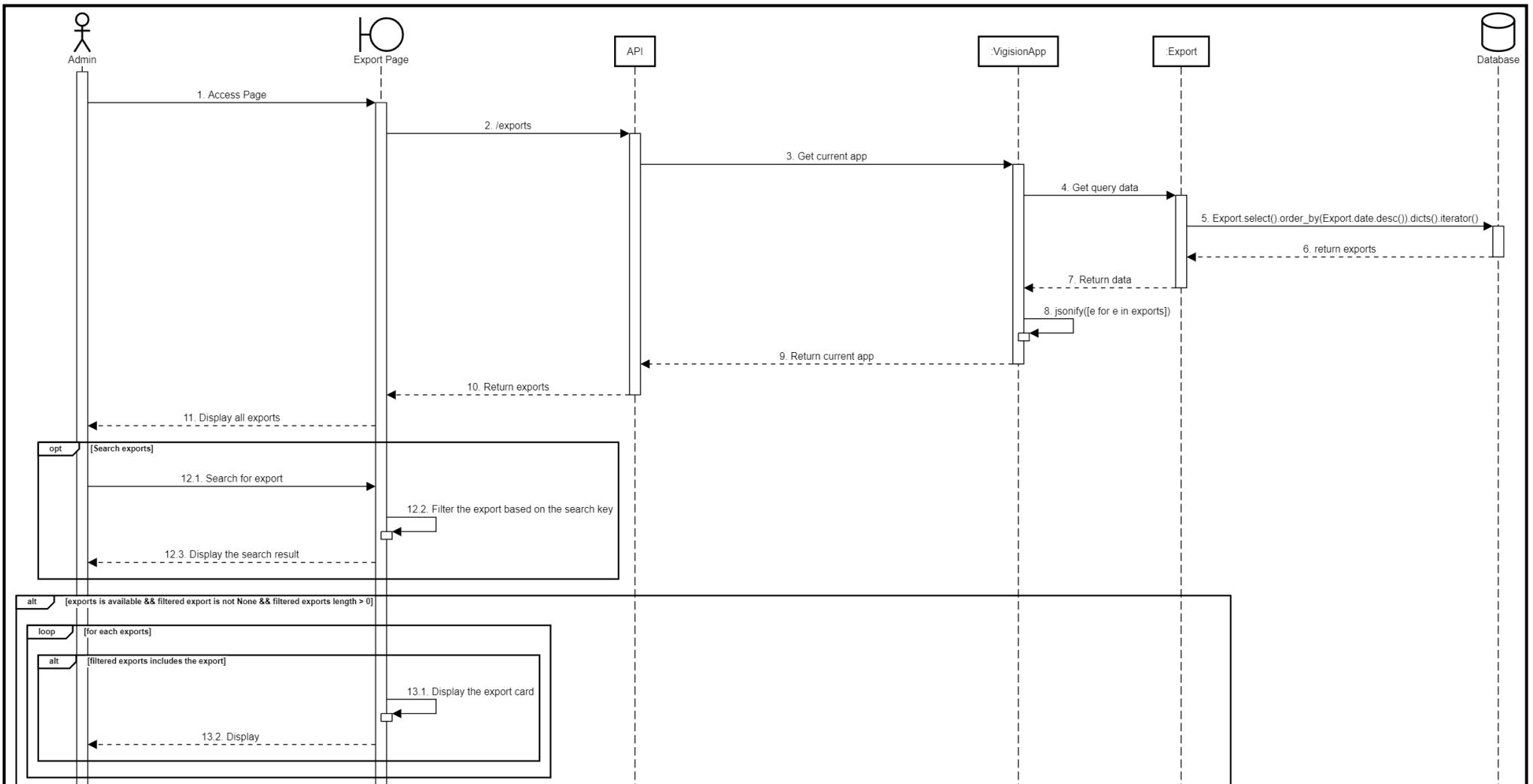


Figure 227. Sequence Diagram - Download export

d. Rename export



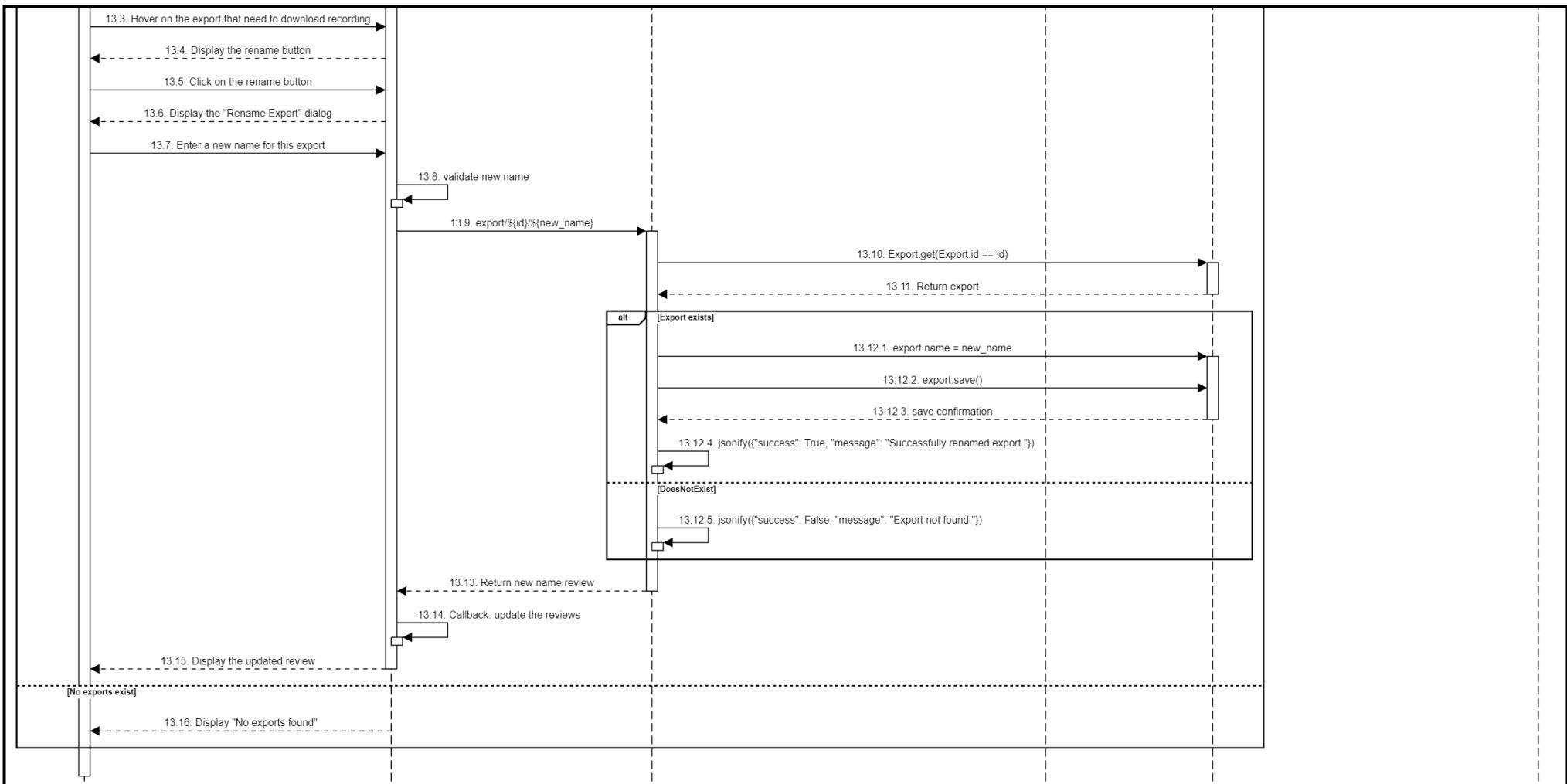
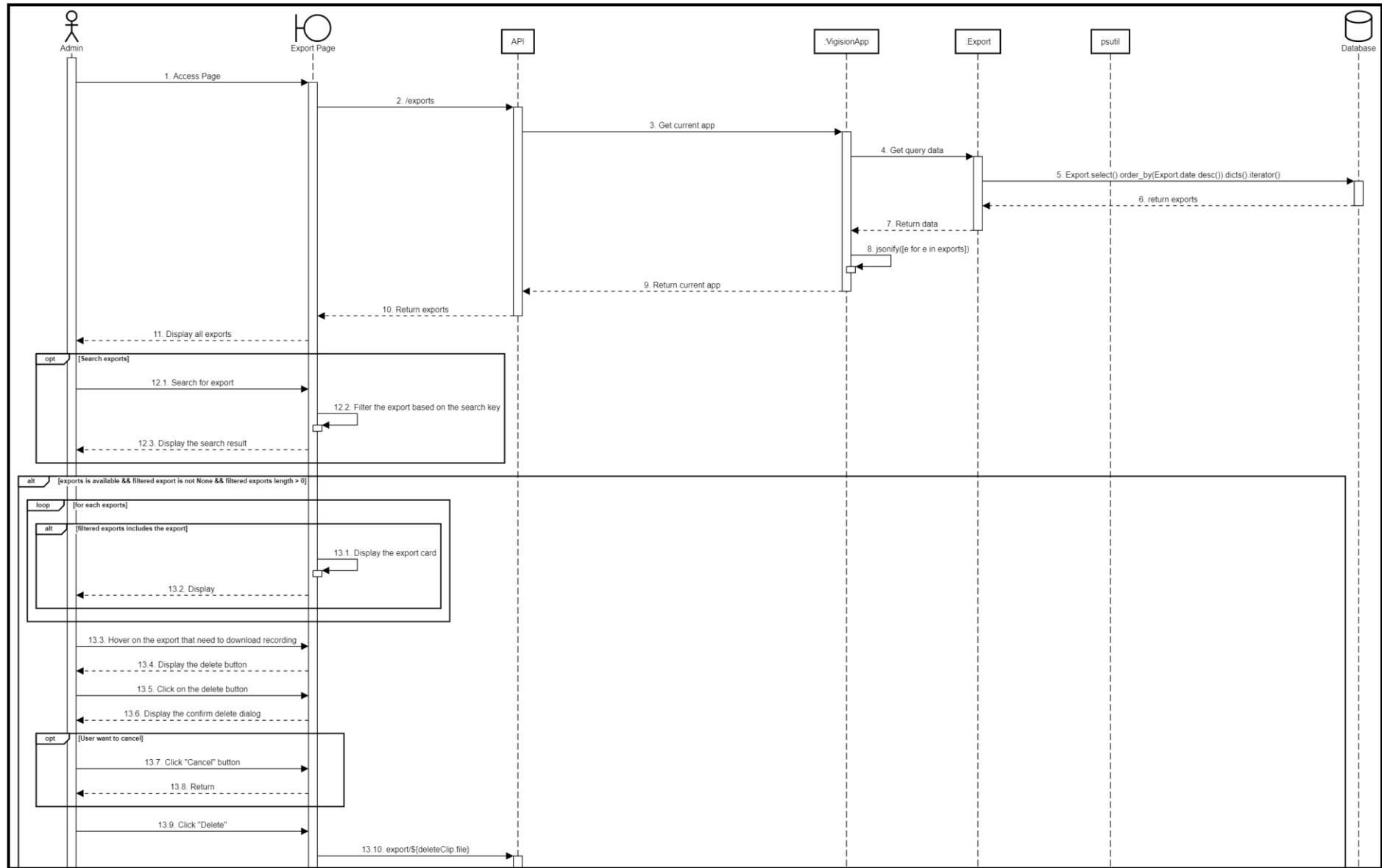


Figure 228. Sequence Diagram - Rename export

e. Delete export



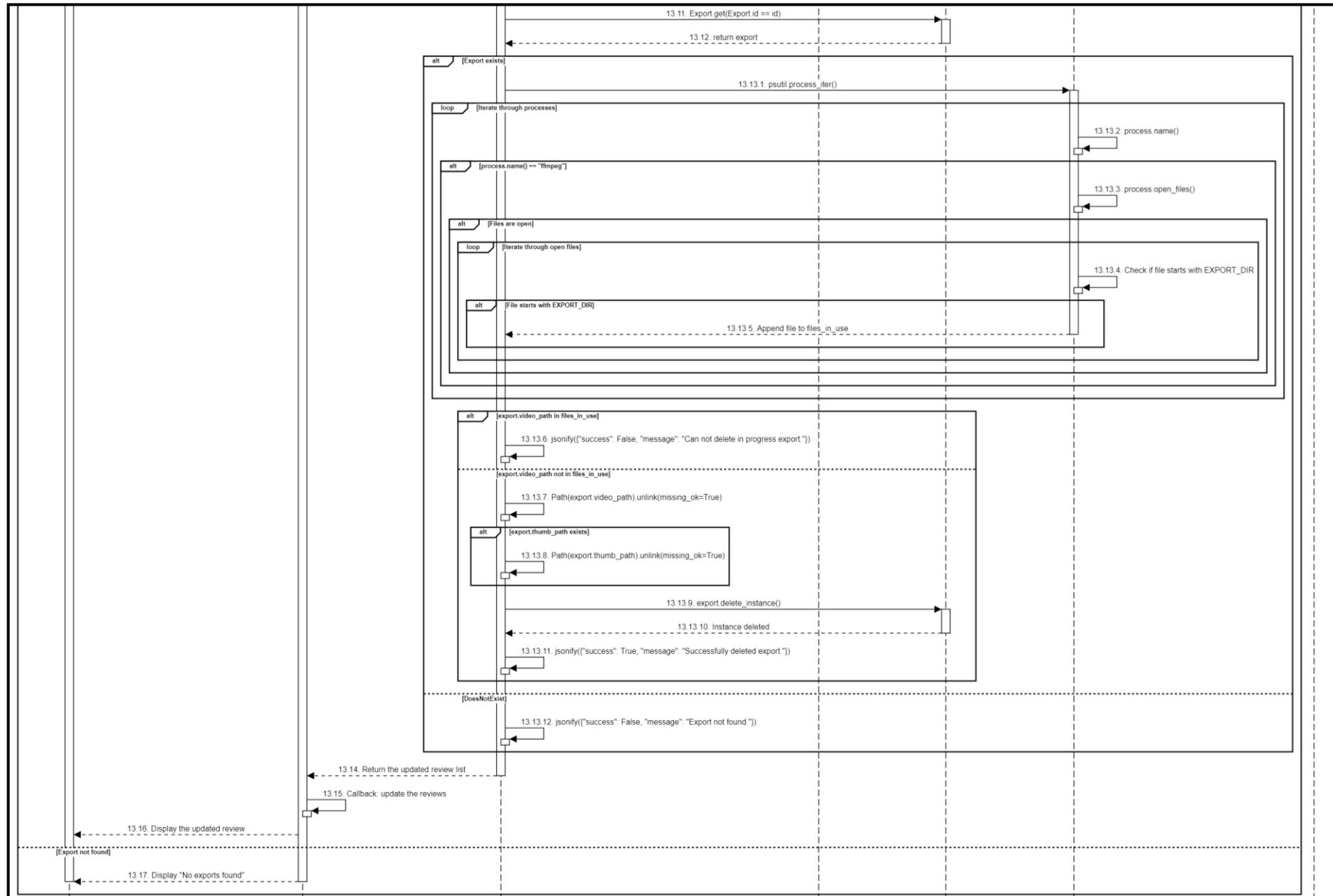


Figure 229. Sequence Diagram - Delete export

3.23 Config

3.23.1 Class Diagram

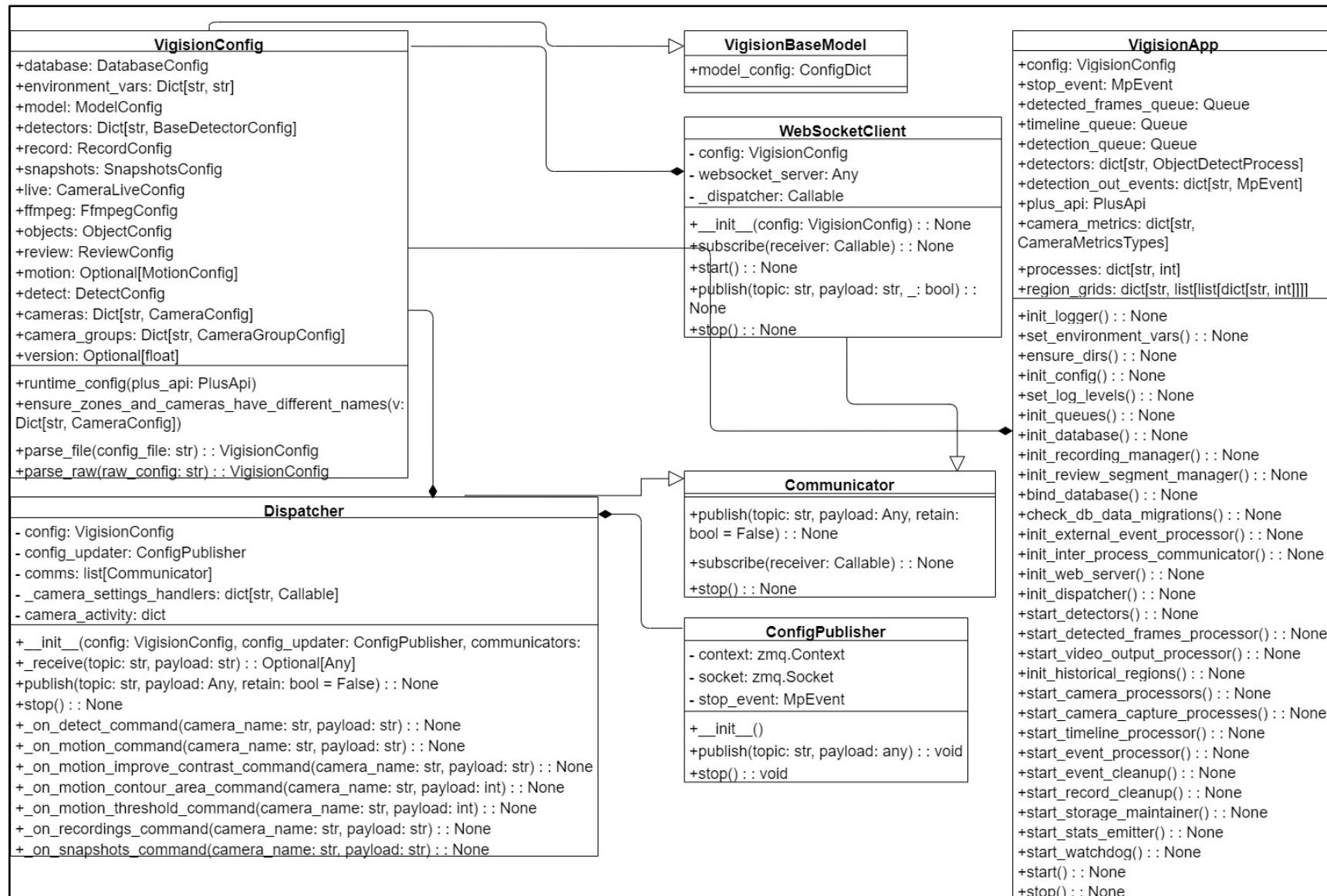


Figure 230. Class Diagram - Config

3.23.2 Sequence Diagram

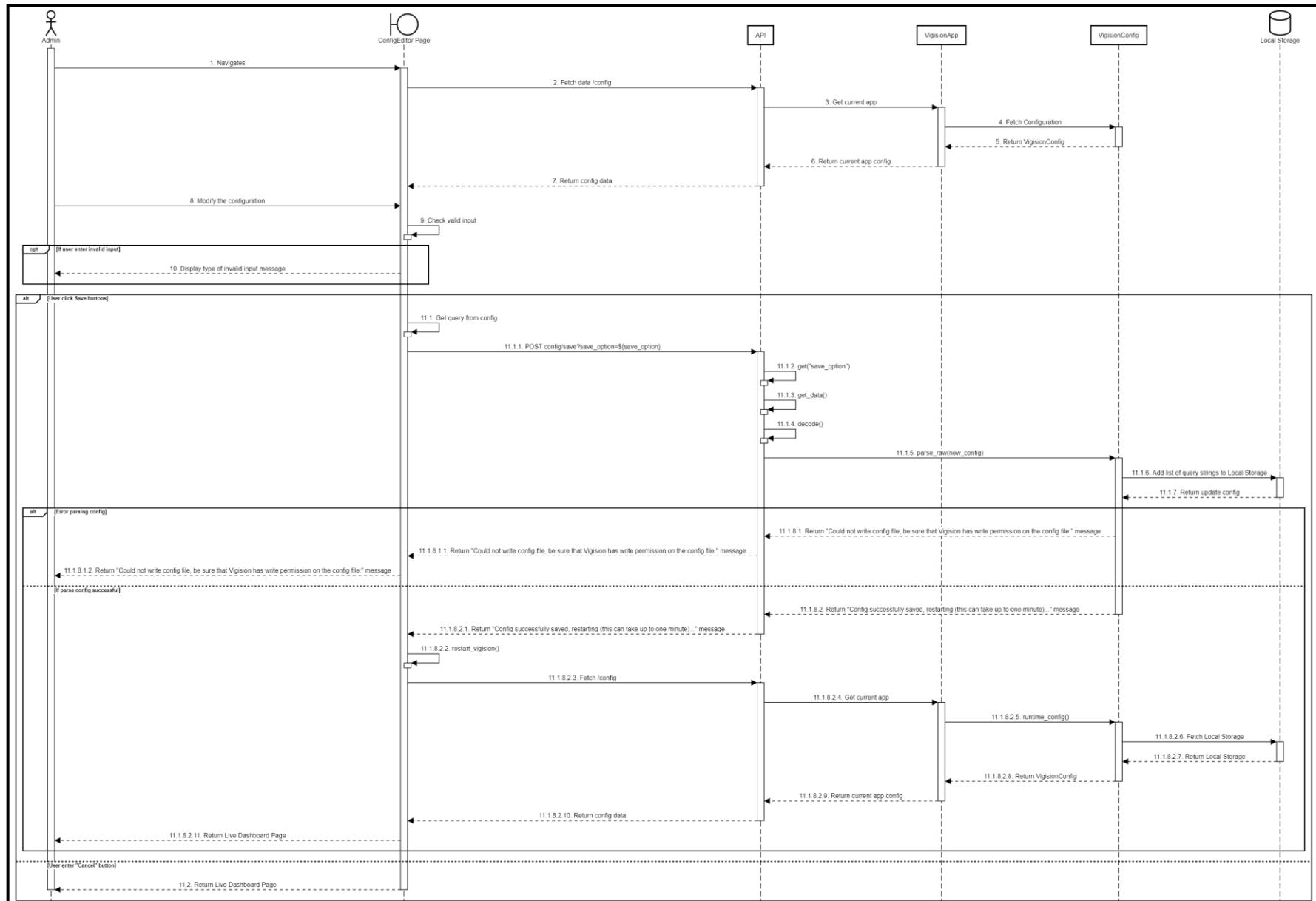


Figure 231. Sequence Diagram - Config

3.24 View system metrics

3.24.1 Class Diagram

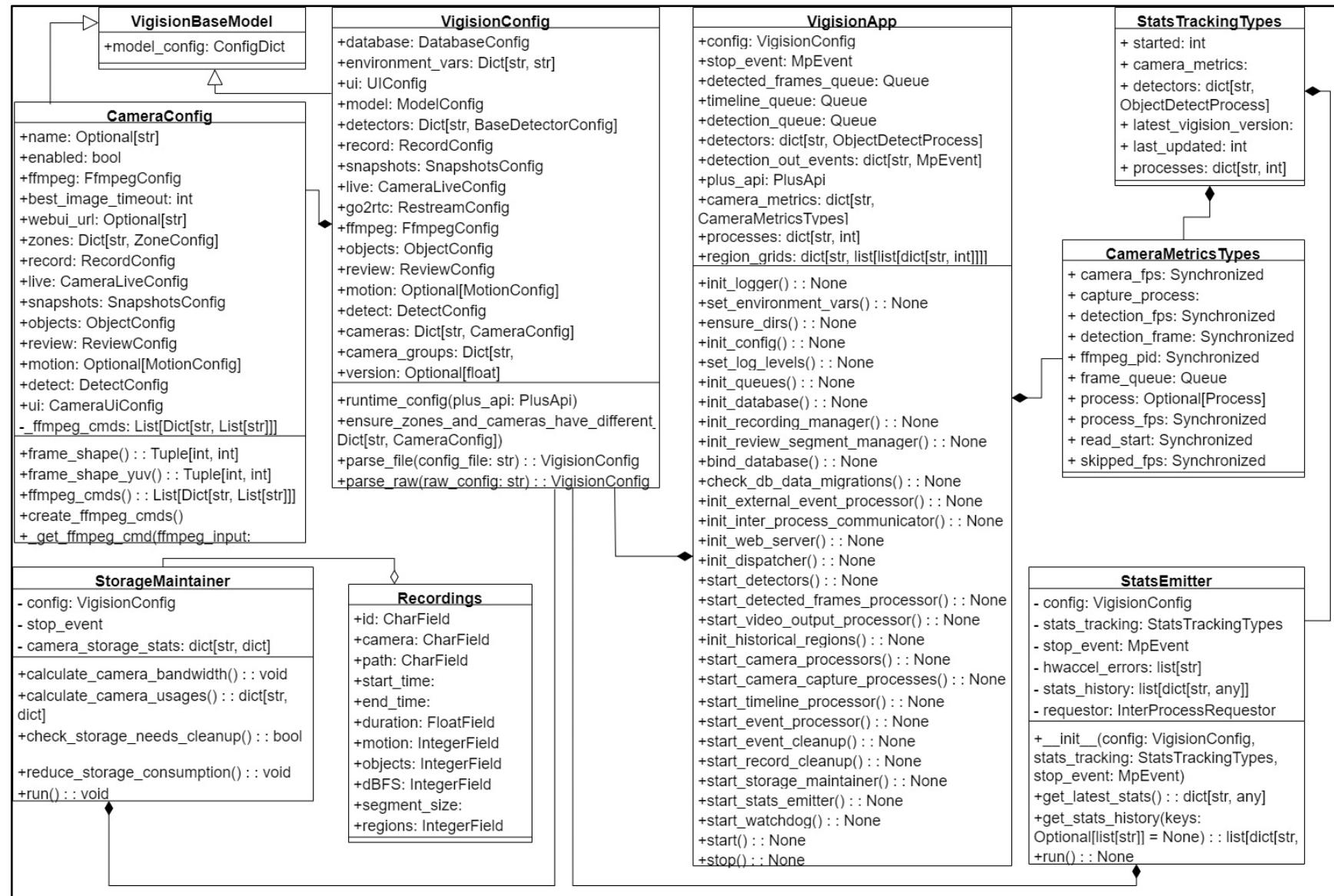
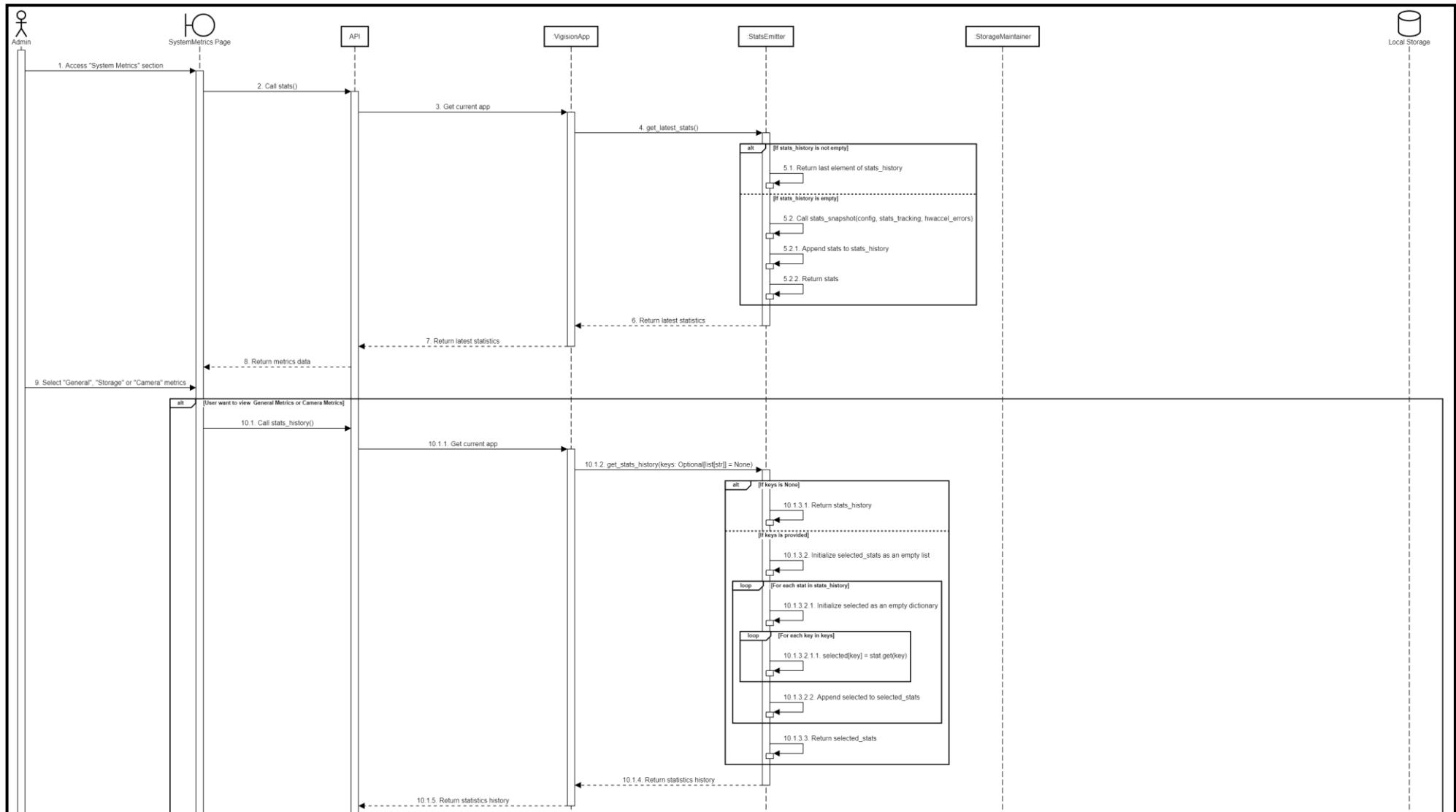


Figure 232. Class Diagram - View system metrics

3.24.2 Sequence Diagram



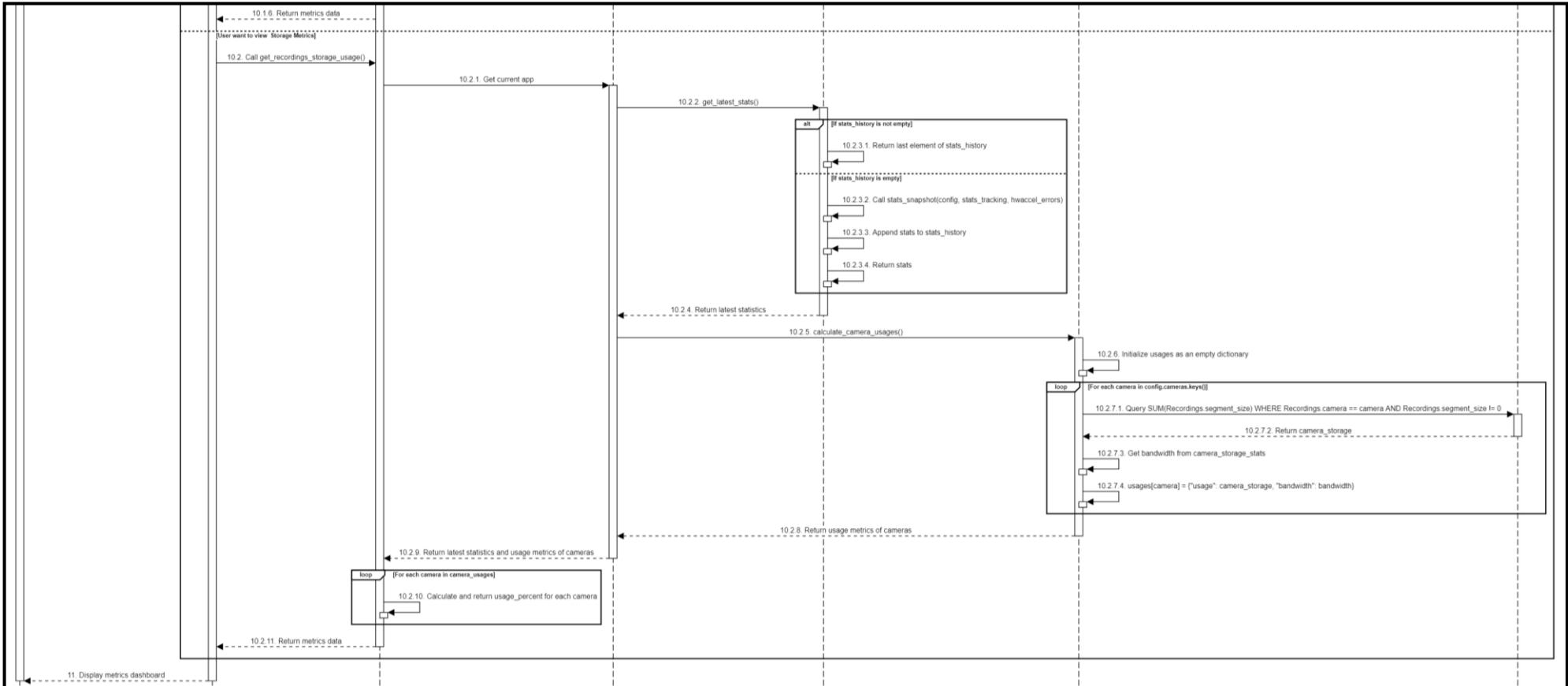


Figure 233. Sequence Diagram - View system metrics

3.25 Restart application

3.25.1 Class Diagram

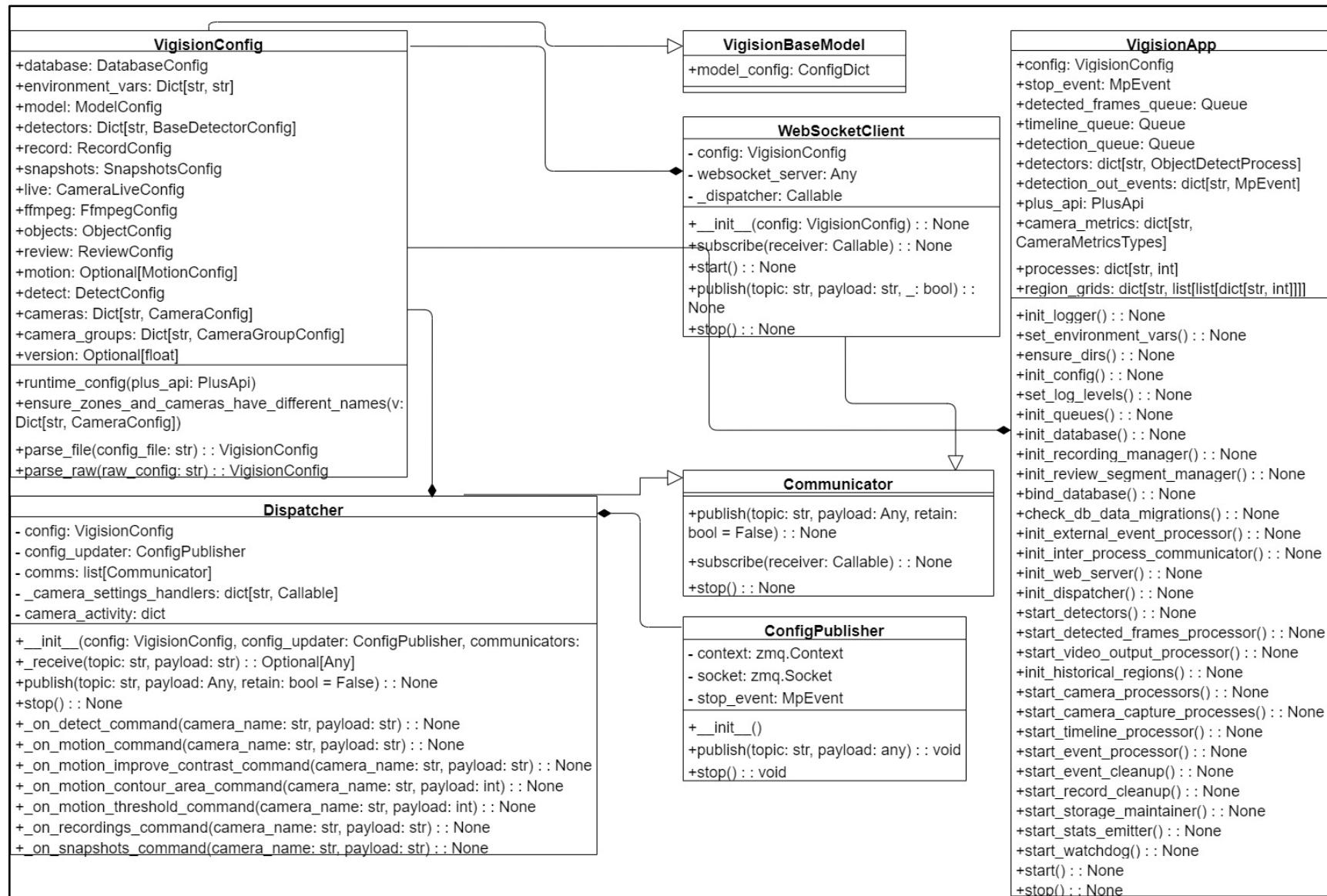


Figure 234. Class Diagram - Restart application

3.25.2 Sequence Diagram

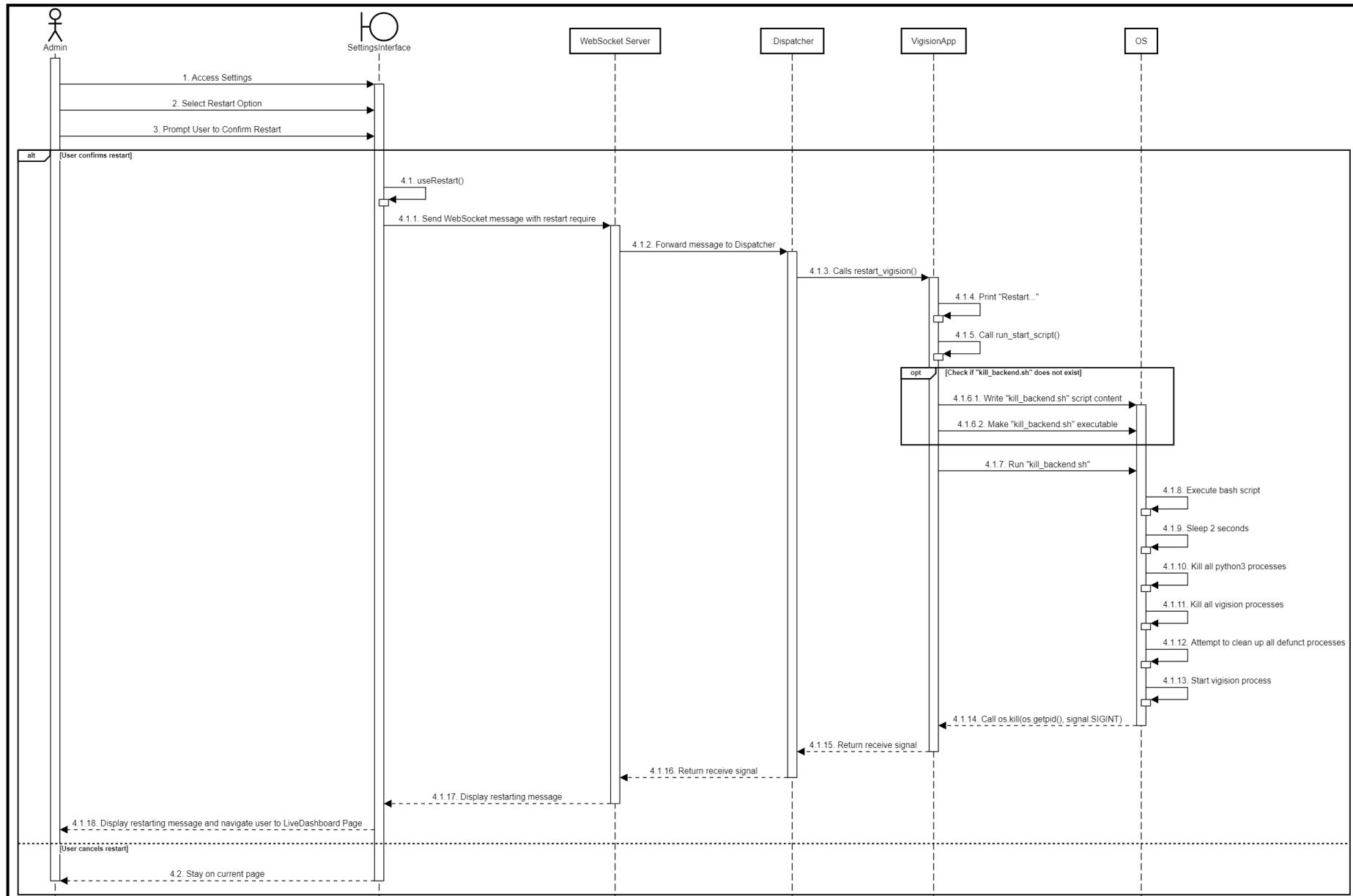


Figure 235. Sequence Diagram - Restart application

3.26 Change application theme

3.26.1 Class Diagram

This is frontend feature so don't have class diagram

3.26.2 Sequence Diagram

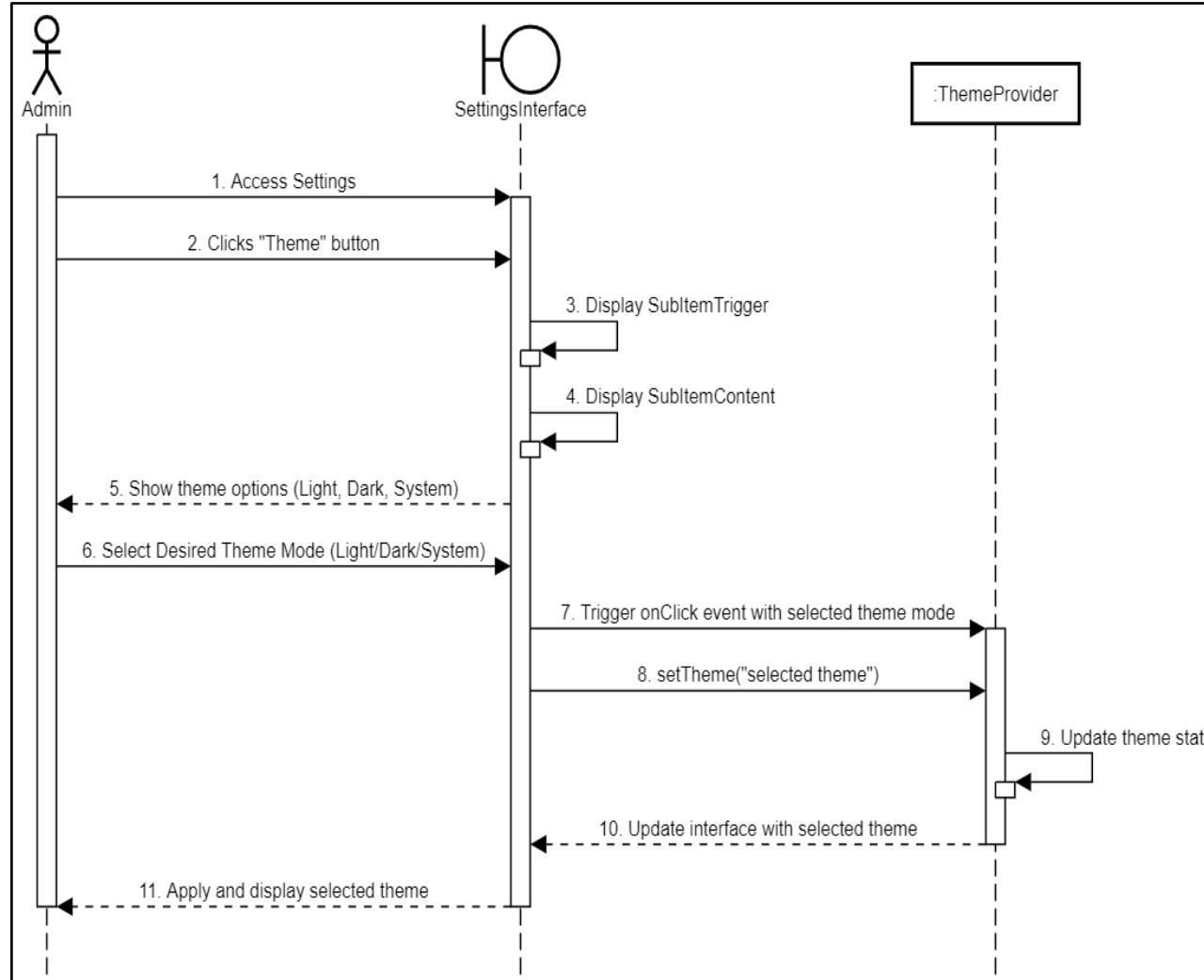


Figure 236. Sequence Diagram - Change application theme

4. Class Specifications

4.1 VigisionConfig

4.1.1 Properties of class

Table 69. Properties of class - VigisionConfig

No	Attribute	Description
01	database	Settings for database connectivity and operations.
02	environment_vars	A dictionary of environment variables specific to Vigision, allowing for dynamic configuration.
03	ui	User Interface configuration, affecting how the UI behaves or appears.
04	model	Configuration for the detection models used within the system.
05	detectors	Hardware-specific settings for detectors, possibly including types and sensitivity.
06	record	Global settings for recording data or events within the system.
07	snapshots	Configuration for taking and managing snapshots.
08	live	Settings for live playback features.
09	go2rtc	Configuration for restreaming or redistributing streams.
10	ffmpeg	Global configuration for FFmpeg, a multimedia framework for handling audio, video, and other streams.
11	objects	Configuration related to object management within the system.
12	review	Settings for reviewing recorded data or events.
13	motion	Optional settings for motion detection features.
14	detect	Configuration for object tracking and detection.
15	cameras	Settings for individual cameras, including their configurations.
16	camera_groups	Configuration for grouping cameras, possibly for collective operations or views.
17	version	An optional field indicating the current configuration version, useful for managing updates or migrations.

4.1.2 Class methods

Table 70. Class methods - VigisionConfig

No	Method	Description
01	runtime_config()	Merges and updates the configuration for a Vigision system at runtime, incorporating global settings, environment variables, and hardware-specific adjustments.
02	ensure_zones_and_cameras_have_different_names()	Helps maintain a clear and non-overlapping naming convention between zones and cameras in the configuration.
03	parse_file()	This method reads configuration data from a file specified by config_file. It supports both YAML and JSON formats.
04	parse_raw()	This method directly parses raw configuration data passed as a string (raw_config).

4.2 VigisionApp

4.2.1 Properties of class

Table 71. Properties of class - VigisionApp

No	Attribute	Description
01	model_config	Define configuration options for the model
02	stop_event	A multiprocessing event used to signal all running processes to stop, facilitating a graceful shutdown
03	detected_frames_queue	Queue for cameras to push tracked objects to
04	timeline_queue	Queue for timeline events
05	detection_queue	A multiprocessing queue for passing detection tasks or data between processes
06	detectors	A dictionary mapping identifiers to ObjectDetectProcess instances, managing different object detection processes
07	detection_out_events	A dictionary of multiprocessing events, likely used to signal the completion of detection tasks for each detector
08	plus_api	A custom API for additional functionalities or integrations
09	camera_metrics	Dictionary containing corresponding metrics or configuration for camera controls.
10	processes	A dictionary tracking process identifiers, likely used

		to manage and monitor the state of spawned subprocesses
11	region_grids	A dictionary mapping identifiers to grid configurations, possibly used for defining regions of interest in camera feeds for detection tasks

4.2.2 Class methods

Table 72. Class methods - VigisionApp

No	Method	Description
01	set_environment_vars()	Sets the environment variables based on the configuration. This method iterates through the `environment_vars` dictionary in the configuration and sets each key-value pair as an environment variable using `os.environ`.
02	set_log_levels()	This method sets the log levels for the root logger and other loggers specified in the configuration. It uses the log levels defined in the configuration to set the corresponding loggers' levels.
03	init_queues()	Initializes the queues for the application. This method creates two queues: - detected_frames_queue: A queue for cameras to push tracked objects to. The size of the detected_frames_queue is determined by the number of enabled cameras in the configuration. - timeline_queue: A queue for timeline events.
04	init_database()	Initializes the database by performing the following tasks: 1. Vacuums the database to optimize its performance. 2. Cleans up the timeline database by removing entries that do not have corresponding events. 3. Migrates the database location if necessary. 4. Migrates the database schema using peewee_migrate. 5. Makes a backup of the database before running migrations if there are any differences. 6. Cleans up the user database from beta (temporary check). 7. Checks if vacuum needs to be run based on the last vacuum timestamp.
05	init_recording_manager()	This method initializes and starts a separate process dedicated to managing recordings, as part of a larger application that involves video processing or surveillance. It creates a new process using

		Python's multiprocessing (mp) module, specifying manage_recordings as the target function to run in this process, which contains configuration settings necessary for the recording management task
06	init_review_segment_manager()	This method initializes and starts a multiprocessing process aimed at managing review segments within an application. It creates a new process with the specific task (manage_review_segments) and configuration to handle review segments, such as video segment reviews, efficiently.
07	bind_database()	This method initializes and configures a SQLite database connection for the main process of an application. It creates an instance of SqliteQueueDatabase with specific settings, including path from the application's configuration, database pragmas for performance and behavior, and a dynamic timeout based on the number of enabled cameras in the configuration. After establishing the database connection, it binds a list of model classes to this database instance, enabling these models to interact with the database for operations like querying and storing data.
08	check_db_data_migrations()	This method performs a check to determine if data migrations related to exports need to be executed
09	init_inter_process_communicator()	This method sets up the infrastructure needed for different parts of the application to communicate and coordinate actions across processes.
10	init_dispatcher()	This method initializes a Dispatcher object, which is likely responsible for managing and routing messages or commands within the application, playing a central role in handling various communication and control tasks within the application, leveraging different protocols and methods as needed.
11	start_detectors()	Initializes the infrastructure required for object detection processes in an application, specifically targeting configurations involving multiple cameras and detectors. This method effectively sets up a multi-process object detection system, leveraging shared memory for efficient data exchange and multiprocessing events for synchronization, tailored to the application's configured cameras and detectors.
12	start_video_output_processor()	Starts the video output processor as a separate process. This method creates a new process using the `output_frames` function as the target. The `output_frames` function is responsible for processing and outputting video frames. The

		process is started as a daemon process, which means it will automatically terminate when the main process exits.
13	init_historical_regions()	This method ensures that the application's region grids are up-to-date with the current camera configuration, removing outdated entries and adding or updating grids as necessary for operational cameras.
14	start_detected_frames_processor()	Initializes and starts a TrackedObjectProcessor for handling detected frames. This method calls its start method to begin processing detected frames, likely involving tracking objects across frames based on detection results.
15	start_camera_processors()	This method enables concurrent processing of video streams from multiple cameras, leveraging multiprocessing to handle each camera's video feed in parallel for tasks such as object detection, tracking based on detection results.
16	start_camera_capture_processes()	This method enables the application to concurrently capture video from multiple cameras by running separate processes for each, improving the efficiency and responsiveness of video processing tasks.
17	start_storage_maintainer()	This method ensures that the application has an active component dedicated to managing storage efficiently, potentially preventing issues related to storage overflow or the accumulation of outdated data.
18	init_external_event_processor()	This method initializes an instance of ExternalEventProcessor with the application's configuration and assigns it to the self.external_event_processor attribute. The ExternalEventProcessor is responsible for handling or processing external events, using the provided configuration to tailor its behavior to the application's needs.
19	start_stats_emitter()	This method is responsible for setting up and starting the statistics emission functionality, which involve collecting, aggregating, and possibly reporting or logging runtime statistics related to the application's performance, camera metrics, detection processes.
20	init_web_server()	The init_web_server method initializes the application's web server by creating a Flask application instance, setting up the web server with the necessary context and functionalities derived from the preset components.

21	start_timeline_processor()	This method initializes and starts a TimelineProcessor instance. After initialization, it calls the start method on the TimelineProcessor instance to begin its operation, which involves processing timeline-related tasks asynchronously.
22	start_event_processor()	This method initializes and starts an EventProcessor instance, it calls the start method on the EventProcessor instance to begin its operation, which involves processing events asynchronously based on the provided configuration and queue.
23	start_event_cleanup()	This method initializes and starts an EventCleanup instance. After initialization, it calls the start method on the EventCleanup instance to begin its operation, which involves cleaning up or managing events asynchronously based on the provided configuration and stop signal.
24	start_record_cleanup()	This method initializes and starts a RecordingCleanup instance for managing and cleaning up recording files. It creates this instance with the application's configuration and a stop event, then begins its operation by calling its start method, running the cleanup process in a separate thread or process.
25	start_watchdog()	This method initializes and starts a VigisionWatchdog instance. After initialization, it calls the start method on the VigisionWatchdog instance to begin its operation, which involves monitoring or managing the detectors in a separate thread or process, ensuring they are functioning correctly and taking action based on the stop event if necessary.
26	check_shm()	This method calculates and checks if the available shared memory is sufficient for the application's requirements based on the configured cameras.
27	start()	This method is the main entry point for starting the Vigision application, a Network Video Recorder (NVR) with real-time local object detection for IP cameras. It orchestrates the entire lifecycle of the Vigision application, from initialization and configuration validation to starting all necessary components.
28	stop()	This method ensures that all components of the application are properly terminated, resources are cleaned up, and the application exits cleanly.

4.3 CameraConfig

4.3.1 Properties of class

Table 73. Properties of class - CameraConfig

No	Attribute	Description
01	name	An optional string to name the camera.
02	enabled	A boolean indicating if the camera is enabled or not.
03	ffmpeg	Configuration for FFmpeg, specifying how video streams are processed.
04	best_image_timeout	Time in seconds to wait for the highest confidence score image from the camera.
05	webui_url	An optional URL for direct access to the camera from the system page.
06	zones	A dictionary mapping zone names to their configurations, for defining areas of interest within the camera's field of view.
07	record	Configuration for recording camera footage.
08	live	Settings for live playback of the camera feed.
09	snapshots	Configuration for taking and managing snapshots from the camera.
10	objects	Object detection configuration, specifying which objects to detect and how.
11	review	Configuration for reviewing footage or detections.
12	motion	Optional motion detection configuration.
13	detect	Configuration for object detection.
14	ui	Modifications for the camera's representation in the user interface.
15	_ffmpeg_cmds	A private attribute storing FFmpeg commands generated based on the configuration.

4.3.2 Class methods

Table 74. Class methods - CameraConfig

No	Method	Description
01	frame_shape()	A property that returns the frame dimensions (height and width) used for detection.

02	frame_shape_yuv()	A property that returns the YUV frame dimensions, which are different due to the YUV color space's structure.
03	ffmpeg_cmds()	A property that provides access to the FFmpeg commands stored in _ffmpeg_cmds.
04	create_ffmpeg_cmds()	Generates FFmpeg commands based on the camera's configuration, particularly focusing on input roles and processing requirements.
05	_get_ffmpeg_cmd()	Construct individual FFmpeg command lines based on specific input configurations and roles.

4.4 RecordConfig

4.4.1 Properties of class

Table 75. Properties of class - RecordConfig

No	Attribute	Description
01	enabled	This boolean attribute determines whether recording is enabled across all cameras in the system.
02	sync_recordings	This attribute specifies whether the system should synchronize recordings with the disk at startup and then once daily. This ensures that recordings are backed up and consistent with the disk's state.
03	expire_interval	An integer that sets the number of minutes between cleanup runs, where the system checks for and removes expired recordings based on retention policies.
04	retain	Embeds a RecordRetainConfig instance, defining the retention policy for recordings, such as how long they should be kept. This allows for detailed control over data retention, ensuring compliance with storage limitations or privacy regulations.
05	events	Contains an EventsConfig instance, which configures settings specific to event-based recordings, such as pre-capture and post-capture durations, and which events trigger recordings.
06	export	Holds a RecordExportConfig instance, detailing configurations for exporting recordings, like timelapse creation settings. This allows for customization of how recordings are processed and shared.
07	preview	Embeds a RecordPreviewConfig instance, specifying the quality of recording previews. This affects how previews are generated and displayed,

		balancing between quality and resource usage.
08	enabled_in_config	An optional boolean that tracks the original state of recording enablement, allowing the system to remember if recording was initially enabled or disabled.

4.4.2 Class methods

This class doesn't have class methods.

4.5 RecordingMaintainer

4.5.1 Properties of class

Table 76. Properties of class - RecordingMaintainer

No	Attribute	Description
01	name	A string identifier for the thread, set to "recording_maintainer". This helps in identifying the thread during debugging or logging.
02	config	An instance of VigisionConfig, which holds the configuration settings for the recording maintainer. This could include settings related to recording durations, storage limits, and criteria for retaining recordings.
03	requestor	An instance of InterProcessRequestor, likely used for making requests to other processes or services, possibly to fetch or send data related to recordings.
04	config_subscriber	An instance of ConfigSubscriber initialized with a path "config/record/". This suggests it listens for configuration changes related to recording settings, allowing the maintainer to adapt to configuration updates dynamically.
05	detection_subscriber	An instance of DetectionSubscriber set to listen for all detection types (DetectionTypeEnum.all). This implies the maintainer uses detection events (e.g., motion, sound, object detection) to make decisions about recordings.
06	stop_event	An instance of MpEvent (likely a multiprocessing event), used to signal the thread to stop its execution. This allows for a graceful shutdown of the thread when necessary.
07	object_recordings_info	A dictionary mapping strings to lists, initialized as a defaultdict(list). This structure is likely used to keep track of information related to object detections within recordings.
08	end_time_cache	A dictionary mapping strings to tuples of

		datetime.datetime and float, initialized as an empty dictionary. This cache might be used to store the end times of recordings along with some additional float value (possibly a metric or identifier) for quick access.
--	--	---

4.5.2 Class methods

Table 77. Class methods - RecordingMaintainer

No	Method	Description
01	<code>__init__()</code>	Initializes the thread with the provided configuration and stop event. It sets up the necessary infrastructure for inter-process communication and subscription to configuration and detection events. It also initializes dictionaries to manage object and audio recording information and a cache for recording end times.
02	<code>move_files()</code>	To manage and organize video files stored in a cache directory, ensuring efficient use of storage while maintaining the integrity and relevance of the video data.
03	<code>validate_and_move_segment()</code>	To validate individual video segments and decide their fate (i.e., whether to move them to permanent storage or delete them) based on various criteria including system configuration, file integrity, and relevance to recorded events.
04	<code>segment_stats()</code>	Calculates statistics for a video segment between a start and end time for a specific camera.
05	<code>move_segment()</code>	Moves a video segment from a cache to permanent storage based on certain conditions, including manual events and segment content.
06	<code>run()</code>	Continuously checks for updated configurations, processes incoming detection data (video), and manages recording files by calling “ <code>move_files()</code> ”. It performs these tasks every 5 seconds, adjusting wait times based on execution duration, and handles any errors. The method exits cleanly when the stop event is triggered.

4.6 Event

4.6.1 Properties of class

Table 78. Properties of class - Event

No	Attribute	Description
01	id	A unique identifier for each event, serving as the primary key in the database.
02	label	A categorization or brief description of the event, indexed for faster searches.
03	sub_label	A more detailed categorization or description of the event, which can be null.
04	camera	The identifier of the camera that captured the event, indexed for faster searches.
05	start_time	Timestamps marking the beginning of the event.
06	end_time	Timestamps marking the end of the event.
07	top_score	A floating-point number representing the highest score of significance or confidence for the event.
08	score	A floating-point number representing the score of significance or confidence for the event.
09	false_positive	A boolean indicating whether the event was identified as a false positive.
10	zones	A JSON field storing information about specific zones related to the event.
11	thumbnail	A text field likely storing a path or URL to a thumbnail image for the event.
12	has_clip	A boolean indicating whether a video clip associated with the event is available.
13	has_snapshot	A boolean indicating whether a snapshot image associated with the event is available.
14	region	A JSON field storing information about the region related to the event.
15	box	A JSON field storing bounding box information for the event.
16	area	An integer field representing the area covered by the event in the camera's field of view.
17	retain_indefinitely	A boolean indicating whether the event data should be retained indefinitely.

18	ratio	A floating-point number representing some aspect ratio related to the event.
19	model_hash	A hash value representing the model used to detect or analyze the event.
20	detector_type	A string indicating the type of detector that identified the event.
21	model_type	A string indicating the type of model used for event detection or analysis.
22	data	A JSON field storing additional data related to the event, such as tracked object boxes or regions.

4.6.2 Class methods

This class doesn't have class methods.

4.7 Recordings

4.7.1 Properties of class

Table 79. Properties of class - Recordings

No	Attribute	Description
01	id	Serves as the unique identifier for each recording. It is the primary key in the database, ensuring that each recording can be uniquely identified.
02	camera	Identifies the camera that made the recording. This field is indexed to optimize queries filtering by camera ID, facilitating faster searches based on the camera source.
03	path	Stores the file path or location where the recording is saved. This field is marked as unique, ensuring that no two recordings share the same file path.
04	start_time	Record the timestamps for when the recording started, respectively. These fields are crucial for determining the duration of the recording and for querying recordings within specific time frames.
05	end_time	Record the timestamps for when the recording ended, respectively. These fields are crucial for determining the duration of the recording and for querying recordings within specific time frames.
06	duration	Represents the total length of the recording in seconds (or another unit of time), calculated as the difference between end_time and start_time.
07	motion	An optional field that may indicate the amount of motion detected in the recording, quantified in some manner

08	objects	An optional field that may represent the number of objects identified in the recording, useful for analytics or searching recordings based on object detection.
09	dBFS	An optional field indicating the decibels relative to full scale (dBFS) of the audio track within the recording, if applicable. This could be used to gauge the loudness or presence of sound.
10	segment_size	Stores the size of the recording file in megabytes (MB) by default. This information is useful for managing storage and bandwidth considerations.
11	regions	An optional field that may indicate the number of distinct regions of interest captured or analyzed in the recording. This could relate to specific areas within the camera's field of view where events or objects were detected.

4.7.2 Class methods

This class doesn't have class methods.

4.8 RecordRetainConfig

4.8.1 Properties of class

Table 80. Properties of class - RecordRetainConfig

No	Attribute	Description
01	days	This attribute sets the default retention period in days.
02	mode	The mode attribute utilizes the RetainModeEnum to define the criteria under which data is retained.

4.8.2 Class methods

This class doesn't have class methods.

4.9 EventsConfig

4.9.1 Properties of class

Table 81. Properties of class - EventsConfig

No	Attribute	Description
01	pre_capture	This integer attribute specifies the number of seconds of footage to retain before an event officially starts.
02	post_capture	This integer attribute defines the number of seconds of footage to retain after an event ends.
03	objects	This optional list of strings specifies which objects

		must be detected within the footage for an event to be considered worth saving.
04	retain	This attribute embeds a RetainConfig instance, which defines how long the captured event footage should be retained before being eligible for deletion. This allows for detailed control over data retention policies, enabling customization of how long event-related data is kept based on factors like storage capacity, privacy concerns, or legal requirements.

4.9.2 Class methods

This class doesn't have class methods.

4.10 RetainConfig

4.10.1 Properties of class

Table 82. Properties of class - RetainConfig

No	Attribute	Description
01	default	This attribute specifies the default retention period, in days, for data.
02	mode	The mode attribute utilizes the RetainModeEnum to define the retention strategy.
03	objects	This attribute is a dictionary mapping specific object types (as strings) to their respective retention periods (as floats, representing days). It allows for granular control over the retention period of data related to specific objects.

4.10.2 Class methods

This class doesn't have class methods.

4.11 RetainModeEnum

4.11.1 Properties of class

Table 83. Properties of class - RetainModeEnum

No	Attribute	Description
01	all	Represents a mode where all data or objects are retained.
02	motion	Represents a mode where only data or objects related to motion are retained. This could be used in contexts like video surveillance where only frames with motion are kept.
03	active_objects	Represents a mode where only data or objects that are considered "active" are retained. The exact

		definition of "active" would depend on the specific application context.
--	--	--

4.11.2 Class methods

This class doesn't have class methods.

4.12 ConfigSubscriber

4.12.1 Properties of class

Table 84. Properties of class - ConfigSubscriber

No	Attribute	Description
01	context	An instance of zmq.Context. This is the ZeroMQ context that manages the lifecycle of sockets. It's essential for creating and managing the communication channel.
02	socket	A ZeroMQ socket of type SUB (subscriber), created from the ZeroMQ context. This socket is used to subscribe to messages on a specified topic and receive updates published to that topic.

4.12.2 Class methods

Table 85. Class methods - ConfigSubscriber

No	Method	Description
01	<code>__init__()</code>	Initializes the ConfigSubscriber instance. It sets up the ZeroMQ context and subscriber socket, subscribes to the specified topic using <code>setsockopt_string(zmq.SUBSCRIBE, topic)</code> , and connects the socket to the address specified in <code>SOCKET_PUB_SUB</code> . This setup prepares the subscriber to receive messages published on the specified topic.
02	<code>check_for_update()</code>	Checks if there is an updated configuration available. This method allows the subscriber to periodically check for updates without being blocked.
03	<code>stop()</code>	Closes the ZeroMQ socket and destroys the ZeroMQ context, effectively stopping the subscriber and cleaning up resources. This method ensures a graceful shutdown of the subscriber, releasing the resources it has acquired.

4.13 DetectionSubscriber

4.13.1 Properties of class

Table 86. Properties of class - DetectionSubscriber

No	Attribute	Description
01	context	An instance of zmq.Context. This is the ZeroMQ context that manages the lifecycle of sockets. It's essential for creating and managing the communication channel.
02	socket	A ZeroMQ socket of type SUB (subscriber), created from the ZeroMQ context. This socket is used to subscribe to messages on a specified topic and receive updates published to that topic.

4.13.2 Class methods

Table 87. Class methods - DetectionSubscriber

No	Method	Description
01	<code>__init__()</code>	Initializes the DetectionSubscriber instance. It sets up the ZeroMQ context and subscriber socket, subscribes to the specified topic (using the value attribute of the DetectionTypeEnum instance to get the actual topic string), and connects the socket to the address specified in SOCKET_SUB. This setup prepares the subscriber to receive messages published on the specified topic.
02	<code>get_data()</code>	Attempts to receive detection data within an optional timeout period. It uses zmq.select to check if there's data available on the socket within the specified timeout. This method allows the subscriber to check for new data without being blocked indefinitely.
03	<code>stop()</code>	Closes the ZeroMQ socket and destroys the ZeroMQ context, effectively stopping the subscriber and cleaning up resources. This method ensures a graceful shutdown of the subscriber, releasing the resources it has acquired.

4.14 InterProcessRequestor

4.14.1 Properties of class

Table 88. Properties of class - InterProcessRequestor

No	Attribute	Description
01	context	An instance of zmq.Context, essential for managing ZeroMQ's operations. It's used to create and manage the lifecycle of ZeroMQ sockets.

02	socket	A ZeroMQ socket of type REQ (request). This socket is used to send requests and receive replies in a synchronous manner. It connects to an endpoint specified by SOCKET_REQ_REPLY, which is a constant likely containing the address for IPC.
----	--------	---

4.14.2 Class methods

Table 89. Class methods - InterProcessRequestor

No	Method	Description
01	<code>__init__()</code>	Initializes the InterProcessRequestor instance by setting up the ZeroMQ context and socket, and connecting the socket to the SOCKET_REQ_REPLY endpoint. This setup prepares the instance to send requests and receive replies.
02	<code>send_data()</code>	Sends data to the InterProcessCommunicator and waits for a reply.
03	<code>stop()</code>	Cleans up resources before shutting down.

4.15 SegmentInfo

4.15.1 Properties of class

Table 90. Properties of class - SegmentInfo

No	Attribute	Description
01	<code>motion_count</code>	An integer representing the number of motion events detected within the segment.
02	<code>active_object_count</code>	An integer indicating the number of active objects (e.g., people, vehicles) identified within the segment.
03	<code>region_count</code>	An integer specifying the number of distinct regions (areas of interest) within the segment where motion or activity was detected.
04	<code>average_dBFS</code>	An integer representing the average audio level of the segment in decibels relative to the full scale (dBFS), a common unit for measuring audio levels.

4.15.2 Class methods

Table 91. Class methods - SegmentInfo

No	Method	Description
01	<code>should_discard_segment()</code>	This method takes a single parameter, <code>retain_mode</code> , which is an instance of <code>RetainModeEnum</code> . The method returns a boolean value indicating whether

		the segment should be discarded based on the criteria defined by the retain_mode.
--	--	---

4.16 DatabaseConfig

4.16.1 Properties of class

Table 92. Properties of class - DatabaseConfig

No	Attribute	Description
01	path	This attribute specifies the filesystem path to the database file.

4.16.2 Class methods

This class doesn't have class methods.

4.17 VigisionBaseModel

4.17.1 Properties of class

Table 93. Properties of class - VigisionBaseModel

No	Attribute	Description
01	model_config	Define configuration options for the model

4.17.2 Class methods

This class doesn't have class methods.

4.18 ReviewSegmentMaintainer

4.18.1 Properties of class

Table 94. Properties of class - ReviewSegmentMaintainer

No	Attribute	Description
01	name	A string that identifies the thread.
02	config	An instance of VigisionConfig (presumably a configuration class specific to the application). This attribute stores the configuration settings that the maintainer needs to operate, such as paths, thresholds, or operational parameters.
03	active_review_segments	A dictionary that maps string identifiers to PendingReviewSegment objects or None. This attribute tracks the review segments that are currently active and being processed. The use of Optional[PendingReviewSegment] suggests that there can be entries in this dictionary that don't currently have an associated review segment object (i.e., they are None).

04	frame_manager	An instance of SharedMemoryFrameManager. This attribute likely manages frames (images) in shared memory, facilitating efficient inter-process communication of video data.
05	requestor	An instance of InterProcessRequestor. This attribute is used for sending data or requests between processes, enabling the ReviewSegmentMaintainer to communicate with other parts of the system, such as a database or another service that handles review segments.
06	config_subscriber	An instance of ConfigSubscriber initialized with a path ("config/record/"). This attribute subscribes to configuration updates, allowing the ReviewSegmentMaintainer to adjust its behavior based on changes in the system's configuration.
07	detection_subscriber	An instance of DetectionSubscriber initialized to subscribe to all detection types (DetectionTypeEnum.all). This attribute listens for detection events from the surveillance system, which can trigger the creation or update of review segments.
08	indefinite_events	A dictionary that tracks manual events. The keys are string identifiers, and the values are dictionaries with arbitrary data (dict[str, any]). This attribute likely stores information about manual interventions or events that don't have a predefined end time, affecting how review segments are processed or prioritized.
09	stop_event	An instance of MpEvent (presumably a multiprocessing event). This attribute is used to signal the thread to stop running, allowing for a graceful shutdown of the ReviewSegmentMaintainer.

4.18.2 Class methods

Table 95. Class methods - ReviewSegmentMaintainer

No	Method	Description
01	new_segment()	To initiate a new review segment.
02	update_segment()	To update an existing review segment with new information.
03	end_segment()	To mark a review segment as ended.
04	update_existing_segment()	To evaluate and potentially update an existing review segment based on new detections.

05	check_if_new_segment()	Determines whether a new review segment should be created based on the presence of active objects in a video frame.
06	run()	The main loop of the system that continuously checks for updates from different sources (video, audio, API/manual events) and updates the review segments accordingly.

4.19 PendingReviewSegment

4.19.1 Properties of class

Table 96. Properties of class - PendingReviewSegment

No	Attribute	Description
01	id	A unique identifier for the segment, generated by combining the frame time with a random string.
02	camera	The identifier of the camera that captured the segment.
03	start_time	The timestamp of when the segment started.
04	severity	An enum value indicating the severity of the events captured in the segment.
05	detections	A dictionary mapping detected object types to their labels.
06	sub_labels	A set of additional labels or tags associated with the segment.
07	zones	A list of zones where the detections occurred.
08	last_update	The timestamp of the last update to the segment.
09	_frame	A NumPy array representing a thumbnail image for the segment.
10	has_frame	A boolean indicating whether a thumbnail frame has been successfully generated.
11	frame_active_count	The number of active objects in the frame.
12	frame_path	The file path where the thumbnail image is stored.

4.19.2 Class methods

Table 97. Class methods - PendingReviewSegment

No	Method	Description
01	update_frame()	Updates the thumbnail image for the segment based on the provided frame and detected objects.

02	save_full_frame()	Processes the entire frame without cropping to the bounding box of detected objects. It converts the frame to color, resizes it to a thumbnail size, and saves the thumbnail image to disk.
03	get_data()	Generates and returns a dictionary containing the segment's data, including its ID, camera identifier, start and end times, severity, path to the thumbnail image, and lists of detections, objects, sub_labels, zones, and audio labels.

4.20 SharedMemoryFrameManager

4.20.1 Properties of class

Table 98. Properties of class - SharedMemoryFrameManager

No	Attribute	Description
01	shm_store	A dictionary that maps frame names to their corresponding shared_memory.SharedMemory objects. This attribute is used to keep track of all shared memory segments created or opened by the SharedMemoryFrameManager.

4.20.2 Class methods

Table 99. Class methods - SharedMemoryFrameManager

No	Method	Description
01	__init__()	Initializes the SharedMemoryFrameManager instance with an empty dictionary self.shm_store, ready to keep track of shared memory objects.
02	create()	Creates a new shared memory segment with the specified name and size.
03	get()	This method allows for efficient manipulation of frame data stored in shared memory.
04	close()	Closes the shared memory segment identified by name and removes it from self.shm_store. Closing a shared memory segment disassociates it from the process but does not delete it from the system.
05	delete()	Closes and unlinks (deletes) the shared memory segment identified by name, then removes it from self.shm_store. Unlinking a shared memory segment marks it for deletion, allowing the system to reclaim the memory once all processes have disassociated from it.

4.21 DetectionTypeEnum

4.21.1 Properties of class

Table 100. Properties of class - DetectionTypeEnum

No	Attribute	Description
01	video	Represents detections that are identified through video analysis. This indicates that the detection is related to video content, such as motion detection, object recognition, etc.

4.21.2 Class methods

This class doesn't have class methods.

4.22 TrackedObject

4.22.1 Properties of class

Table 101. Properties of class - TrackedObject

No	Attribute	Description
01	score_history	A list of detection scores for the object, used to determine its presence and relevance.
02	obj_data	General data about the object, including its label, score, and bounding box coordinates.
03	camera	References to the camera that detected the object.
04	colormap	References to the camera that detected a colormap for visualization.
05	camera_config	References to the camera that detected the camera's configuration settings.
06	frame_cache	A cache of recent frames from the camera, used for retrieving thumbnails or processing images.
07	zone_presence	Dictionaries tracking the object's presence time in defined zones.
08	zone_loitering	Dictionaries tracking the object's loitering time in defined zones.
09	current_zones	Lists tracking which zones the object is currently in and which it has entered during its detection period.
10	entered_zones	Lists tracking which zones the object is currently in and which it has entered during its detection period.
11	attributes	A dictionary of additional attributes detected for the object, such as specific features or characteristics.
12	false_positive	A flag indicating whether the object is considered a false positive detection.

13	has_clip	Flags indicating whether the object has associated video clips.
14	has_snapshot	Flags indicating whether the object has associated snapshots.
15	top_score	The highest score for the object's detection.
16	computed_score	The currently computed score for the object's detection.
17	thumbnail_data	Data for generating a thumbnail image of the object.
18	last_updated	Timestamps for the last updates to the object's data, respectively.
19	last_published	Timestamps for the last publication, respectively.
20	frame	The current frame in which the object is detected.

4.22.2 Class methods

Table 102. Class methods - TrackedObject

No	Method	Description
01	_is_false_positive()	Determines if the object is considered a false positive based on its computed score and a threshold.
02	compute_score()	Calculates the object's current score, typically using the median of its score history.
03	update()	Updates the object's state based on new data, including its score history, detection zones, attributes, and whether it's considered a false positive. It also determines if there's a significant change in its state that warrants further action.
04	to_dict()	Converts the object's current state into a dictionary, optionally including thumbnail data. This is useful for serialization or sending data over a network.
05	get_thumbnail()	Retrieves or generates a thumbnail image for the object.
06	get_clean_png()	Generates a clean PNG image of the object, without any overlays or annotations.
07	get_jpg_bytes()	Generates a JPEG image of the object, with options for including timestamps, bounding boxes, cropping, and adjusting the image quality.

4.23 SeverityEnum

4.23.1 Properties of class

Table 103. Properties of class - SeverityEnum

No	Attribute	Description
01	alert	Represents a high severity level, indicating that immediate attention is required. This could be used to categorize or trigger alerts that signify critical conditions or issues that need urgent intervention.
02	detection	Represents a lower severity level than alert, used for conditions that are noteworthy but not immediately critical. This could be used to log or notify conditions that should be monitored or reviewed but do not require immediate action.

4.23.2 Class methods

This class doesn't have class methods.

4.24 ReviewConfig

4.24.1 Properties of class

Table 104. Properties of class - ReviewConfig

No	Attribute	Description
01	alerts	This attribute holds the configuration for how alerts are generated and managed within the system.
02	detections	This attribute manages the configuration for detections, specifying which objects should be detected and under what conditions an event is saved as a detection.

4.24.2 Class methods

This class doesn't have class methods.

4.25 AlertsConfig

4.25.1 Properties of class

Table 105. Properties of class - AlertsConfig

No	Attribute	Description
01	labels	A list of strings representing the object labels that, when detected, can trigger an alert.
02	required_zones	Specifies the zones that objects must enter to trigger an alert. This can be a single zone as a string or a list of zones. The use of zones allows for more precise alerting, focusing on specific areas of interest within the video or image feed.

4.25.2 Class methods

Table 106. Class methods - AlertsConfig

No	Method	Description
01	validate_required_zones()	Ensure that the required_zones attribute is always in list format

4.26 DetectionsConfig

4.26.1 Properties of class

Table 107. Properties of class - DetectionsConfig

No	Attribute	Description
01	labels	This attribute allows for targeted detection, focusing on objects of interest.
02	required_zones	Define required zones, allowing the system can filter detections based on location, ensuring that only objects entering specified areas are recorded as detections.

4.26.2 Class methods

Table 108. Class methods - DetectionsConfig

No	Method	Description
01	validate_required_zones()	This method ensures that the required_zones attribute is always in a list format, facilitating further processing.

4.27 ManualEventState

4.27.1 Properties of class

Table 109. Properties of class - ManualEventState

No	Attribute	Description
01	complete	This state indicates that a manual event has been fully processed or completed. It's used when an event has gone through all necessary stages from initiation to conclusion without needing further action.
02	start	This state signifies the beginning or initiation of a manual event. It's used when an event is first created or triggered, marking its initial phase.
03	end	This state marks the termination or conclusion of a manual event. It's used when an event has reached

		its final stage, but it does not necessarily imply that the event has been fully processed like the complete state does.
--	--	--

4.27.2 Class methods

This class doesn't have class methods.

4.28 ReviewSegment

4.28.1 Properties of class

Table 110. Properties of class - ReviewSegment

No	Attribute	Description
01	id	Acts as the unique identifier for each review segment. Being the primary key, it ensures that each segment can be uniquely identified within the database.
02	camera	Identifies the camera that captured the segment. This field is indexed to optimize queries by camera ID, facilitating easier retrieval of segments based on the camera source.
03	start_time	These fields record the timestamps for when the segment starts, respectively. They are crucial for defining the exact duration of the segment under review.
04	end_time	These fields record the timestamps for when the segment ends, respectively. They are crucial for defining the exact duration of the segment under review.
05	has Been Reviewed	A boolean field indicating whether the segment has been reviewed. This field allows for tracking the review status of segments.
06	severity	Describes the severity or importance of the segment, with possible values like 'alert', 'detection', or 'significant_motion'. This classification can help prioritize review efforts based on the nature of the detected event.
07	thumb_path	Stores the file path or location for a thumbnail image representing the segment. This field is unique, ensuring that each segment has a distinct thumbnail for quick reference or identification.
08	data	A JSON field that holds additional data about the detection, such as a list of labels (e.g., types of objects detected), zones (specific areas within the camera's field of view), or areas of significant motion. The use of a JSON field allows for storing a

		flexible and structured set of data that can vary widely between segments.
--	--	--

4.28.2 Class methods

This class doesn't have class methods.

4.29 ModelConfig

4.29.1 Properties of class

Table 111. Properties of class - ModelConfig

No	Attribute	Description
01	path	Optional path to the custom object detection model.
02	labelmap_path	Optional path to the label map for the custom object detector.
03	width	Dimensions (width) for the object detection model input.
04	height	Dimensions (height) for the object detection model input.
05	labelmap	A dictionary mapping integer labels to string representations, allowing for label customization.
06	input_tensor	Enum specifying the model input tensor shape (e.g., NHWC or NCHW).
07	input_pixel_format	Enum specifying the model input pixel color format (e.g., RGB or BGR).
08	model_type	Enum specifying the type of object detection model (e.g., SSD, Faster R-CNN).
09	_merged_labelmap	A private attribute that combines the default label map with any customizations provided.
10	_colormap	A private attribute storing a dictionary mapping integer labels to color tuples for visualization.
11	_model_hash	A private attribute storing a hash of the model file, used for versioning or caching.

4.29.2 Class methods

Table 112. Class methods - ModelConfig

No	Method	Description
01	merged_labelmap()	A property that returns the merged label map.
02	colormap()	A property that returns the colormap dictionary.

03	model_hash()	A property that returns the hash of the model file.
04	__init__()	Constructor that initializes the model configuration with the provided parameters and merges the default label map with any customizations.
05	check_and_load_plus_model()	Checks if the model path starts with "plus://" and, if so, downloads the model and its information from a specified API, updating the model configuration accordingly. It also validates if the model supports a specified detector type.
06	compute_model_hash()	Computes and updates the hash of the model file, useful for detecting changes to the model file.
07	create_colormap()	Generates a colormap for a set of enabled labels, useful for visualizing detection results.

4.30 BaseDetectorConfig

4.30.1 Properties of class

Table 113. Properties of class - BaseDetectorConfig

No	Attribute	Description
01	type	A string that specifies the type of detector (e.g., "cpu", "gpu"). This field is essential for determining the processing unit or method the detector will use. It's marked to be overridden in subclasses, indicating that different detector configurations will specify their own default types.
02	model	An optional field that can hold a ModelConfig object. ModelConfig is presumably another class (defined elsewhere) that contains configuration details specific to the object detection model being used by the detector (e.g., model path, input size, label map). This allows for flexibility in configuring the model specifics for each detector.
03	model_config	The intended use appears to be for additional, arbitrary model configuration settings not covered by the ModelConfig class.

4.30.2 Class methods

This class doesn't have class methods.

4.31 ObjectDetectProcess

4.31.1 Properties of class

Table 114. Properties of class - ObjectDetectProcess

No	Attribute	Description

01	name	A string representing the name of the detection process. This can be used for identification and logging purposes.
02	detection_queue	A multiprocessing queue through which detection tasks (e.g., images or video frames) are sent to the detection process.
03	out_events	A multiprocessing queue or similar structure used to receive detection results or events from the detection process.
04	avg_inference_speed	A multiprocessing Value used to store and share the average inference speed (e.g., frames per second) of the detection process.
05	detection_start	A multiprocessing Value used to track the start time of the detection process, potentially for performance monitoring or synchronization purposes.
06	detect_process	Holds the reference to the multiprocessing Process object that runs the detection task.
07	detector_config	Configuration parameters for the detector, which could include model details, threshold values, etc.

4.31.2 Class methods

Table 115. Class methods - ObjectDetectProcess

No	Method	Description
01	<code>__init__()</code>	The constructor initializes the class with the provided parameters, sets up shared values for tracking performance, and starts (or restarts) the detection process.
02	<code>start_or_restart()</code>	Starts or restarts the detection process. If an existing process is alive, it stops it before starting a new one. This method initializes a new multiprocessing Process with the target function <code>run_detector</code> and the necessary arguments, including queues for input and output, shared values for performance tracking, and the detector configuration.
03	<code>stop()</code>	Gracefully terminates the detection process. It first attempts to terminate the process and waits for it to exit. If the process does not exit within a specified timeout, it forcefully kills the process. This method ensures that resources are properly released and that the process does not hang indefinitely.

4.32 LocalObjectDetector

4.32.1 Properties of class

Table 116. Properties of class - LocalObjectDetector

No	Attribute	Description
01	fps	An instance of EventsPerSecond, used to track the frame processing speed of the detector.
02	labels	A dictionary mapping object IDs to their labels. If no labels are provided during initialization, it defaults to an empty dictionary.
03	input_transform	Stores a transformation function for the input tensor based on the detector_config.model.input_tensor. This transformation is applied to the input tensor before detection.
04	detect_api	Responsible for the actual detection process.

4.32.2 Class methods

Table 117. Class methods - LocalObjectDetector

No	Method	Description
01	__init__	The constructor initializes the object detector with optional detector_config and labels. It sets up label mapping, input transformation, and the detection API.
02	detect()	This method takes an input tensor (and an optional threshold) to perform object detection. It processes raw detections, filters them based on the threshold and label validity, and returns a list of detections. Each detection includes the label, confidence score, and bounding box coordinates. It also updates the fps attribute to track performance.
03	detect_raw()	A helper method that applies the input_transform to the tensor_input if necessary, then uses the detect_api to perform raw detection on the transformed input. This method is called by detect to get raw detection results.

4.33 TrackedObjectProcessor

4.33.1 Properties of class

Table 118. Properties of class - TrackedObjectProcessor

No	Attribute	Description
01	name	A string identifier for the thread, set to "detected_frames_processor". This name is useful for

		debugging or logging purposes, helping to identify the thread's role within the system.
02	config	An instance of VigisionConfig, which likely contains configuration settings for the vision system. This could include parameters for detection thresholds, camera configurations, or other system-wide settings.
03	dispatcher	An instance of Dispatcher, which is probably responsible for managing and dispatching events or messages between different components of the system.
04	tracked_objects_queue	A queue (the exact type is not specified) used for holding tracked objects that need to be processed. This queue acts as a buffer between the detection component and the processing component, allowing for asynchronous processing.
05	stop_event	An instance of MpEvent (presumably a multiprocessing event), used to signal when the thread should stop running. This allows for a controlled shutdown of the thread.
06	camera_states	A dictionary mapping camera identifiers (strings) to CameraState objects. This attribute likely holds the current state or status of each camera being monitored, such as active, idle, or error states.
07	frame_manager	An instance of SharedMemoryFrameManager, which is probably used for managing access to video frames stored in shared memory. This allows for efficient inter-process communication and frame sharing.
08	last_motion_detected	A dictionary mapping camera identifiers to timestamps (floats) of the last motion detected. This is useful for tracking activity and possibly for optimizing processing or detection algorithms based on motion.
09	requestor	An instance of InterProcessRequestor, likely used for making requests to other processes or services within the system. This could be used for fetching data, sending commands, or other inter-process interactions.
10	detection_publisher	An instance of DetectionPublisher initialized with DetectionTypeEnum.video, indicating it's used for publishing detection events or data specifically related to video analysis.
11	event_sender	An instance of EventUpdatePublisher, which is probably used for publishing updates about events detected by the system, such as object detections, motion events, or system status changes.
12	event_end_subscriber	An instance of EventEndSubscriber, which likely subscribes to notifications or messages indicating the end of an event, such as the conclusion of a motion event or the

		disappearance of a tracked object.
13	camera_activity	A dictionary to hold activity data or statistics for each camera. The exact structure isn't specified, but it could include information on detected objects, motion events, or other relevant metrics.

4.33.2 Class methods

Table 119. Class methods - TrackedObjectProcessor

No	Method	Description
01	should_save_snapshot()	Determines whether a snapshot of a tracked object should be saved. It returns False if the object is marked as a false positive, if snapshot capturing is disabled in the configuration, if the object hasn't changed position, or if the object hasn't entered any required zones specified in the configuration. Otherwise, it returns True.
02	should_retain_recording()	Decides whether a recording involving a tracked object should be retained. It checks similar conditions as should_save_snapshot, including whether the object is a false positive, if recording is enabled, and if the object has changed position. Additionally, it checks if the object has entered required zones and if the object matches required object labels for recording. If any condition is not met, it returns False; otherwise, True.
03	get_best()	Retrieves the best snapshot of a specified object (identified by label) from a particular camera. It returns a dictionary containing the thumbnail data and the frame from the camera's frame cache based on the timestamp of the thumbnail.
04	get_current_frame()	Returns the current frame for a specified camera. If the camera is a special "birdseye" view, it fetches the frame with specific dimensions from the frame manager; otherwise, it gets the current frame from the camera's state.
05	get_current_frame_time()	Returns the timestamp of the latest frame for a given camera, indicating when the frame was captured or processed.
06	run()	The main loop of the system, which continuously processes tracked objects from a queue until a stop event is signaled.

4.34 Dispatcher

4.34.1 Properties of class

Table 120. Properties of class - Dispatcher

No	Attribute	Description
01	config	An instance of VigisionConfig. This attribute stores the configuration settings for Vigision, likely including parameters for operation, thresholds, and preferences that affect how Vigision behaves or processes data.
02	config_updater	An instance of ConfigPublisher. This is likely a utility or service responsible for publishing updates to the Vigision configuration, possibly to other components or services that need to be aware of configuration changes.
03	comms	A list of Communicator instances. These are objects that adhere to the Communicator interface defined elsewhere, capable of publishing messages to and subscribing to messages from a communication channel. This list represents the different communication channels or protocols the Dispatcher can use to send or receive messages.
04	_camera_settings_handlers	A private dictionary mapping strings to callable functions. Each key represents a command related to camera settings (e.g., "audio", "detect", "motion"), and each value is a method of the Dispatcher class designed to handle that specific command. This mapping allows the Dispatcher to dynamically respond to various commands by invoking the corresponding handler method.
05	camera_activity	An initially empty dictionary. This attribute is likely intended to track the state or activity of cameras, such as which cameras are currently active, recording, or tracking motion. The specific use of this dictionary would depend on further implementation details not provided in the snippet.

4.34.2 Class methods

Table 121. Class methods - Dispatcher

No	Method	Description
01	_receive()	Handles incoming messages from communicators based on the topic and payload.
02	publish()	Publishes a message to all communicators.
03	stop()	Stops all communicators.

04	<code>_on_detect_command()</code>	Handles commands related to enabling or disabling detection for a specific camera.
05	<code>_on_motion_command()</code>	Handles commands related to enabling or disabling motion detection for a specific camera.
06	<code>_on_motion_improve_contrast_command()</code>	To enable or disable the "improve contrast" feature for motion detection on a specified camera based on the received command.
07	<code>_on_motion_contour_area_command()</code>	Adjusts the contour area setting for motion detection on a specified camera.
08	<code>_on_motion_threshold_command()</code>	Adjusts the motion threshold setting for a specified camera.
09	<code>_on_recordings_command()</code>	Controls the recording functionality of a specified camera based on the received command.
10	<code>_on_snapshots_command()</code>	Manages the snapshots functionality for a specified camera based on the received command.

4.35 CameraState

4.35.1 Properties of class

Table 122. Properties of class - CameraState

No	Attribute	Description
01	<code>name</code>	A string representing the unique identifier or name of the camera. This is used to distinguish between different cameras in the system.
02	<code>config</code>	An instance of VigisionConfig, which likely contains global configuration settings for the surveillance system, including settings for all cameras.
03	<code>camera_config</code>	A subset of config, specifically holding the configuration settings for this camera, extracted using the camera's name. This includes parameters like resolution, frame rate, detection zones, etc.
04	<code>frame_manager</code>	An instance of SharedMemoryFrameManager, which manages the frames captured by the camera, likely allowing for efficient sharing and processing of frame data across different components of the system.
05	<code>best_objects</code>	A dictionary mapping object identifiers to TrackedObject instances, representing the most relevant or "best" objects detected by the camera. This could be based on criteria like detection confidence, size, or relevance to the current monitoring goals.

06	object_counts	A defaultdict initialized with int, used to keep count of detected objects, possibly categorized by type or other criteria.
07	tracked_objects	A dictionary mapping unique identifiers to TrackedObject instances for all objects currently being tracked by the camera. This allows for continuous monitoring and state updating of each object.
08	frame_cache	A dictionary or similar structure used to cache recent frames or frame metadata, facilitating quick access for processing, analysis, or display purposes.
09	zone_objects	A defaultdict initialized with list, used to track objects within specific predefined zones in the camera's field of view. This can be used for applications like zone-specific counting or behavior analysis.
10	_current_frame	A NumPy array initialized to zeros with the shape specified by camera_config.frame_shape_yuv, representing the current frame in YUV color space. This serves as a placeholder for the latest frame captured by the camera.
11	current_frame_lock	An instance of threading.Lock, used to ensure thread-safe access to the current frame, especially important in a multi-threaded environment where frames are processed in parallel.
12	current_frame_time	A floating-point number representing the timestamp of the current frame. This is crucial for time-based processing and synchronization of video data.
13	motion_boxes	A list that likely stores bounding boxes or regions where motion has been detected in the current or recent frames. This can be used for motion-based triggering of alerts or further analysis.
14	regions	A list that could store predefined regions of interest within the camera's field of view, for focused analysis or object detection.
15	previous_frame_id	An identifier for the previous frame processed by the system. This can be used for frame-to-frame comparisons, motion detection, or tracking object movement.
16	callbacks	A defaultdict initialized with list, used to store callback functions that should be triggered on specific events, such as new object detection, motion detection, or system alerts.

4.35.2 Class methods

Table 123. Class methods - CameraState

No	Method	Description
01	get_current_frame()	Retrieves the current frame from a camera, applies various optional visual enhancements based on the provided draw_options, and returns the modified frame. These enhancements can include drawing bounding boxes around tracked objects, highlighting motion-detected areas, delineating regions or zones of interest, and overlaying timestamps. The method ensures thread-safe access to the frame data and performs several image processing steps, such as color space conversion and conditional drawing operations based on the camera's configuration and the state of tracked objects and zones.
02	finished()	Removes an object identified by obj_id from the tracked_objects dictionary, effectively marking it as no longer being tracked.
03	on()	Registers a callback function to be invoked when a specific event type occurs. It adds the provided callback function to a list of callbacks associated with the event_type in the callbacks dictionary. This allows for event-driven programming, where actions can be triggered in response to specific events.
04	update()	The update method in a camera tracking system performs several key functions to manage and update the state of tracked objects based on new detections, motion boxes, and defined regions.

4.36 DetectionPublisher

4.36.1 Properties of class

Table 124. Properties of class - DetectionPublisher

No	Attribute	Description
01	topic	An instance of DetectionTypeEnum, which presumably is an enumeration defining different types of detections (e.g., video, audio). This attribute specifies the topic under which the detection data will be published.
02	context	An instance of zmq.Context. This is the ZeroMQ context that manages the lifecycle of sockets within the ZeroMQ framework. It's essential for creating and managing the communication channel.
03	socket	A ZeroMQ socket of type PUB (publisher), created from the ZeroMQ context. This socket is used to publish detection data to all subscribers interested in that data.

4.36.2 Class methods

Table 125. Class methods - DetectionPublisher

No	Method	Description
01	<code>__init__()</code>	Initializes the DetectionPublisher instance by setting the topic for detection data, setting up the ZeroMQ context, creating a publisher socket, and connecting this socket to the address specified in SOCKET_PUB. This setup prepares the publisher to send detection data on the specified topic.
02	<code>send_data()</code>	Publishes detection data. This method allows for the transmission of complex data structures directly, without needing to serialize them manually. This is a one-way communication from the publisher to the subscribers.
03	<code>stop()</code>	Signals the DetectionPublisher to stop by closing the ZeroMQ socket and destroying the ZeroMQ context. This method ensures a graceful shutdown of the publisher, releasing the resources it has acquired.

4.37 EventUpdatePublisher

4.37.1 Properties of class

Table 126. Properties of class - EventUpdatePublisher

No	Attribute	Description
01	<code>context</code>	An instance of <code>zmq.Context</code> , which is the main entry point of ZeroMQ library. It manages sockets and facilitates the establishment of connections. Each instance can handle many sockets.
02	<code>socket</code>	A ZeroMQ socket of type PUSH. This socket is used to send messages to a PULL socket, following the PUSH/PULL pattern. This pattern allows for load balancing messages across multiple receivers.

4.37.2 Class methods

Table 127. Class methods - EventUpdatePublisher

No	Method	Description
01	<code>__init__()</code>	Initializes the EventUpdatePublisher instance.
02	<code>publish()</code>	Publishes an event to the connected endpoint.
03	<code>stop()</code>	Cleans up resources before shutting down.

4.38 EventEndSubscriber

4.38.1 Properties of class

Table 128. Properties of class - EventEndSubscriber

No	Attribute	Description
01	context	An instance of zmq.Context, essential for managing ZeroMQ operations. It facilitates the creation and management of sockets.
02	socket	A ZeroMQ socket of type PULL. This socket is configured to receive messages from a corresponding PUSH socket. It binds to an endpoint specified by SOCKET_PUSH_PULL_END, which is a predefined constant likely containing the address of the publisher.

4.38.2 Class methods

Table 129. Class methods - EventEndSubscriber

No	Method	Description
01	<code>__init__()</code>	Initializes the EventEndSubscriber instance.
02	<code>check_for_update()</code>	Checks for new updates about ended events within a specified timeout period.
03	<code>stop()</code>	Cleans up resources before shutting down.

4.39 EventTypeEnum

4.39.1 Properties of class

Table 130. Properties of class - EventTypeEnum

No	Attribute	Description
01	api	Represents an event type related to API interactions. This could imply events that are triggered by API calls, such as requests to a web service or internal API endpoints.
02	tracked_object	Represents an event type related to tracking objects. This indicates events that are generated in the context of object tracking, such as the detection, movement, or status change of objects within a system.

4.39.2 Class methods

This class doesn't have class methods.

4.40 EventStateEnum

4.40.1 Properties of class

Table 131. Properties of class - EventStateEnum

No	Attribute	Description
01	start	Represents the initial state of an event. This could be used to indicate the beginning or initiation of an event.
02	update	Represents an intermediate state of an event, where the event is ongoing but has changed in some way. This could be used to indicate modifications or progress in the event's lifecycle.
03	end	Represents the final state of an event. This could be used to indicate the conclusion or termination of an event.

4.40.2 Class methods

This class doesn't have class methods.

4.41 SnapshotsConfig

4.41.1 Properties of class

Table 132. Properties of class - SnapshotsConfig

No	Attribute	Description
01	enabled	Allows for the dynamic enabling or disabling of snapshot capturing.
02	clean_copy	Specifies that a clean copy of the snapshot image should be created. This implies no overlays (like timestamps or bounding boxes) are added to the snapshot.
03	timestamp	A boolean indicating whether to add a timestamp overlay to the snapshot. This can be useful for tracking when an event occurred.
04	bounding_box	Adds a bounding box overlay to the snapshot, typically around detected objects.
05	crop	Controls whether the snapshot should be cropped to the detected object.
06	required_zones	Allows for snapshots to be conditionally saved based on the presence of objects in specified areas of the image.
07	height	Specify the height of the snapshot image.
08	retain	Manage storage by ensuring that only relevant

		snapshots are retained.
09	quality	Specify the quality of the encoded JPEG snapshot.

4.41.2 Class methods

This class doesn't have class methods.

4.42 MotionConfig

4.42.1 Properties of class

Table 133. Properties of class - MotionConfig

No	Attribute	Description
01	enabled	A boolean indicating whether motion detection is enabled on all cameras.
02	threshold	An integer setting the sensitivity of motion detection
03	lightning_threshold	Specify the sensitivity for detecting sudden changes in lighting, which could indicate motion
04	improve_contrast	Enhances the contrast of the video feed to potentially improve motion detection accuracy
05	contour_area	Define the minimum area (in pixels) that a moving object must cover to be considered motion. This helps filter out minor movements.
06	delta_alpha	Influence how motion detection adapts to changes in the video feed over time, affecting the sensitivity of the detection algorithm.
07	frame_alpha	Adjust the weight of new frames in the motion detection algorithm
08	frame_height	Specify the height (in pixels) to which video frames are resized for motion detection processing.
09	mask	Represent the coordinates of polygons that define areas to be masked or specifically monitored for motion.
11	enabled_in_config	An optional boolean used to track the original state of motion detection enablement, allowing the system to remember if motion detection was initially enabled or disabled.
12	raw_mask	A string or list of strings intended to store the raw mask data.

4.42.2 Class methods

Table 134. Class methods - MotionConfig

No	Method	Description
01	serialize_mask()	This method is responsible for serializing the mask field. When the class instance is being serialized to JSON and the mask field is included, this method returns the value of self.raw_mask instead of the actual mask field's value. This approach allows for the use of raw mask data (potentially more detailed or in a different format) for serialization purposes.
02	serialize_raw_mask()	This method handles the serialization of the raw_mask field. When the class instance is serialized to JSON and the raw_mask field is included, this method ensures that None is returned instead of the actual raw_mask field's value. Essentially, this prevents the raw_mask field from being included in the serialized output, likely because it's meant for internal use or because its inclusion in the serialized data is not desired.

4.43 ObjectConfig

4.43.1 Properties of class

Table 135. Properties of class - ObjectConfig

No	Attribute	Description
01	track	Allows the system to focus on relevant objects, ignoring others that do not match the criteria.
02	filters	Allows for detailed configuration of filters applied to each tracked object. Filters can be used to refine tracking based on specific criteria, such as object size, speed, or color.
03	mask	Define areas within the frame where objects should be tracked or ignored. This can be useful in scenarios where tracking needs to be limited to specific regions of interest or excluded from areas where detections are likely to be false positives.

4.43.2 Class methods

This class doesn't have class methods.

4.44 PreviewRecorder

4.44.1 Properties of class

Table 136. Properties of class - PreviewRecorder

No	Attribute	Description
01	config	Stores the camera configuration (CameraConfig object) which includes settings like detection area dimensions, preview quality, etc.
02	start_time	The timestamp of the first frame in the current segment being processed.
03	last_output_time	The timestamp of the last frame that was processed and included in the preview.
04	output_frames	A list of timestamps for frames that have been processed and included in the preview.
05	out_width	The dimensions for the output preview frames, calculated based on the camera's detection area to maintain aspect ratio.
06	out_height	The dimensions for the output preview frames, calculated based on the camera's detection area to maintain aspect ratio.
07	requestor	An instance of InterProcessRequestor for communication, likely used for signalling when a preview clip is ready or for other inter-process communication.
08	segment_end	A timestamp indicating when the current segment (or hour) ends, used to determine when to start a new preview clip.

4.44.2 Class methods

Table 137. Class methods - PreviewRecorder

No	Method	Description
01	should_write_frame()	Determines whether a given frame should be included in the preview. It considers factors like the presence of active objects (e.g., cars), motion within the frame, and ensuring a minimum frame rate is maintained.
02	write_frame_to_cache()	Processes and saves a frame to the cache directory. This includes resizing the frame to the output dimensions and converting it from YUV to BGR color space before saving it as a WEBP image.
03	write_data()	The main method for processing frames. It decides whether to write the current frame to cache based on

		the logic in <code>should_write_frame</code> . It also handles the creation of new preview clips at the end of each segment (hour) by invoking <code>FFMpegConverter</code> to convert the cached frames into a preview clip.
04	<code>stop()</code>	Stops the requestor for clean-up or shutdown purposes.

4.45 RecordQualityEnum

4.45.1 Properties of class

Table 138. Properties of class - `RecordQualityEnum`

No	Attribute	Description
01	<code>very_low</code>	Represents the lowest recording quality level.
02	<code>low</code>	Represents a low but not the lowest recording quality level.
03	<code>medium</code>	Represents a medium or standard recording quality level.
04	<code>high</code>	Represents a high recording quality level.
05	<code>very_high</code>	Represents the highest recording quality level.

4.45.2 Class methods

This class doesn't have class methods.

4.46 PreviewFFMpegConverter

4.46.1 Properties of class

Table 139. Properties of class - `PreviewFFMpegConverter`

No	Attribute	Description
01	<code>name</code>	A string that uniquely identifies the converter thread, based on the camera's name and its role as a preview converter.
02	<code>config</code>	An instance of <code>CameraConfig</code> , holding configuration details for the camera, such as name, recording settings, and FFmpeg hardware acceleration arguments.
03	<code>frame_times</code>	A list of timestamps (floats) for each frame to be included in the preview. These timestamps are used to calculate the duration of each frame in the video.
04	<code>requestor</code>	An instance of <code>InterProcessRequestor</code> , used to communicate with other processes, for example, to insert metadata about the generated preview into a database.

05	path	The file path where the generated MP4 preview will be saved. It is constructed from the camera's name and the range of frame times.
06	ffmpeg_cmd	The FFmpeg command constructed to convert the list of frames into a VFR MP4. It includes various parameters like hardware acceleration arguments, input and output settings, and the path to save the file.

4.46.2 Class methods

Table 140. Class methods - PreviewFFMpegConverter

No	Method	Description
01	run()	This method enables efficient, automated generation of preview videos from still images in a background thread, minimizing impact on the main application's performance.

4.47 RecordPreviewConfig

4.47.1 Properties of class

Table 141. Properties of class - RecordPreviewConfig

No	Attribute	Description
01	quality	This attribute specifies the quality level of recording previews using predefined values from the RecordQualityEnum, affecting how recording previews are rendered or processed, potentially affecting file size, clarity, and the overall user experience when viewing these previews.

4.47.2 Class methods

This class doesn't have class methods.

4.48 JsmpegCamera

4.48.1 Properties of class

Table 142. Properties of class - JsmpegCamera

No	Attribute	Description
01	config	An instance of CameraConfig, which holds configuration details for the camera, such as frame dimensions, quality, and frames per second (fps).
02	input	A queue.Queue object used to hold the video frames before they are processed by the converter. The maximum size of the queue is set to the fps value from the camera's configuration, aiming to balance memory usage and frame rate.
03	converter	An instance of FFmpegConverter, responsible for converting the raw video frames into a format suitable for streaming. It takes several configuration parameters, including the camera name, input queue, stop event, frame dimensions, and quality settings.
04	broadcaster	An instance of BroadcastThread, which handles the broadcasting of the converted video frames to connected WebSocket clients. It is initialized with the camera name, the converter, the WebSocket server, and a stop event.

4.48.2 Class methods

Table 143. Class methods - JsmpegCamera

No	Method	Description
01	<code>__init__()</code>	The constructor initializes the camera with its configuration, input queue, converter, and broadcaster. It calculates the width of the video frames based on the configured height and aspect ratio, then starts both the converter and broadcaster threads.
02	<code>write_frame()</code>	This method attempts to add a new frame (in bytes) to the input queue for processing. If the queue is full (indicating that the processing is slower than the frame capture rate), the frame is dropped to avoid blocking.
03	<code>stop()</code>	Gracefully stops the streaming process by joining both the converter and broadcaster threads, ensuring that all processing is completed before the application is closed.

4.49 FallDetectConfig

4.49.1 Properties of class

Table 144. Properties of class - FallDetectConfig

No	Attribute	Description
01	enabled	This is a boolean attribute that indicates whether the fall detection feature is active or not.

4.49.2 Class methods

This class doesn't have class methods.

4.50 CameraMetricsTypes

4.50.1 Properties of class

Table 145. Properties of class - CameraMetricsTypes

No	Attribute	Description
01	camera_fps	A Synchronized object representing the frames per second (FPS) that the camera is capturing. This metric is crucial for understanding the camera's performance and ensuring it's operating at the desired frame rate.
02	capture_process	An Optional[Process] indicating the subprocess (if any) responsible for capturing video from the camera. This could be None if no separate process is used for capture.
03	detection_fps	A Synchronized object representing the FPS at which object detection or other analysis is performed on the video frames. This helps in monitoring the performance of the detection algorithm.
04	detection_frame	A Synchronized object that likely represents the current frame being processed for detection, ensuring thread-safe access to this frame across different parts of the system.
05	ffmpeg_pid	A Synchronized object storing the process ID (PID) of an FFmpeg process, if used for processing or streaming video data. This allows for monitoring and potentially controlling the FFmpeg process.
06	frame_queue	A Queue object used for storing frames that are awaiting processing. This queue facilitates communication between the capture process and the processing/detection algorithms.
07	process	An Optional[Process] that may represent a general or specific processing task related to the camera's video stream, distinct from the capture process.

08	process_fps	A Synchronized object indicating the FPS at which the general processing (not specifically detection) is occurring on the video frames.
09	read_start	A Synchronized object that could represent the timestamp or condition indicating when frame reading or processing started, useful for timing and performance metrics.
10	skipped_fps	A Synchronized object tracking the FPS of frames that are skipped or dropped, which is important for diagnosing performance issues or tuning the processing pipeline.

4.50.2 Class methods

This class doesn't have class methods.

4.51 FilterConfig

4.51.1 Properties of class

Table 146. Properties of class - *FilterConfig*

No	Attribute	Description
01	min_area	Sets the minimum area (in pixels) of the bounding box surrounding a detected object for it to be counted.
02	max_area	Establishes the maximum area (in pixels) of the bounding box for an object to be considered.
03	min_ratio	Specifies the minimum ratio of width to height (or height to width) of the bounding box for an object to be counted.
04	max_ratio	Defines the maximum acceptable ratio of width to height for the bounding box of detected objects.
05	threshold	Sets the average detection confidence threshold for an object to be counted.
06	min_score	Determines the minimum detection confidence score for an object to be counted.
07	mask	Allows specifying one or more polygon masks that define the detection area within the video or image stream.
08	raw_mask	Holds the raw representation of the mask(s) used for detection.

4.51.2 Class methods

Table 147. Class methods - FilterConfig

No	Method	Description
01	serialize_mask()	This method is responsible for serializing the mask field when the object is converted to JSON.
02	serialize_raw_mask()	This method handles the serialization of the raw_mask field when the object is serialized to JSON.

4.52 ImprovedMotionDetector

4.52.1 Properties of class

Table 148. Properties of class - ImprovedMotionDetector

No	Attribute	Description
01	name	A string identifier for the motion detector instance.
02	config	An instance of MotionConfig, holding configuration settings for motion detection.
03	frame_shape	The shape of the frames being processed.
04	resize_factor	A factor used to resize frames based on the configuration's frame height.
05	motion_frame_size	The target size for frames after resizing, used for motion detection.
06	avg_frame	A floating-point array representing the average frame for background subtraction.
07	motion_frame_count	A counter for frames with detected motion.
08	frame_counter	A general counter for processed frames.
09	mask	A mask applied to frames, likely for ignoring certain areas in motion detection.
10	save_images	A boolean indicating whether to save processed frames for debugging.
11	calibrating	A boolean indicating whether the detector is in calibration mode.
12	blur_radius	The radius for the Gaussian blur applied to frames.
13	interpolation	The interpolation method used for resizing frames.
14	contrast_values	An array for tracking contrast values for dynamic contrast adjustment.

15	contrast_values_index	An index for managing the contrast_values array.
16	config_subscriber	An instance for subscribing to configuration updates.

4.52.2 Class methods

Table 149. Class methods - *ImprovedMotionDetector*

No	Method	Description
01	is_calibrating()	Returns whether the detector is in calibration mode.
02	detect()	Processes a single frame to detect motion.
03	stop()	Stops the motion detector, including any configuration subscription.

4.53 DetectConfig

4.53.1 Properties of class

Table 150. Properties of class - *DetectConfig*

No	Attribute	Description
01	height	Specifies the height of the video stream for detection purposes. This can be used to resize or crop the input stream to a specific height.
02	width	Specifies the width of the video stream for detection purposes, similar to height, for resizing or cropping the input.
03	fps	Defines the number of frames per second (fps) that the detection algorithm should process. This can help manage computational load and real-time processing requirements.
04	enabled	A flag to enable or disable detection functionality. This allows for easy toggling of the detection process without altering the configuration.
05	min_initialized	Sets the minimum number of consecutive detections (hits) required for an object to be considered as initialized by the tracker. This helps in filtering out false positives.
06	max_disappeared	Defines the maximum number of frames an object can be missing (not detected) before the detection of that object is considered to have ended.
07	stationary	Holds configuration settings for detecting and handling stationary objects within the video stream. This includes settings like detection intervals,

		thresholds for stationary behavior, and maximum frame counts for stationary objects.
08	annotation_offset	Specifies a time offset (in milliseconds) for annotations related to detected objects. This can be used to synchronize annotations with the video stream, especially if there are delays in processing or if the annotations need to be aligned with external data sources.

4.53.2 Class methods

This class doesn't have class methods.

4.54 RemoteObjectDetector

4.54.1 Properties of class

Table 151. Properties of class - RemoteObjectDetector

No	Attribute	Description
01	labels	A dictionary mapping object IDs to their corresponding labels.
02	name	A unique identifier for the detector instance, used for naming shared memory segments.
03	fps	An instance of EventsPerSecond, used to measure the rate at which detections are processed.
04	detection_queue	A multiprocessing queue used to submit detection tasks to a worker or processing thread.
05	event	A synchronization event used to signal when a detection task is ready or completed.
06	stop_event	Another synchronization event used to signal the detector to stop processing.
07	shm	A shared memory segment used to share input data (e.g., images) between processes.
08	np_shm	A NumPy ndarray mapped to the shm shared memory, representing the input data in a structured format.
09	out_shm	A shared memory segment used to share detection results between processes.
10	out_np_shm	A NumPy ndarray mapped to the out_shm shared memory, used to store detection results.

4.54.2 Class methods

Table 152. Class methods - RemoteObjectDetector

No	Method	Description
01	<code>__init__()</code>	Initializes the detector with the necessary configuration, including shared memory for input and output data. It sets up the shared memory segments based on the provided <code>model_config</code> dimensions for the input and a fixed size for the output.
02	<code>detect()</code>	Performs object detection on the given <code>tensor_input</code> . If the <code>stop_event</code> is set, it immediately returns an empty list of detections. Otherwise, it copies the input tensor to shared memory, signals the detection task, and waits for the result. If the detection process completes within the timeout, it processes the results from the output shared memory, applying the threshold to filter detections, and returns a list of detections.
03	<code>cleanup()</code>	Cleans up the shared memory segments by unlinking them. This is necessary to free the shared memory resources when they are no longer needed.

4.55 NorfairTracker

4.55.1 Properties of class

Table 153. Properties of class - NorfairTracker

No	Attribute	Description
01	<code>tracked_objects</code>	A dictionary to keep track of objects currently being tracked.
02	<code>untracked_object_boxes</code>	A list of lists, where each inner list contains integers representing the bounding box coordinates of objects that have been detected but are not currently being tracked.
03	<code>disappeared</code>	A dictionary to manage objects that have been tracked but have now disappeared from view.
04	<code>positions</code>	A dictionary to store the current positions of tracked objects.
05	<code>stationary_box_history</code>	A dictionary mapping object identifiers to their historical bounding box positions, specifically for objects that have remained stationary.
06	<code>camera_config</code>	An instance of <code>CameraConfig</code> that holds the configuration settings for the camera.

07	detect_config	Extracted from camera_config, this holds detection-specific configuration settings.
08	camera_name	The name of the camera, derived from camera_config.
09	track_id_map	A dictionary to map tracking IDs, presumably to associate detected objects with unique identifiers.
10	tracker	An instance of Tracker configured with a custom distance function, thresholds, and a Kalman filter factory for object tracking. It is tailored to the specific needs of the tracking system, such as adjusting the trust level on bounding box positions and the prediction reliance.

4.55.2 Class methods

Table 154. Class methods - NorfairTracker

No	Method	Description
01	register()	This method is responsible for registering a new object to be tracked. It generates a unique identifier for the object, updates the tracking maps with the new object, and initializes various attributes for the object, such as start time, motionless count, position changes, and score history based on past detections. The method also initializes the object's position and stationary box history.
02	deregister()	This method removes an object from tracking. It deletes the object from the tracked objects list, removes it from the disappeared list, and updates the internal tracker's list of tracked objects to exclude the object.
03	update_position()	This method updates the current position of a tracked object based on the latest bounding box and determines if the object has moved outside its previously known position. It uses Intersection Over Union (IOU) calculations to assess movement and updates the object's position accordingly.
04	is_expired()	Determines if a tracked object should be deregistered based on its motionlessness. It checks if the object has exceeded a predefined number of frames without significant movement, using a label-specific maximum frame count or a default value.
05	update()	Updates tracking information for a specific object. It resets the disappearance count, checks if the object is stationary, and updates its motionless count accordingly. If the object has moved, it resets the motionless count and updates its position. It also

		handles deregistration of expired objects.
06	update_frame_times()	Prepares a list of current detections to be processed, assuming objects detected in the last frame are still present. It's a preparatory step for matching current detections with tracked objects.
07	match_and_update()	Matches current detections with tracked objects and updates their positions. It involves converting detections into a format suitable for the tracking algorithm, potentially applying coordinate transformations if PTZ (Pan-Tilt-Zoom) camera tracking is enabled, and updating or creating tracks based on the detections.
08	debug_draw()	Visualizes tracking information on a given frame. It draws bounding boxes around active and missing detections, and estimated positions of tracked objects. It also displays the distance calculation for the last detection, aiding in debugging and visual analysis of the tracking performance.

4.56 SPPEFastPose

4.56.1 Properties of class

Table 155. Properties of class - SPPEFastPose

No	Attribute	Description
01	inp_h	Defines the height of the input image to the model.
02	inp_w	Defines the width of the input image to the model.
03	device	Specifies the device on which the model will run. Default is 'cuda', indicating that the model will use an NVIDIA GPU if available.
04	model	This attribute holds the neural network model instance used for inference.

4.56.2 Class methods

Table 156. Class methods - SPPEFastPose

No	Method	Description
01	__init__()	The constructor method initializes the class with the specified backbone, input dimensions, and device.
02	predict()	This method performs the pose estimation on a given image and bounding boxes.

4.57 TSSTG

4.57.1 Properties of class

Table 157. Properties of class - TSSTG

No	Attribute	Description
01	graph_args	A dictionary with graph arguments. In this case, it has a single key 'strategy' with the value 'spatial', indicating that the spatial strategy is used for the graph construction.
02	class_names	A list of class names for the actions the model can predict. It includes 'Non-fall' and 'Fall', indicating the model's purpose is to distinguish between fall and non-fall actions.
03	num_class	The number of action classes, derived from the length of class_names.
04	device	The device on which the model is loaded and run, either 'cpu' or 'cuda'.
05	model	An instance of the DenseSTGCN model, initialized with the number of classes and graph arguments. The model is moved to the specified device.

4.57.2 Class methods

Table 158. Class methods - TSSTG

No	Method	Description
01	<code>__init__()</code>	The constructor method initializes the class with the specified weight file and device.
02	<code>predict()</code>	This method predicts actions based on skeletal points and scores over time.

4.58 Graph

4.58.1 Properties of class

Table 159. Properties of class - Graph

No	Attribute	Description
01	max_hop	The maximum distance between two connected nodes. It controls the range of node connections considered in the graph.
02	dilation	Controls the spacing between kernel points in the graph. This affects the adjacency matrix by determining the distance between connected nodes.
03	num_node	The number of nodes in the graph, which

		corresponds to the number of joints in the skeleton.
04	edge	A list of edges in the graph. Each edge is a tuple representing a connection between two nodes.
05	hop_dis	A matrix representing the distance between nodes in terms of hops (i.e., steps along edges).
06	A	The adjacency matrix of the graph, which is used in the graph convolution operations. Its structure depends on the partition strategy.
07	center	The center node of the graph, used in spatial partitioning strategies. It typically represents the root of the skeleton (e.g., the spine or hip).

4.58.2 Class methods

Table 160. Class methods - Graph

No	Method	Description
01	__init__()	The constructor method initializes the class with the specified parameters.
02	get_edge()	Defines the edges of the graph based on the specified layout.
03	get_adjacency()	Creates the adjacency matrix based on the specified strategy.

4.59 DenseSTGCN

4.59.1 Properties of class

Table 161. Properties of class - DenseSTGCN

No	Attribute	Description
01	st_gcn	This attribute is an instance of the StreamSpatialTemporalGraph class. It is responsible for handling the spatial-temporal graph convolution operations.
02	fcn	This attribute is a fully connected (linear) layer that maps the output from the st_gcn module to the final class predictions.

4.59.2 Class methods

Table 162. Class methods - DenseSTGCN

No	Method	Description
01	__init__()	This is the constructor method for initializing an

		instance of the DenseSTGCN class.
02	forward()	The forward pass method defines how the input tensor inputs is processed through the network.

4.60 StreamSpatialTemporalGraph

4.60.1 Properties of class

Table 163. Properties of class - StreamSpatialTemporalGraph

No	Attribute	Description
01	data_bn	A nn.BatchNorm1d layer that normalizes the input data over the batch dimension. This is important for stabilizing and speeding up the training process.
02	residual_0	An empty nn.Sequential layer, which is defined but not used in this implementation. It can be used to add residual connections if needed.
03	gcn_0	An instance of STGCNLayer, which performs both spatial and temporal convolutions on the input data. It processes the input features with the specified number of input channels and outputs 32 channels.
04	dense_block	An instance of DenseSTGCNBlock with 32 input channels and 2 layers, used to further process the features extracted by gcn_0.
05	cls	A classification layer, which is a nn.Conv2d layer with a kernel size of 1. It maps the 256 output channels from the previous layers to the number of classes if num_class is provided.
06	A	The adjacency matrix of the graph, which is loaded from the Graph class based on the provided graph_args. It is registered as a buffer in the model, which means it is not considered a model parameter but will be moved to the appropriate device (CPU/GPU) along with the model.
07	edge_importance	A nn.ParameterList containing learnable parameters for edge importance weighting. This allows the model to learn the importance of each edge in the graph dynamically during training.

4.60.2 Class methods

Table 164. Class methods - StreamSpatialTemporalGraph

No	Method	Description
01	__init__()	The constructor method initializes the class with the specified parameters.

02	forward()	The forward pass method defines how the input tensor x is processed through the network.
----	-----------	--

4.61 DenseSTGCNBlock

4.61.1 Properties of class

Table 165. Properties of class - DenseSTGCNBlock

No	Attribute	Description
01	st_gcn	This attribute is an instance of the StreamSpatialTemporalGraph class. It is responsible for handling the spatial-temporal graph convolution operations.
02	fcn	This attribute is a fully connected (linear) layer that maps the output from the st_gcn module to the final class predictions.

4.61.2 Class methods

Table 166. Class methods - DenseSTGCNBlock

No	Method	Description
01	<code>__init__()</code>	This is the constructor method for initializing an instance of the DenseSTGCN class.
02	forward()	The forward pass method defines how the input tensor inputs is processed through the network.

4.62 STGCNLayer

4.62.1 Properties of class

Table 167. Properties of class - STGCNLayer

No	Attribute	Description
01	gcn	An instance of the GraphConvolution class, which performs graph convolution operations. It captures spatial relationships in the data.
02	tcn	A nn.Sequential module that performs temporal convolutions and related operations.
03	relu	A ReLU activation layer applied after the temporal convolution block.

4.62.2 Class methods

Table 168. Class methods - STGCNLayer

No	Method	Description
01	<code>__init__()</code>	The constructor method initializes the class with the specified parameters.
02	<code>forward()</code>	The forward pass method defines how the input tensor x and adjacency matrix A are processed through the layer.

4.63 GraphConvolution

4.63.1 Properties of class

Table 169. Properties of class - GraphConvolution

No	Attribute	Description
01	<code>kernel_size</code>	An integer representing the number of adjacency matrices or the number of "hops" in the graph. It controls the size of the convolutional kernel along the spatial dimension.
02	<code>conv</code>	A <code>nn.Conv2d</code> layer that performs a 2D convolution over the input data. It is initialized with parameters for the number of input channels, number of output channels, kernel size, padding, stride, dilation, and whether to use a bias term.

4.63.2 Class methods

Table 170. Class methods - GraphConvolution

No	Method	Description
01	<code>__init__()</code>	The constructor method initializes the class with the specified parameters.
02	<code>forward()</code>	The forward pass method defines how the input tensor x and adjacency matrix A are processed through the graph convolutional layer.

4.64 SEResnet

4.64.1 Properties of class

Table 171. Properties of class - SEResnet

No	Attribute	Description
01	<code>inplanes</code>	An attribute set to 64, representing the number of input channels for the first layer of the network.
02	<code>layers</code>	A list defining the number of blocks in each layer of

		the ResNet model. For resnet50, it is [3, 4, 6, 3], and for resnet101, it is [3, 4, 23, 3].
03	block	The building block used in the ResNet layers. In this case, it is set to the Bottleneck class.
04	conv1	A nn.Conv2d layer that performs a 7x7 convolution with a stride of 2, padding of 3, and no bias term. This is the first convolutional layer of the network.
05	bn1	A nn.BatchNorm2d layer applied after conv1 to normalize the activations.
06	relu	A nn.ReLU layer that applies the ReLU activation function in place.
07	maxpool	A nn.MaxPool2d layer that performs max pooling with a kernel size of 3, stride of 2, and padding of 1.
08	layer1	The first layer of the ResNet model, created using the make_layer method with the specified number of blocks and output channels.
09	layer2	The second layer of the ResNet model, created using the make_layer method with the specified number of blocks, output channels, and a stride of 2.
10	layer3	The third layer of the ResNet model, created using the make_layer method with the specified number of blocks, output channels, and a stride of 2.
11	layer4	The fourth layer of the ResNet model, created using the make_layer method with the specified number of blocks, output channels, and a stride of 2.

4.64.2 Class methods

Table 172. Class methods - SEResnet

No	Method	Description
01	<code>__init__()</code>	The constructor method initializes the class with the specified ResNet architecture
02	<code>forward()</code>	The forward pass method defines how the input tensor <code>x</code> is processed through the layers of the SEResnet model.
03	<code>stages()</code>	This method returns a list of the four ResNet layers.
04	<code>make_layer()</code>	This method creates a ResNet layer consisting of a specified number of blocks with the specified number of output channels (planes).

4.65 Bottleneck

4.65.1 Properties of class

Table 173. Properties of class - Bottleneck

No	Attribute	Description
01	conv1	A nn.Conv2d layer that performs a 1x1 convolution to reduce the number of input channels to the number of planes.
02	bn1	A nn.BatchNorm2d layer applied after conv1 to normalize the activations.
03	conv2	A nn.Conv2d layer that performs a 3x3 convolution with the specified stride, keeping the number of channels the same as planes.
04	bn2	A nn.BatchNorm2d layer applied after conv2 to normalize the activations.
05	conv3	A nn.Conv2d layer that performs a 1x1 convolution to increase the number of channels to planes * 4.
06	bn3	A nn.BatchNorm2d layer applied after conv3 to normalize the activations.
07	se	An instance of the SELayer class, which is used for Squeeze-and-Excitation (SE) if reduction is set to True. It applies channel-wise attention to the output.
08	reduc	A boolean attribute that indicates whether to apply the SELayer.
09	downsample	An optional downsampling layer (typically a convolutional layer with a stride) applied to the input (residual) to match the dimensions of the output if the input and output dimensions differ.
10	stride	The stride used in conv2, affecting the spatial resolution of the feature map.

4.65.2 Class methods

Table 174. Class methods - Bottleneck

No	Method	Description
01	<code>__init__()</code>	The constructor method initializes the class with the specified parameters.
02	<code>forward()</code>	The forward pass method defines how the input tensor <code>x</code> is processed through the layers of the bottleneck block.

4.66 FastPose

4.66.1 Properties of class

Table 175. Properties of class - FastPose

No	Attribute	Description
01	DIM	This attribute is used as the number of output channels for some layers in the model.
02	preact	This attribute holds an instance of the SEResnet model, which serves as the backbone for feature extraction.
03	shuffle1	This layer rearranges elements in a tensor to increase spatial resolution by a factor.
04	duc1	An instance of the DUC (Dense Upsampling Convolution) layer. It takes 512 input channels and produces 1024 output channels, with an upscale factor of 2.
05	duc2	Another instance of the DUC layer. It takes 256 input channels and produces 512 output channels, with an upscale factor of 2.
06	conv_out	A nn.Conv2d layer that performs a 2D convolution. It takes self.DIM (128) input channels and produces num_join output channels (the number of keypoints to be predicted), with a kernel size of 3, stride of 1, and padding of 1.

4.66.2 Class methods

Table 176. Class methods - FastPose

No	Method	Description
01	<code>__init__()</code>	The constructor method initializes the class with the specified backbone and number of keypoints.
02	<code>forward()</code>	This method defines the forward pass of the model. It takes an input tensor <code>x</code> and processes it through the layers defined in the class.

4.67 InferenceNetFastRes50

4.67.1 Properties of class

Table 177. Properties of class - InferenceNetFastRes50

No	Attribute	Description
01	pyranet	This attribute holds an instance of the FastPose model initialized with a 'resnet50' backbone and 17 keypoints (common for human pose estimation)

		tasks). The model is moved to the GPU using .cuda().
--	--	--

4.67.2 Class methods

Table 178. Class methods - InferenceNetFastRes50

No	Method	Description
01	<code>__init__()</code>	The constructor method initializes the class with the provided weights file.
02	<code>forward()</code>	This method defines the forward pass of the model. It takes an input tensor <code>x</code> and passes it through the FastPose model.

4.68 EventsPerSecond

4.68.1 Properties of class

Table 179. Properties of class - EventsPerSecond

No	Attribute	Description
01	<code>_start</code>	Stores the timestamp when the event tracking started. It's initially None and set to the current time when <code>start()</code> or <code>update()</code> is called for the first time.
02	<code>_max_events</code>	The maximum number of events to track. This helps in limiting the size of <code>self._timestamps</code> to avoid excessive memory usage.
03	<code>_last_n_seconds</code>	The time window (in seconds) for which the events per second (EPS) rate is calculated. Only events within this window are considered in the EPS calculation.
04	<code>_timestamps</code>	A list that records the timestamps of each event. It's used to calculate the EPS by counting how many events occurred within the last n seconds.

4.68.2 Class methods

Table 180. Class methods - EventsPerSecond

No	Method	Description
01	<code>__init__()</code>	The constructor initializes the class with the maximum number of events to track and the time window for calculating the EPS.
02	<code>start()</code>	Initializes the start time for event tracking. This method sets <code>self._start</code> to the current timestamp.
03	<code>update()</code>	Records an event by appending the current

		timestamp to self._timestamps.
04	eps()	Calculates the EPS by first ensuring that old timestamps are removed (via expire_timestamps(now)) and then dividing the number of remaining timestamps by the number of seconds since tracking started (capped at last_n_seconds).
05	expire_timestamps()	Removes timestamps that are older than last_n_seconds from the start of the list. This method ensures that only events within the specified time window are considered for the EPS calculation.

4.69 SELayer

4.69.1 Properties of class

Table 181. Properties of class - SELayer

No	Attribute	Description
01	avg_pool	An instance of nn.AdaptiveAvgPool2d that performs adaptive average pooling. This layer reduces the spatial dimensions of the input tensor to 1x1, effectively computing the global average of each channel.
02	fc	A sequential container that defines the fully connected (FC) layers for the squeeze and excitation operations.

4.69.2 Class methods

Table 182. Class methods - SELayer

No	Method	Description
01	__init__()	The constructor method initializes the class with the specified number of channels and reduction ratio.
02	forward()	The forward pass method defines how the input tensor x is processed through the SE block.

4.70 CameraWatchdog

4.70.1 Properties of class

Table 183. Properties of class - CameraWatchdog

No	Attribute	Description
01	camera_name	Identifier for the camera being monitored.

02	config	Configuration settings for the camera, likely including frame shapes, ffmpeg command templates, and retry intervals.
03	capture_thread	A thread for capturing video frames from the camera.
04	ffmpeg_detect_process	The process running FFmpeg for detecting objects or events in the video stream.
05	ffmpeg_other_processes	A list of dictionaries, each containing information about other FFmpeg processes related to the camera (e.g., for recording).
06	camera_fps	A shared, synchronized object tracking the camera's frames per second.
07	skipped_fps	A shared, synchronized object tracking the number of frames per second that are skipped.
08	ffmpeg_pid	A shared, synchronized object storing the process ID of the FFmpeg detection process.
09	frame_queue	A queue for frames that are captured and awaiting processing.
10	frame_shape	The shape of the frames being captured, derived from the camera configuration.
11	frame_size	The size of the frames, calculated from frame_shape.
12	stop_event	An event used to signal when the watchdog should stop running.
13	sleeptime	The interval between retries or checks, based on the camera configuration.

4.70.2 Class methods

Table 184. Class methods - CameraWatchdog

No	Method	Description
01	run()	The main loop of the watchdog thread. It starts the FFmpeg detection process, monitors the health of the capture and FFmpeg processes, and restarts them if necessary. It also handles logging and process management for other FFmpeg processes related to the camera.
02	start_ffmpeg_detect()	Starts the FFmpeg process for detecting objects or events in the video stream, initializes the capture thread, and sets the FFmpeg process ID.

03	get_latest_segment_datetime()	Checks the latest segment datetime for recorded video files, ensuring that recording processes are functioning correctly.
----	-------------------------------	---

4.71 CameraCapture

4.71.1 Properties of class

Table 185. Properties of class - CameraCapture

No	Attribute	Description
01	name	A unique name for the thread, incorporating the camera's name for easier identification.
02	camera_name	The identifier for the camera from which frames are being captured.
03	frame_shape	The dimensions of the video frames being captured.
04	frame_queue	A queue used to hold the frames that have been captured and are ready for further processing.
05	fps	Frames per second, indicating how many frames are intended to be captured per second.
06	skipped_fps	A counter or tracker for the number of frames per second that are being skipped, which can be used for performance adjustments or quality control.
07	stop_event	An event that, when set, signals the thread to stop running. This allows for a controlled shutdown of the frame capture process.
08	frame_manager	An instance of SharedMemoryFrameManager, which is likely responsible for managing the shared memory where frames are temporarily stored for processing.
09	ffmpeg_process	The FFmpeg process used for capturing or processing the video stream from the camera.
10	current_frame	A multiprocessing value that tracks the timestamp or identifier of the current frame being processed. This allows for synchronization or monitoring of the capture process.
11	last_frame	A variable to keep track of the last frame that was processed, which can be used for calculating the difference or delay between frames.

4.71.2 Class methods

Table 186. Class methods - CameraCapture

No	Method	Description
01	run()	This method is called when the thread starts. It encapsulates the main functionality of the CameraCapture thread, which involves calling a function capture_frames with all necessary parameters to capture video frames from the camera.

4.72 CameraFfmpegConfig

4.72.1 Properties of class

Table 187. Properties of class - CameraFfmpegConfig

No	Attribute	Description
01	inputs	This attribute allows for specifying multiple camera sources, each with its own set of FFmpeg arguments and roles.

4.72.2 Class methods

Table 188. Class methods - CameraFfmpegConfig

No	Method	Description
01	validate_roles()	Prevent configuration errors that could lead to functional issues or inefficiencies in the application's video processing capabilities.

4.73 TimelineProcessor

4.73.1 Properties of class

Table 189. Properties of class - TimelineProcessor

No	Attribute	Description
01	config	An instance of VigisionConfig, containing system configuration details, including camera configurations.
02	queue	A Queue instance where events are pushed for processing.
03	stop_event	A MpEvent (multiprocessing event) used to signal when the thread should stop running.
04	pre_event_cache	A dictionary caching events that cannot be immediately saved to the database because they lack associated clips or snapshots.

4.73.2 Class methods

Table 190. Class methods - TimelineProcessor

No	Method	Description
01	run()	The main loop of the thread, which continuously checks the queue for new events and processes them unless a stop signal is received. It handles object detection events and API entries differently based on the event type.
02	insert_or_save()	Determines whether to insert an event into the database immediately or cache it. Events without clips or snapshots are cached until they can be associated with such media.
03	handle_object_detection()	Processes object detection events by creating timeline entries with relevant data (e.g., bounding boxes, labels) and deciding based on the event type (start, update, end) and other conditions whether to save the event immediately or update cached events.

4.74 Timeline

4.74.1 Properties of class

Table 191. Properties of class - Timeline

No	Attribute	Description
01	timestamp	Records the exact date and time when the event occurred. This is crucial for placing events in chronological order on a timeline.
02	camera	Identifies the camera that captured the event, allowing events to be filtered or searched by camera. This field is indexed to optimize queries by camera ID.
03	source	Specifies the origin of the event, such as a tracked object, an audio signal, or an external input. This field helps categorize events by their source, facilitating easier filtering and analysis.
04	source_id	Provides a unique identifier for the source of the event, enabling precise tracking and correlation of events to their origins. This field is indexed to improve search performance.
05	class_type	Describes the type of event, such as an object entering a zone or an audio event being detected. This classification allows for the categorization and filtering of events based on their nature.

06	data	Stores additional information about the event in a flexible JSON format. This could include details like the ID of a tracked object, the region of interest, bounding box coordinates, and more. The use of a JSON field allows for the storage of structured data that can vary between different types of events.
----	------	---

4.74.2 Class methods

This class doesn't have class methods.

4.75 EventProcessor

4.75.1 Properties of class

Table 192. Properties of class - EventProcessor

No	Attribute	Description
01	config	Stores the configuration (VigisionConfig) for the event processor, likely containing settings and parameters for event handling.
02	timeline_queue	A Queue object used to hold events that are being processed or need to be processed.
03	events_in_process	A dictionary (Dict[str, Event]) that tracks events currently being processed, keyed by event ID.
04	stop_event	An MpEvent (multiprocessing event) used to signal the thread to stop running.
05	event_receiver	An instance of EventUpdateSubscriber, likely used to receive updates about events.
06	event_end_publisher	An instance of EventEndPublisher, likely used to publish notifications when events end.

4.75.2 Class methods

Table 193. Class methods - EventProcessor

No	Method	Description
01	__init__()	The constructor initializes the thread and its attributes with the provided configuration, queue, and stop event.
02	run()	The main method of the thread, which continuously checks for updates from event_receiver, processes received events based on their type (tracked_object or api), and handles them accordingly. It also ensures that events without an end time are updated at startup and stops the event_receiver and event_end_publisher upon receiving a stop signal.

03	handle_object_detection()	Handles updates for events related to object detection. It decides whether to update the database based on the event data, processes the event (including calculating start and end times, scores, regions, and bounding boxes), and updates or inserts the event into the database. It also manages the lifecycle of events in the events_in_process dictionary.
----	---------------------------	---

4.76 EventEndPublisher

4.76.1 Properties of class

Table 194. Properties of class - EventEndPublisher

No	Attribute	Description
01	context	An instance of zmq.Context, which is essential for managing ZeroMQ's underlying resources. It's used here to create and manage the lifecycle of the ZeroMQ socket.
02	socket	A ZeroMQ socket of type PULL. This socket is used to receive messages sent by a PUSH socket. It's set up to bind to the address specified by SOCKET_PUSH_PULL, which is a constant defining the endpoint where messages are received.

4.76.2 Class methods

Table 195. Class methods - EventEndPublisher

No	Method	Description
01	__init__()	Initializes the EventEndPublisher instance.
02	publish()	Publishes information about an event that has ended.
03	stop()	Cleans up resources before shutting down.

4.77 EventUpdateSubscriber

4.77.1 Properties of class

Table 196. Properties of class - EventUpdateSubscriber

No	Attribute	Description
01	context	An instance of zmq.Context, which is essential for managing ZeroMQ's underlying resources. It's used here to create and manage the lifecycle of the ZeroMQ socket.
02	socket	A ZeroMQ socket of type PULL. This socket is used to receive messages sent by a PUSH socket. It's set up to bind to the address specified by SOCKET_PUSH_PULL, which is a constant defining the endpoint where messages are received.

4.77.2 Class methods

Table 197. Class methods - EventUpdateSubscriber

No	Method	Description
01	__init__(self)	Initializes the EventUpdateSubscriber instance.
02	check_for_update(self)	Checks for new event updates within a specified timeout period.
03	stop(self)	Cleans up resources before shutting down.

4.78 EventCleanupType

4.78.1 Properties of class

Table 198. Properties of class - EventCleanupType

No	Attribute	Description
01	clips	Represents the type of media related to video clips. Its value is the string "clips". This could be used to specify operations or configurations that apply specifically to video clip data within the system.
02	snapshots	Represents the type of media related to snapshots or still images. Its value is the string "snapshots". This is used similarly to clips, but for operations or configurations that apply to snapshot data.

4.78.2 Class methods

This class doesn't have class methods.

4.79 EventCleanup

4.79.1 Properties of class

Table 199. Properties of class - EventCleanup

No	Attribute	Description
01	name	A string identifier for the thread, set to "event_cleanup".
02	config	An instance of VigisionConfig, storing configuration details for the system, including camera settings and data retention policies.
03	stop_event	An instance of MpEvent (multiprocessing event), used to signal the thread to stop running.

04	camera_keys	A list of camera identifiers extracted from the system's configuration, used to determine which cameras are currently active.
05	removed_camera_labels	Initially None, this list is populated with labels of cameras that have been removed from the system's configuration.
06	camera_labels	A dictionary mapping camera identifiers to their associated labels and the last update timestamp. It's used to cache labels for each camera and reduce database queries.

4.79.2 Class methods

Table 200. Class methods - EventCleanup

No	Method	Description
01	get_removed_camera_labels()	Returns a list of distinct labels for cameras that have been removed from the system's configuration. It ensures that data related to these cameras can be identified for cleanup.
02	get_camera_labels()	Retrieves a list of distinct labels for a specific camera, updating the cache once a day to reflect any new labels. This method helps in identifying data for cleanup based on camera-specific labels.
03	expire()	The core method that handles the expiration of events. It operates based on the type of media (clips or snapshots) and uses the system's configuration to determine how long events should be retained.

04	run()	Handles the deletion of timeline entries for events with expired recordings and removes events from the database if they no longer have associated media.
----	-------	---

4.80 RecordingCleanup

4.80.1 Properties of class

Table 201. Properties of class - RecordingCleanup

No	Attribute	Description
01	config	Stores the configuration (VigisionConfig) for the cleanup process, including database paths, retention policies, etc.
02	stop_event	An event (MpEvent) used to signal the thread to stop running.

4.80.2 Class methods

This class doesn't have class methods

4.81 Previews

4.81.1 Properties of class

Table 202. Properties of class - Previews

No	Attribute	Description
01	id	Acts as the unique identifier for each preview entry. It is the primary key in the database, ensuring that each preview can be uniquely identified.
02	camera	Identifies the camera that generated the preview. This field is indexed to optimize queries by camera ID, making it easier to retrieve previews based on the camera source.

03	path	Stores the file path or location where the preview is saved. This field is marked as unique, ensuring that no two previews share the same file path, which helps in maintaining the integrity of the file storage system.
04	start_time	These fields record the timestamps for when the preview starts, respectively. They are crucial for defining the exact duration of the footage from which the preview was generated.
05	end_time	These fields record the timestamps for when the preview ends, respectively. They are crucial for defining the exact duration of the footage from which the preview was generated.
06	duration	Represents the total length of the preview in seconds (or another unit of time), calculated as the difference between end_time and start_time. This field provides a quick reference to the length of the preview without needing to calculate it from the start and end times.

4.81.2 Class methods

This class doesn't have class methods.

4.82 StorageMaintainer

4.82.1 Properties of class

Table 203. Properties of class - StorageMaintainer

No	Attribute	Description
01	config	An instance of VigisionConfig, which contains configuration details for the system, including camera configurations.
02	stop_event	A threading event used to signal when the thread should stop running.

03	camera_storage_stats	A dictionary that holds storage statistics for each camera, including whether the camera's storage stats need refreshing and its bandwidth usage.
----	----------------------	---

4.82.2 Class methods

Table 204. Class methods - StorageMaintainer

No	Method	Description
01	calculate_camera_bandwidth()	Calculates the average bandwidth usage (in MB/hr) for each camera based on the size of recorded segments. It updates camera_storage_stats with this information, flagging cameras with fewer than 50 segments for a refresh to keep size calculations accurate.
02	check_storage_needs_cleanup()	Determines if storage cleanup is necessary by comparing the sum of hourly bandwidth usage across all cameras against the remaining storage space.
03	reduce_storage_consumption()	Deletes the oldest hour of recordings to free up space. It prioritizes deleting non-retained recordings first and, if necessary, moves on to delete retained recordings to meet the required space clearance. It also handles exceptions for missing files and logs the outcome of the cleanup process.

04	run()	The main loop of the thread, which periodically checks if storage maintenance is needed every 5 minutes. It recalculates camera bandwidths if necessary and initiates storage cleanup when the available space falls below the threshold. The loop continues until the stop_event is signaled.
----	-------	--

4.83 StatsEmitter

4.83.1 Properties of class

Table 205. Properties of class - StatsEmitter

No	Attribute	Description
01	config	An instance of VigisionConfig that holds the configuration settings for the system.
02	stats_tracking	An instance of StatsTrackingTypes that specifies the types of statistics to track.
03	stop_event	An instance of MpEvent (multiprocessing.Event) used to signal the thread to stop running.
04	hwaccel_errors	A list that stores hardware acceleration errors encountered during the operation.
05	stats_history	A list that maintains a history of collected statistics snapshots.
06	requestor	An instance of InterProcessRequestor used for inter-process communication, specifically for sending statistics data.

4.83.2 Class methods

Table 206. Class methods - StatsEmitter

No	Method	Description
01	<code>__init__()</code>	The constructor initializes the thread with the given configuration, stats tracking types, and stop event. It also initializes the hwaccel_errors and stats_history lists and creates a requestor for inter-process communication.
02	<code>get_latest_stats()</code>	Returns the most recent statistics snapshot from stats_history. If stats_history is empty, it generates a new snapshot using the stats_snapshot function, appends it to stats_history, and returns it.
03	<code>get_stats_history()</code>	Returns the history of statistics snapshots. If a list of keys is provided, it filters the snapshots to include only the specified keys. Otherwise, it returns the full history.
04	<code>run()</code>	The main method executed by the thread. It waits for a specified initial delay, then enters a loop where it periodically collects statistics, updates stats_history, and sends the latest statistics to another process via requestor if it's time to emit stats. The loop continues until the stop_event is set, signaling the thread to exit.

4.84 StatsTrackingTypes

4.84.1 Properties of class

Table 207. Properties of class - StatsTrackingTypes

No	Attribute	Description
01	camera_metrics	This attribute tracks metrics specific to each camera in the system, such as frame rates, process IDs, and queue sizes, providing a detailed view of each camera's operational performance.
02	detectors	This attribute is used to track the status and performance of object detection processes running within the system, allowing for monitoring and management of these critical components.
03	started	An integer representing a timestamp (likely in Unix epoch format) indicating when the statistics tracking or the system itself was started. This provides a reference point for calculating uptime and analyzing performance over time.
04	latest_vigision_version	A string indicating the latest version of the Vigision software (or a similar surveillance/detection system) that is being used. This helps in ensuring that the system is up-to-date and can be useful for troubleshooting or compatibility checks.
05	last_updated	An integer representing a timestamp (likely in Unix epoch format) of the last time these statistics were updated. This is crucial for understanding the freshness of the data and ensuring that monitoring tools are displaying current information.

06	processes	This attribute tracks the various subprocesses or threads that are part of the system's operation, allowing for management and monitoring of these processes to ensure they are running as expected.
----	-----------	--

4.84.2 Class methods

This class doesn't have class methods.

4.85 User

4.85.1 Properties of class

Table 208. Properties of class - User

No	Attribute	Description
01	username	This attribute represents the username of the user.
02	password_hash	This attribute stores a hash of the user's password. Storing a hash (a fixed-size string or number generated from input data of arbitrary size, in this case, the password) instead of the actual password is a security best practice.

4.85.2 Class methods

This class doesn't have class methods.

4.86 CameraGroupConfig

4.86.1 Properties of class

Table 209. Properties of class - CameraGroupConfig

No	Attribute	Description
01	cameras	This attribute can hold either a single string or a list of strings. Each string represents a camera identifier.
02	icon	A string attribute that holds the identifier for an icon representing the camera group.
03	order	An integer attribute that specifies the sort order for the camera group.

4.86.2 Class methods

Table 210. Class methods - CameraGroupConfig

No	Method	Description
01	validate_cameras()	This method is designed to streamline data handling and ensure consistency and validity of the cameras attribute within the CameraGroupConfig class, making the system more robust and easier to maintain.

4.87 Communicator

4.87.1 Properties of class

There is no property of class.

4.87.2 Class methods

Table 211. Class methods - Communicator

No	Method	Description
01	publish()	This method is intended to send data to all subscribers interested in a specific topic.
02	subscribe()	This method is used to register a receiver (a callable function or method) that will be invoked with messages from the subscribed topics.
03	stop()	This method is responsible for gracefully stopping the communicator, ensuring that all resources (e.g., network connections, threads) are properly released and that the communicator is cleanly shut down.

4.88 StationaryMaxFramesConfig

4.88.1 Properties of class

Table 212. Properties of class - StationaryMaxFramesConfig

No	Attribute	Description
01	default	Specify the default maximum number of frames for which an object can remain stationary in the video or image feed before some action is taken or a condition is triggered.
02	objects	Specific object types or identifiers, and the values represent the maximum number of frames those specific objects can remain stationary.

4.88.2 Class methods

This class doesn't have class methods.

4.89 StationaryConfig

4.89.1 Properties of class

Table 213. Properties of class - StationaryConfig

No	Attribute	Description
01	interval	Specifies the frame interval at which the application checks for stationary objects. This means that the application will assess whether an object has remained stationary every interval number of frames.
02	threshold	Defines the number of consecutive frames an object must remain without a significant position change to be considered stationary. This threshold helps differentiate between truly stationary objects and those that are moving very slowly or intermittently.

03	max_frames	Holds configuration settings related to the maximum number of frames a stationary object is allowed to remain in the scene before triggering a specific action or being flagged. This attribute uses an instance of StationaryMaxFramesConfig, which allows for both a default maximum frame setting and object-specific maximum frame settings.
----	------------	--

4.89.2 Class methods

This class doesn't have class methods.

4.90 RuntimeMotionConfig

4.90.1 Properties of class

Table 214. Properties of class - RuntimeMotionConfig

No	Attribute	Description
01	mask	This attribute is used to define areas within the video frame where motion detection should be applied or ignored.
02	raw_mask	This attribute is intended to store the raw mask data in a more human-readable or serialized format.
03	model_config	Hold additional configuration options for the model.

4.90.2 Class methods

Table 215. Class methods - RuntimeMotionConfig

No	Method	Description
01	<code>__init__()</code>	Initializes the RuntimeMotionConfig instance.

02	dict()	Overrides or extends a method. If the mask key exists in the returned dictionary, it replaces its value with the value of raw_mask and removes the raw_mask key. This adjustment ensures that the serialized representation uses the raw mask data.
03	serialize_mask()	Returns the value of self.raw_mask, allowing the raw mask data to be used in place of the actual mask data during serialization.
04	serialize_raw_mask()	Intend to prevent redundancy or to keep internal data from being exposed in the serialization process.

4.91 WebSocketClient

4.91.1 Properties of class

Table 216. Properties of class - WebSocketClient

No	Attribute	Description
01	config	Stores the configuration passed during initialization, presumably containing settings relevant to the WebSocket client's operation.
02	websocket_server	Initially set to None, this attribute is later assigned a WebSocket server instance that is created when the start method is called.
03	_dispatcher	A callback function set by the subscribe method, intended to handle incoming messages.

4.91.2 Class methods

Table 217. Class methods - *WebSocketClient*

No	Method	Description
01	<code>__init__()</code>	Initializes the instance with a given configuration and sets the <code>websocket_server</code> attribute to <code>None</code> .
02	<code>subscribe()</code>	Sets the <code>_dispatcher</code> attribute to the provided receiver function, which is intended to handle incoming messages. It then starts the WebSocket server by calling the <code>start</code> method.
03	<code>start()</code>	Initializes and starts the WebSocket server. It defines an inner class <code>_WebSocketHandler</code> that inherits from <code>WebSocket</code> and overrides the <code>received_message</code> method to handle incoming WebSocket messages.
04	<code>publish()</code>	Attempts to serialize a message (consisting of a topic and payload) into JSON and broadcast it to all connected WebSocket clients. If serialization fails or the server is not yet connected, the message is not sent. It silently handles <code>ConnectionResetError</code> to avoid crashing when a connection is reset.

05	stop()	Gracefully shuts down the WebSocket server and its associated thread. It closes all WebSocket connections, stops the server's manager, and joins the server thread to ensure a clean exit. Logs a message indicating the client is exiting.
----	--------	---

4.92 WebSocket

4.92.1 Properties of class

There is no property of class.

4.92.2 Class methods

Table 218. Class methods - WebSocket

No	Method	Description
01	unhandled_error()	To handle errors that occur during WebSocket operations, specifically to manage how socket closure errors are reported.

4.93 ConfigPublisher

4.93.1 Properties of class

Table 219. Properties of class - ConfigPublisher

No	Attribute	Description
01	context	An instance of zmq.Context. This is the ZeroMQ context that manages sockets within the ZeroMQ framework. It's essential for establishing communication channels.

02	socket	A ZeroMQ socket created with self.context.socket(zmq.PUB), configured as a publisher (PUB) socket. This socket is used to publish messages to all subscribers interested in those messages.
03	stop_event	An instance of MpEvent (presumably from the multiprocessing module, aliased as mp). This event is used to signal when the ConfigPublisher should stop its operation, facilitating a graceful shutdown.

4.93.2 Class methods

Table 220. Class methods - ConfigPublisher

No	Method	Description
01	__init__()	Initializes the ConfigPublisher instance by setting up the ZeroMQ context and publisher socket, binding the socket to the address specified in SOCKET_PUB_SUB, and initializing the stop_event.
02	publish()	Publishes a message with a given topic and payload to all subscribing processes. This method allows for the transmission of complex data structures directly, without needing to serialize them manually. There's no communication back from the subscribers to the publisher, making this a one-way communication channel.
03	stop()	Signals the ConfigPublisher to stop by setting the stop_event. It then closes the ZeroMQ socket and destroys the ZeroMQ context to clean up resources. This method ensures a graceful shutdown of the publisher, releasing the resources it has acquired.

4.94 RecordExportConfig

4.94.1 Properties of class

Table 221. Properties of class - RecordExportConfig

No	Attribute	Description
01	timelapse_args	This string attribute specifies the arguments to be used with ffmpeg when creating a timelapse from recorded footage, allowing for customization of the timelapse creation process, enabling adjustments to video quality, frame rate, resolution, and other parameters that ffmpeg supports.

4.94.2 Class methods

This class doesn't have class methods.

4.95 RuntimeFilterConfig

4.95.1 Properties of class

Table 222. Properties of class - RuntimeFilterConfig

No	Attribute	Description
01	mask	Stores the mask as a NumPy array, which is more suitable for runtime operations, such as applying the mask to frames of a video stream.
02	raw_mask	Holds the raw mask data, which can be a string or a list of strings. This attribute is used as an intermediate step in generating the mask attribute.
03	model_config	Provides configuration options for the model

4.95.2 Class methods

Table 223. Class methods - RuntimeFilterConfig

No	Method	Description
01	<code>__init__()</code>	Initializes the RuntimeFilterConfig instance.
02	<code>dict()</code>	Overrides the dict method to customize the serialization of the RuntimeFilterConfig instance. This method is typically used to convert the instance into a dictionary, often for JSON serialization.

4.96 FfmpegConfig

4.96.1 Properties of class

Table 224. Properties of class - FfmpegConfig

No	Attribute	Description
01	<code>inputs</code>	This attribute allows for specifying multiple camera sources, each with its own set of FFmpeg arguments and roles.

4.96.2 Class methods

Table 225. Class methods - FfmpegConfig

No	Method	Description
01	<code>validate_roles()</code>	Prevent configuration errors that could lead to functional issues or inefficiencies in the application's video processing capabilities.

4.97 Export

4.97.1 Properties of class

Table 226. Properties of class - Export

No	Attribute	Description
01	id	Serves as the unique identifier for each export entry. It is the primary key in the database, ensuring that each export can be uniquely identified.
02	camera	Identifies the camera from which the video was exported. This field is indexed to optimize queries filtering by camera ID, making it easier to retrieve exports based on the camera source.
03	name	A human-readable name or title for the export. This field is also indexed, which helps in searching or filtering exports based on their names.
04	date	The date and time when the export was created or initiated. This information is crucial for organizing and retrieving exports based on when they occurred.
05	video_path	Stores the file path or location where the exported video is saved. This field is marked as unique, ensuring that no two exports share the same video file path, which helps in maintaining the integrity of the file storage system.
06	thumb_path	Stores the file path or location for the thumbnail image associated with the export. Like video_path, this field is also unique, ensuring that each export has a distinct thumbnail image.

07	in_progress	A boolean field indicating whether the export process is currently in progress or if it has been completed. This allows the system to track the status of exports, potentially enabling users to query for completed exports or monitor the progress of ongoing ones.
----	-------------	---

4.97.2 Class methods

This class doesn't have class methods.

4.98 RecordingExporter

4.98.1 Properties of class

Table 227. Properties of class - RecordingExporter

No	Attribute	Description
01	config	An instance of VigisionConfig, containing configuration details for the export process.
02	camera	A string identifier for the camera whose recordings are to be exported.
03	user_provided_name	A custom name provided by the user for the export file.
04	start_time	The start timestamp (integer) for the range of recordings to be exported.
05	end_time	The end timestamp (integer) for the range of recordings to be exported.
06	playback_factor	An enum value of PlaybackFactorEnum, indicating the playback speed (e.g., realtime, timelapse) for the exported video.

4.98.2 Class methods

Table 228. Class methods - RecordingExporter

No	Method	Description
01	<code>__init__()</code>	Initializes the thread with the necessary configuration, camera details, and export parameters.
02	<code>get_datetime_from_timestamp()</code>	Converts a timestamp into a human-readable date-time string.
03	<code>save_thumbnail()</code>	Generates a thumbnail for the export, either by extracting a frame from a preview video or copying an existing image, and saves it as a .webp file.
04	<code>run()</code>	The main method executed by the thread. It generates an export ID, constructs the export name, prepares the video path, and saves the thumbnail. Depending on the duration of the export and the playback factor, it constructs an appropriate ffmpeg command to export the recordings as a single video file. It handles errors, cleans up in case of failure, and updates the export status upon completion.

4.99 PlaybackFactorEnum

4.99.1 Properties of class

Table 229. Properties of class - PlaybackFactorEnum

No	Attribute	Description
01	<code>realtime</code>	Represents the playback factor for real-time video playback. This could be used to indicate that the

		video should be played back at the speed it was recorded, without any acceleration or deceleration.
02	timelapse_25x	Represents a timelapse playback factor, specifically 25 times faster than real-time. This could be used to indicate that the video should be played back at a speed that is 25 times faster than the speed at which it was recorded.

4.99.2 Class methods

This class doesn't have class methods.

5. Database Design

5.1 Local database

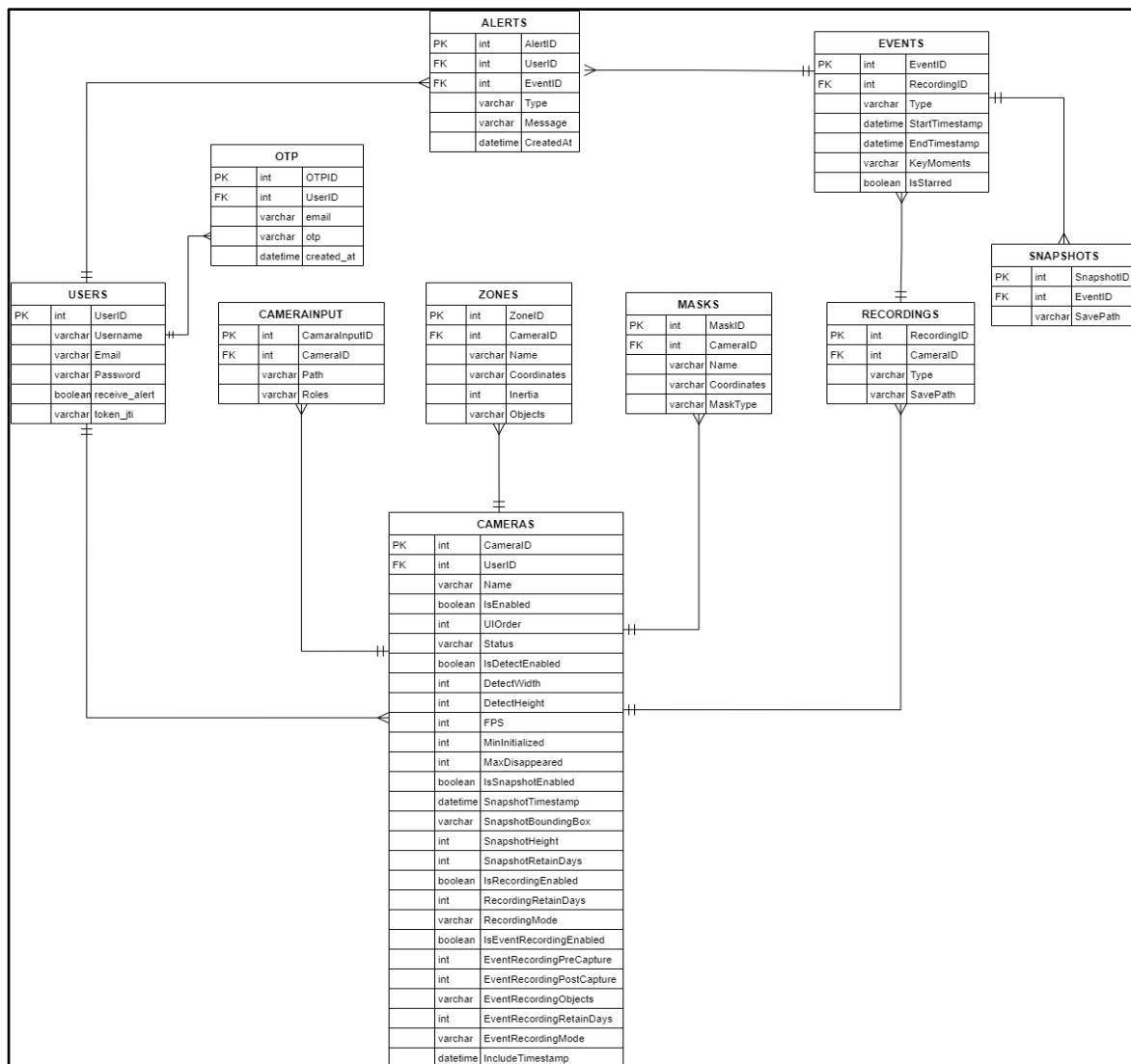


Figure 237. Local database schema diagram

Table 230. Local database description

No	Table	Description
01	Users	Stores information about individuals who interact with the system. - Primary Key: UserID - Foreign Key: None
02	Cameras	Holds details about cameras connected to the system, capturing video footage. - Primary Key: CameraID - Foreign Key: UserID (references User)
03	CameralInput	Contains information about the stream of data received from a connected camera. - Primary Key: CameralInputID - Foreign Key: CameraID (references Camera)
04	Zones	Represents predefined areas within a camera's field of view to determine object presence. - Primary Key: ZoneID - Foreign Key: CameraID (references Camera)
05	Masks	Defines areas within a camera's field of view that are excluded from detection or analysis. - Primary Key: MaskID - Foreign Key: CameraID (references Camera)
06	Recordings	Stores saved video footage captured by a camera. - Primary Key: RecordingID - Foreign Key: CameraID (references Camera)
07	Events	Detected incidents or activities captured by the system. - Primary Key: EventID - Foreign Key: RecordingID (references Recording)
08	Snapshots	Images captured from a camera at specific moments, associated with events. - Primary Key: SnapshotID - Foreign Key: EventID (references Event)
09	Alerts	Message sent to users via Google Mail regarding detected events. - Primary Key: AlertID - Foreign Key: UserID (references User), EventID (references Event)
10	OTP	OTP sent to user email. - Primary Key: OTPID - Foreign Key: UserID

V. Software Testing Documentation

1. Scope of Testing

The scope of the testing for the Vigision Real-time Intelligent Video Surveillance System includes both functional and non-functional requirements. The primary features to be tested include:

1. Login
2. Logout
3. Manage users
4. Get shareable link
5. View live cameras
6. Manage cameras
7. Manage camera groups
8. Edit camera group layout
9. View in full screen
10. View detail of live camera
11. Enable/disable detection
12. Enable/disable recording
13. Enable/disable snapshots
14. View defined masks and zones
15. Manage motion masks
16. Manage zones
17. Manage object masks
18. Fine-tune motion detection
19. Debug
20. View alerts/detections
21. View motion events
22. View event recording
23. Mark events as reviewed
24. Delete events
25. View exports
26. View export detail
27. Export event recording
28. Export camera recording
29. Download export
30. Rename export
31. Delete export
32. Config
33. View system metrics
34. Restart application
35. Change application theme
36. Forgot Password

The features that will not be tested include:

- **Non-critical UI elements:** Aesthetic and stylistic elements of the user interface that do not impact functionality.

- **Third-party Integrations:** Any integrations with external systems or third-party services not essential for core functionalities.
- **Deprecated Features:** Features marked for removal or no longer in active use.
- **Future Enhancements:** Features planned for future releases but not included in the current scope.
- **Performance Under Extreme Conditions:** System performance under conditions beyond typical operational expectations (e.g., extremely high load scenarios).

The testing stages/levels to be applied include:

- **Unit Testing:** To be performed by developers to validate individual components and their functions.
- **Integration Testing:** Conducted by both developers and testers to ensure that different components of the system work together as expected.
- **System Testing:** Performed by the QA team to validate the system as a whole, ensuring it meets all specified requirements.
- **Acceptance Testing:** Conducted with the involvement of end-users to ensure the system meets business needs and requirements.

Constraints or assumptions:

- The system must be operational and connected to the necessary hardware and network infrastructure.
- Test environments must mirror production environments as closely as possible.
- Testing will be based on the assumption that all preliminary configurations and setups are correctly performed.

2. Test Strategy

The test strategy encompasses multiple testing types, each with specific objectives, techniques, and completion criteria. The following sections detail the selected test types, test levels, and supporting tools.

2.1 Testing Levels

Unit Testing

- **Objective:** Validate individual components and their functions.
- **Technique:** White-box testing, automated tests.
- **Completion Criteria:** All unit tests pass with no critical defects.

Integration Testing

- **Objective:** Ensure different components work together as expected.
- **Technique:** Combination of automated and manual testing.
- **Completion Criteria:** All integrations function correctly with no critical defects.

System Testing

- **Objective:** Validate the entire system against the specified requirements.
- **Technique:** Black-box testing, end-to-end testing.
- **Completion Criteria:** System meets all functional and non-functional requirements.

Acceptance Testing

- **Objective:** Ensure the system meets business needs and requirements.
- **Technique:** User acceptance testing (UAT), involving end-users.
- **Completion Criteria:** System is accepted by end-users and stakeholders.

2.2 Test Types

Table 231. Test types

Type of Tests	Test Level			
	Unit	Integration	System	Acceptance
Functional Tests	X	X	X	X
Performance Tests			X	
Security Tests			X	
Usability Tests			X	X

2.3 Supporting Tools

Table 232. Supporting Tools

Purpose	Tool	Vendor/In-house	Version
Unit Testing	Manual	Team	Latest
Integration Testing	Postman	Open Source	Latest
System Testing	Postman	Open Source	Latest
Performance Testing	Manual	Team	Latest
Security Testing	Manual	Team	Latest

3. Test Plan

3.1 Human Resources

Table 233. Human resources

Worker/Doer	Role	Specific Responsibilities/Comments
Phan Hong Phuc	Project Manager	Overseeing testing activities and resources
Dinh The Anh	QA Engineer	System testing and test case documentation
Vo Minh Dat	Test Analyst	Acceptance testing and stakeholder liaison
Nguyen Le Quang Thinh	Developer	Unit testing and integration testing

3.2 Test Environment

Table 234. Test environment

Purpose	Tool	Provider	Version
Unit Testing	Manual	Team	Latest
Integration Testing	Postman	Open Source	Latest
System Testing	Postman	Open Source	Latest
Performance Testing	Manual	Team	Latest
Security Testing	Manual	Team	Latest

3.3 Test Milestones

Table 235. Test milestones

Milestone Task	Start Date	End Date
Test Plan Review	2024-07-27	2024-07-28
Unit Testing Completion	2024-07-29	2024-08-05
Integration Testing Completion	2024-08-06	2024-08-06
System Testing Completion	2024-08-07	2024-08-07
Acceptance Testing Completion	2028-08-08	2024-08-08

4. Test Cases

The test cases are described in the file below:

- *Unit Test Cases: Report5_Unit Test.xls*
- *Other Test Cases (IT, ST, AT): Report5_Test Report.xls*

5. Test Reports



Figure 239. Test Reports

VI. Release Package & User Guides

1. Deliverable Package

1.1 Source codes & documents

Table 236. Source codes & documents

No.	Items	Sub-Items	Type	Version
Code Package				
1	Backend	Vigision_BE	New	7.0.0
2	Frontend	Vigision_FE	New	7.0.0
3	Machine Learning	Vigision_AI	New	7.0.0
Database				
1	Peewee schema migrations for SQLite	create_events_table.py	New	7.0.0
		add_clip_snapshot.py	New	7.0.0
		create_recordings_table.py	New	7.0.0
		add_bbox_region_area.py	New	7.0.0
		add_bbox_region_area.py	New	7.0.0
		make_end_time_nullable.py	New	7.0.0
		add_motion_active_objects.py	New	7.0.0
		add_retain_indefinitely.py	New	7.0.0
		add_sub_label.py	New	7.0.0
		add_object_filter_ratio.py	New	7.0.0
		update_indexes.py	New	7.0.0
		add_segment_size.py	New	7.0.0
		create_timeline_table.py	New	7.0.0
		event_refactor.py	New	7.0.0
		update_indexes.py	New	7.0.0
		create_regions_table.py	New	7.0.0
		update_index_recordings.py	New	7.0.0
		create_previews_table.py	New	7.0.0
		create_review_segment_table.py	New	7.0.0
		add_regions.py	New	7.0.0
		create_export_table.py	New	7.0.0
		create_user_table.py	New	7.0.0
		add_email_user_table.py	New	7.0.0

No.	Items	Sub-Items	Type	Version
		create_otp_table.py	New	7.0.0
		add_receive_alert_user_table.py	New	7.0.0
2	YAML/YML data	config.yml.example	New	7.0.0
		config.yaml	New	7.0.0
Documents				
1	Report 1	Report1_Project Introduction.docx	New	7.0.0
2	Report 2	Report2_Project Management Plan.docx	New	7.0.0
3	Report 3	Report3_Software Requirement Specification.docx	New	7.0.0
4	Report 4	Report4_Software Design Document.docx	New	7.0.0
5	Report 5	Report5_Test Documentation.docx	New	7.0.0
		Report5_Unit Test.xlsx	New	7.0.0
		Report5_Test Report.xlsx	New	7.0.0
6	Report 6	Report6_Software User Guides.docx	New	7.0.0
7	Report 7	Report7_Final Project Report.docx	New	7.0.0

1.2 Known Issues, Limitations & Restrictions

- **Lack of Support for Audio and Pan-Tilt-Zoom (PTZ):** The current system version does not support audio functionalities such as two-way audio or voice commands. Additionally, Pan-Tilt-Zoom (PTZ) control is not yet implemented. These features may be considered for future updates.
- **Limited Debugging Capabilities:** The system lacks a comprehensive console logging mechanism to provide detailed error information. This limitation poses challenges in debugging and troubleshooting issues, as it becomes difficult to pinpoint the exact source or nature of errors.
- **Model and Object Detection Limitations:** At present, the system does not support dynamic model switching or customization of object detection lists. The list of detectable objects is currently limited to predefined categories, restricting its flexibility in diverse environments.

2. Installation Guides

2.1 System Requirements

- Operating system: Windows 10 or higher with WSL2 (Windows Subsystem for Linux 2); Ubuntu 20.4 or higher.
- Storage: about 40GB for installation. Additionally, disk space may be required for data storage, depending on the size and complexity of the system.
- RAM: 8GB or higher.
- CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz (8 CPUs), ~2.4GHz or higher.
- GPU: NVIDIA GeForce GTX 1660 Ti or higher - recommended
- Software:
 - + Ubuntu: Docker v26.1.4, Docker Compose v2.20.3 (recommended).
 - + Windows: Docker Desktop v4.29.0, WSL 2 (recommended).
- Network: required

Note:

- Vigision's performance will depend on number of cameras, power of your CPU and GPU.
- GPU is not required but deep learning features like object detection, pose estimation and fall detection need GPU power to perform well, if your machine has no GPU, motion detection can still work.

2.2 Installation Instruction

2.2.1 Install and setup Docker

1. Download Docker Desktop 4.29.0 at <https://www.docker.com/products/docker-desktop/>

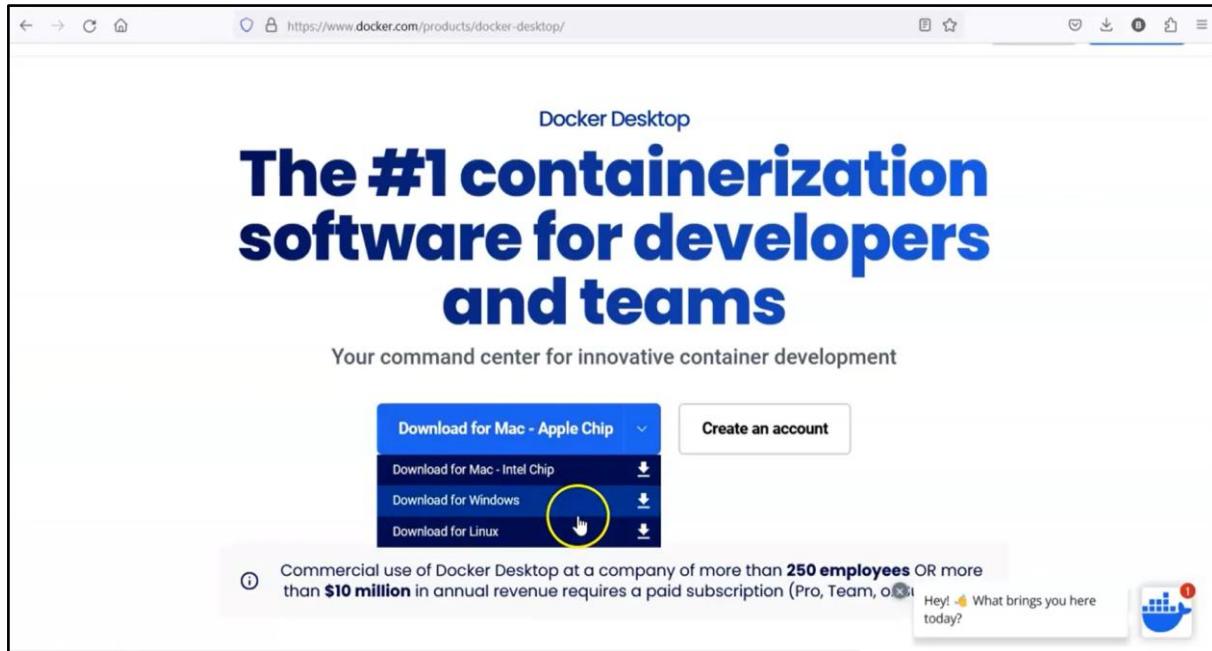


Figure 240. Download Docker version

2. Open File Explorer and double-click on **Docker Desktop Installer.exe**
3. Click on **Ok** to install Docker

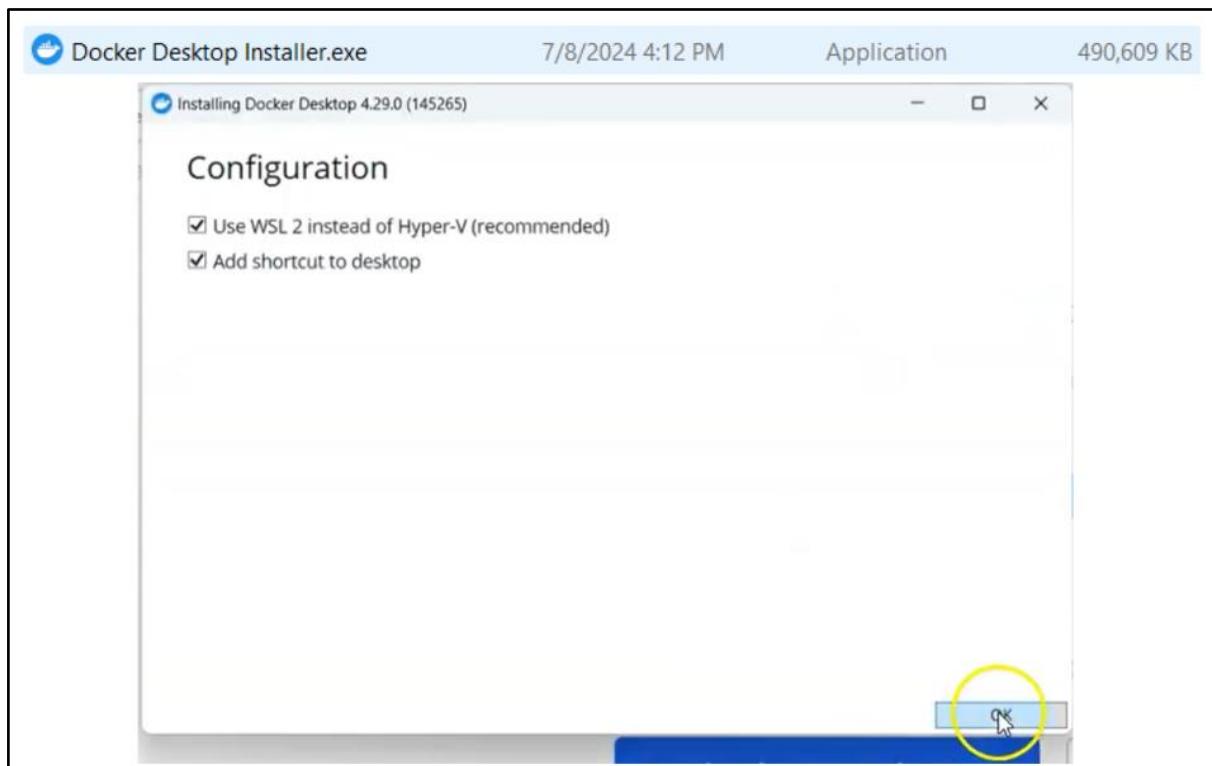


Figure 241. Configuration Docker

4. Click on **Close**

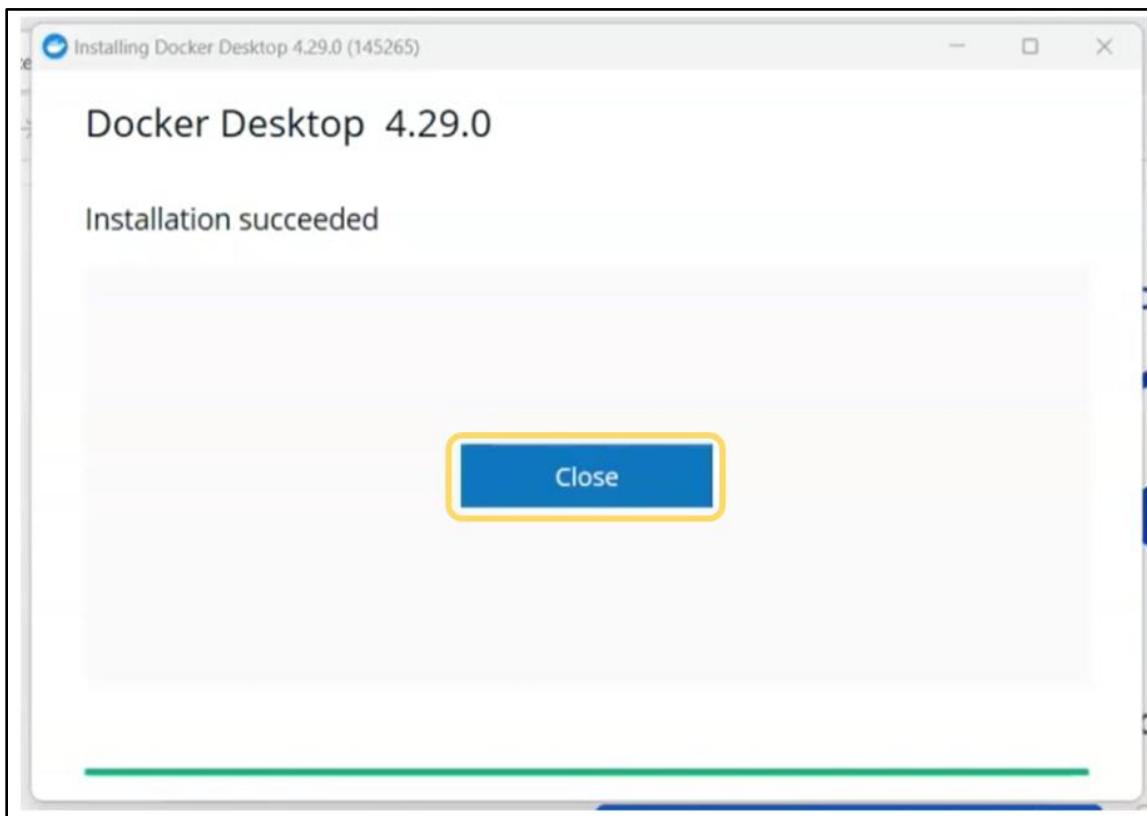


Figure 242. Docker Installer

5. Open **Docker Desktop Application**

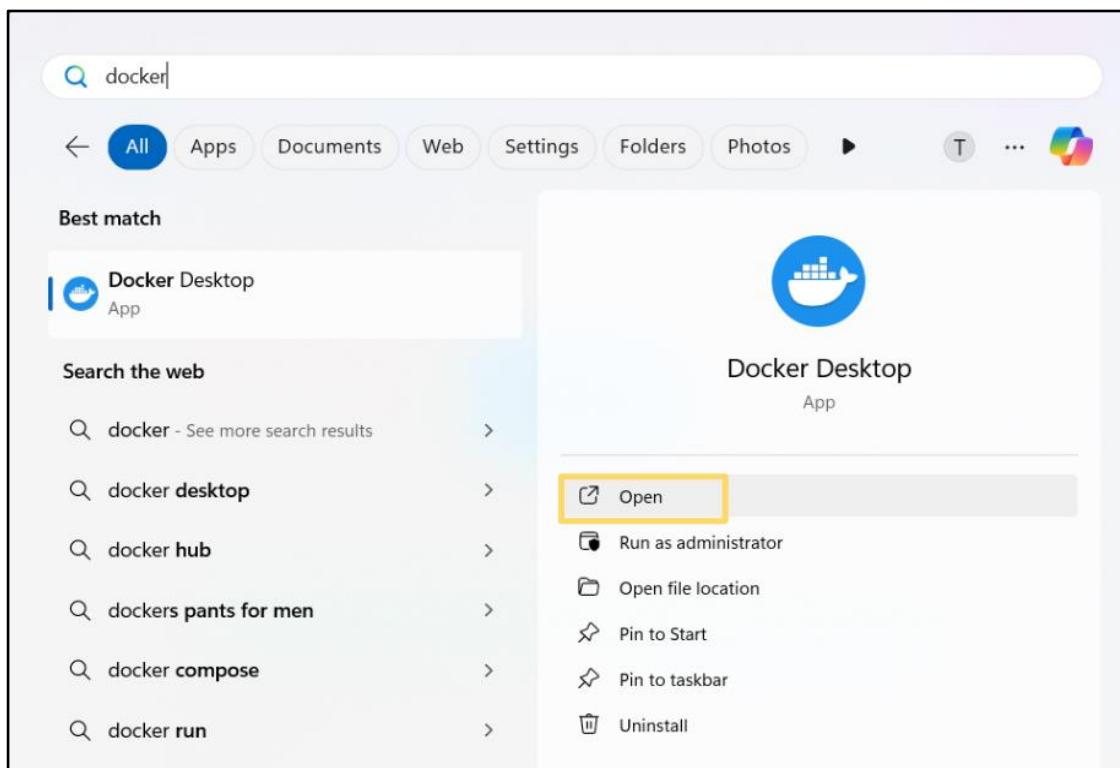


Figure 243. Open Docker Desktop Application

6. Click on **Accept**

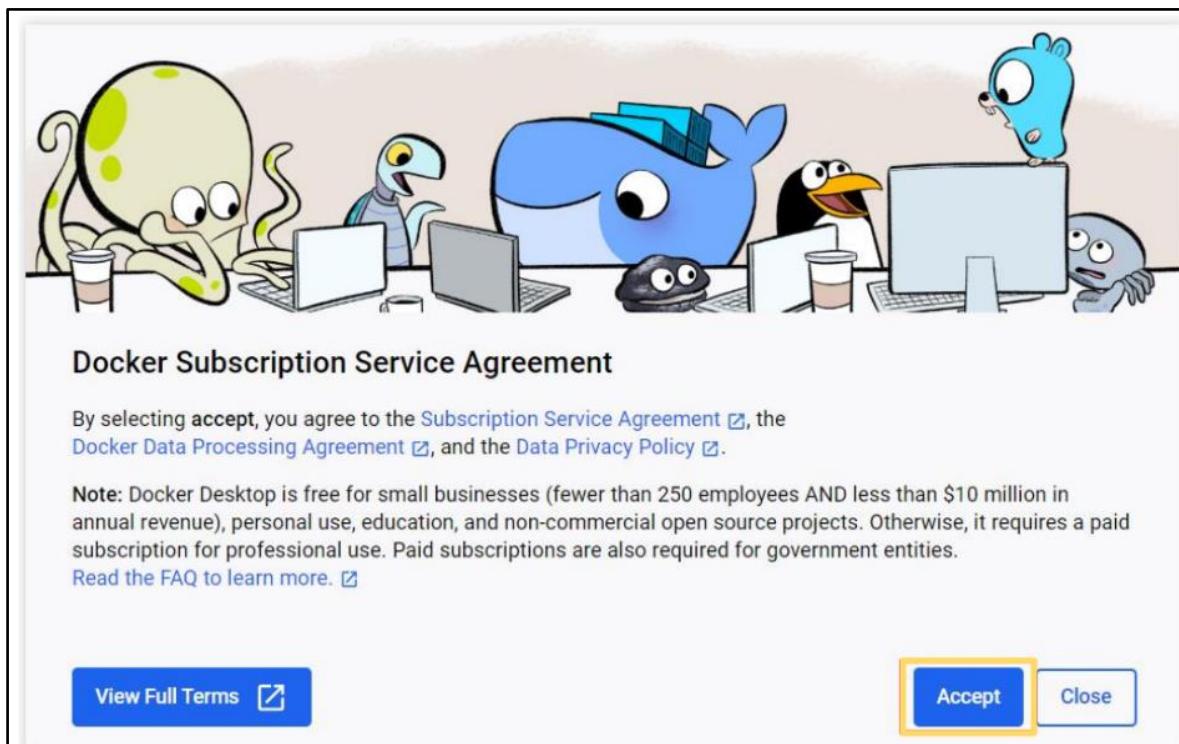


Figure 244. Open Docker

7. Click on **Skip survey**

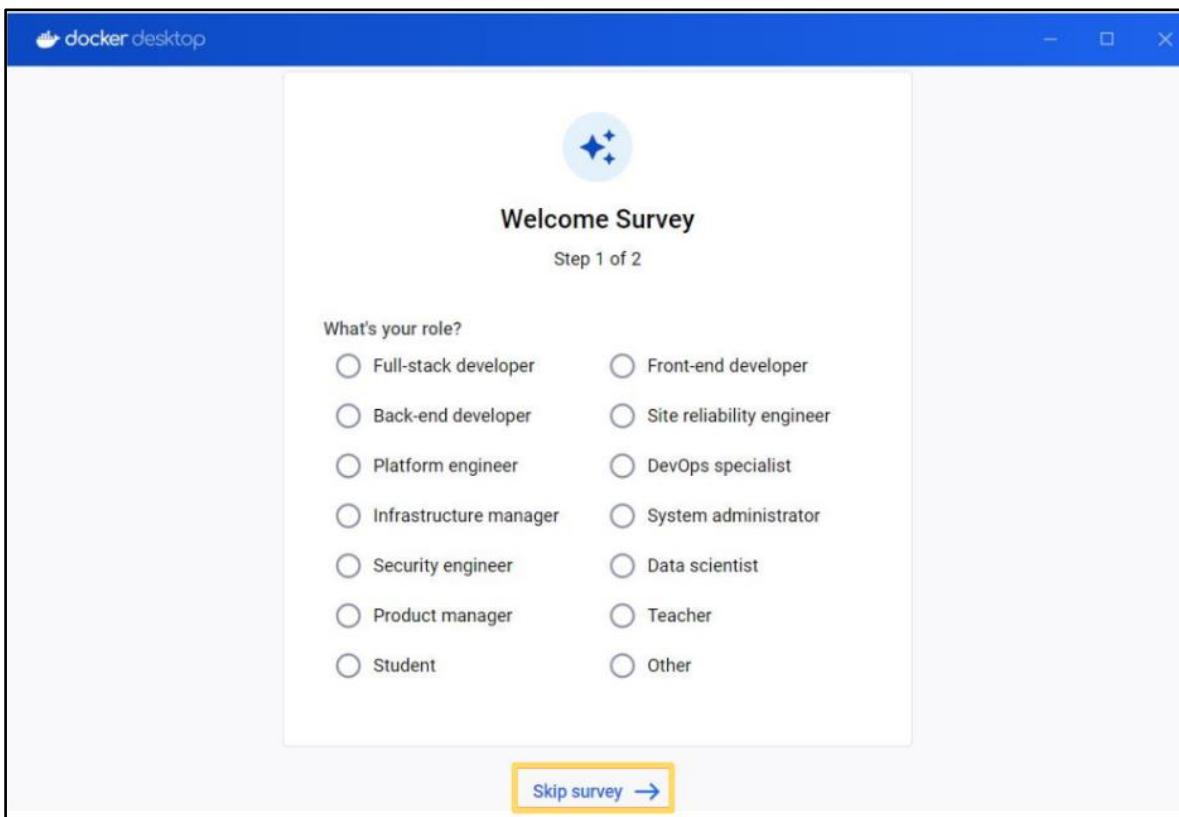


Figure 245. Open Docker

8. Setup completed

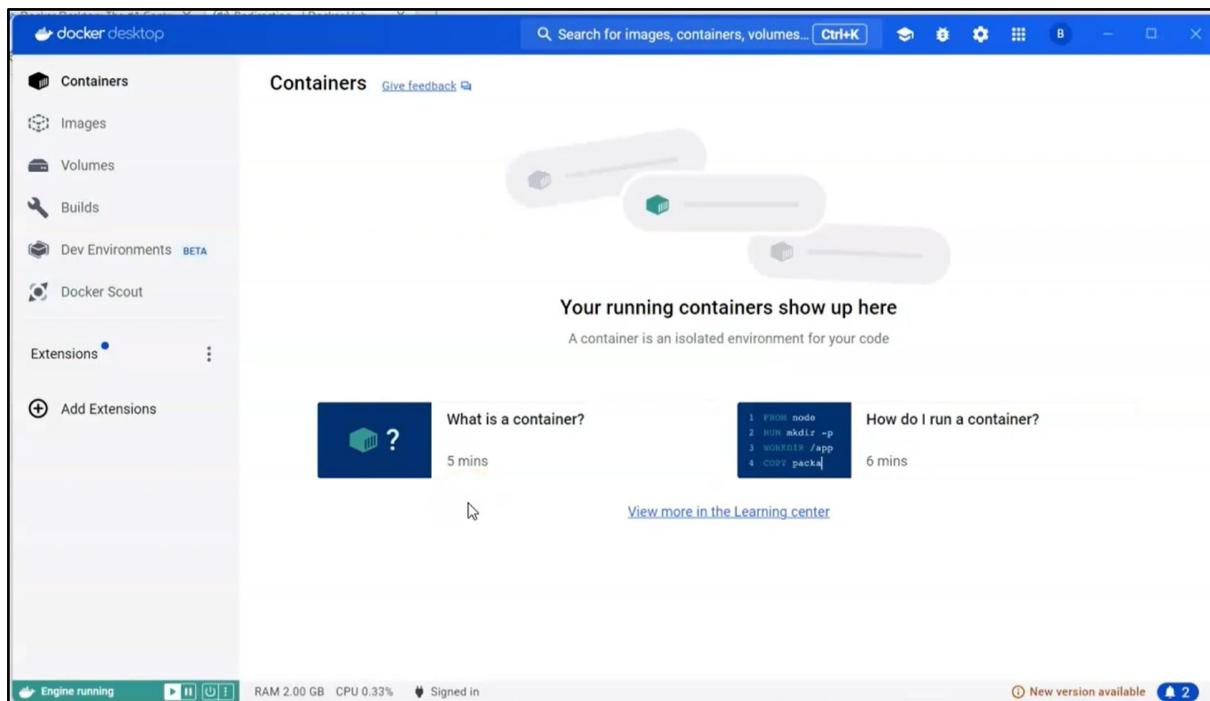


Figure 246. Open Docker successfully

2.2.2 Install and setup WSL 2 (Windows Subsystem for Linux 2)

1. Enable the “windows subsystem for Linux” option in settings
 - + Go to Start. Search for “Turn Windows features on or off.”
 - + Check the option Windows Subsystem for Linux.

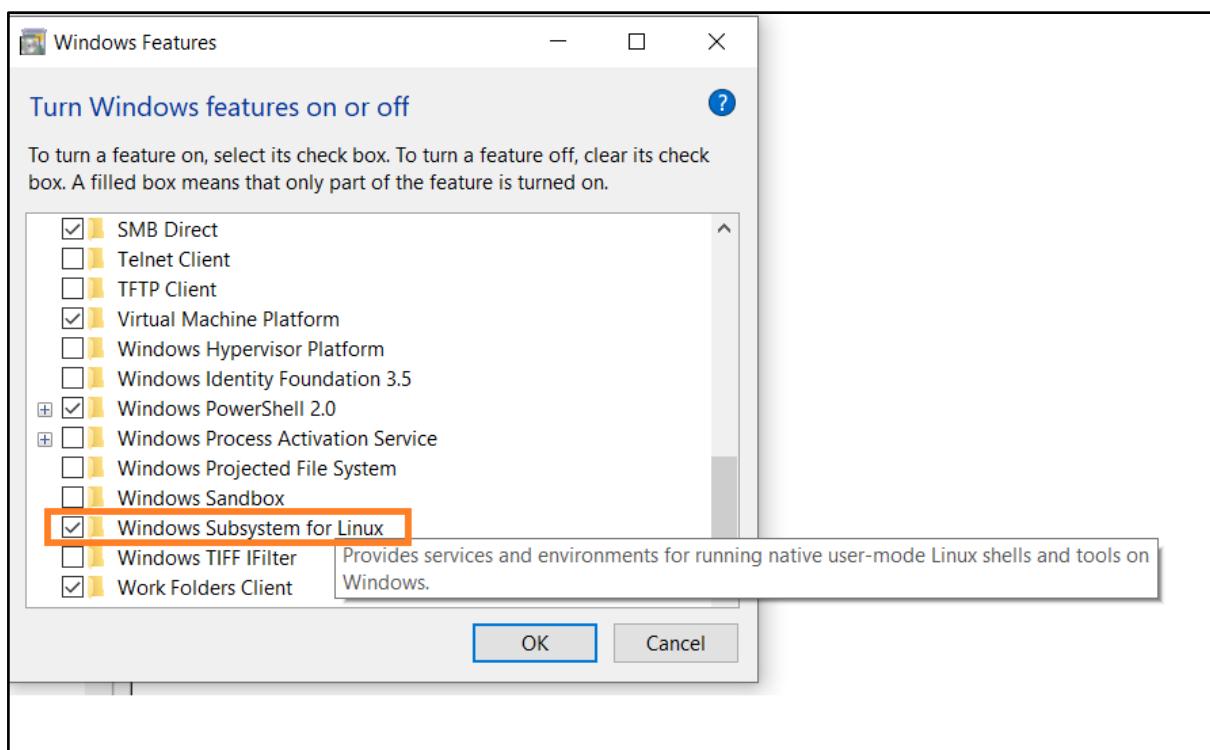


Figure 247. Enable the windows subsystem for Linux option

2. Open Command Prompt as an administrator

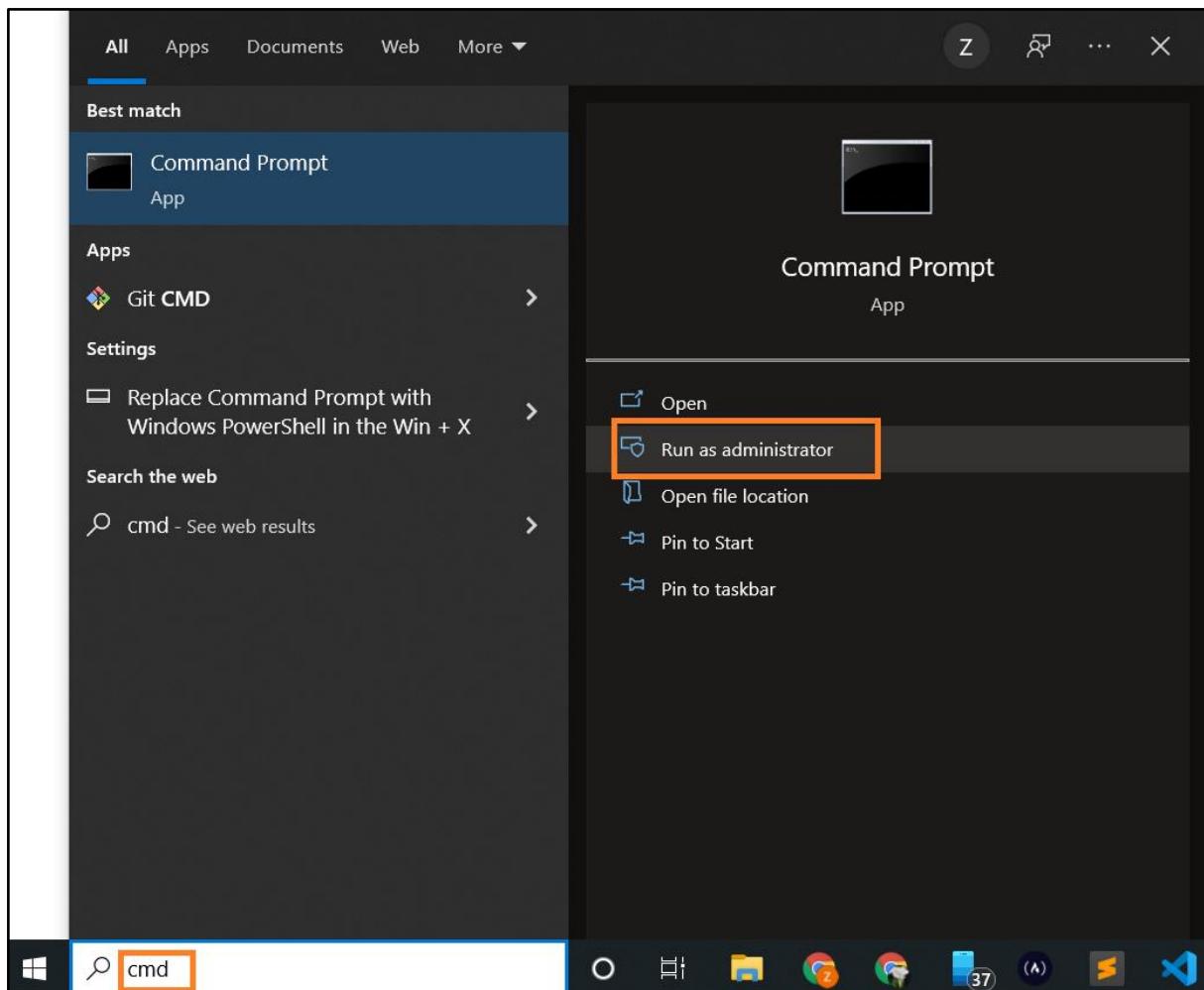


Figure 248. Open Command Prompt

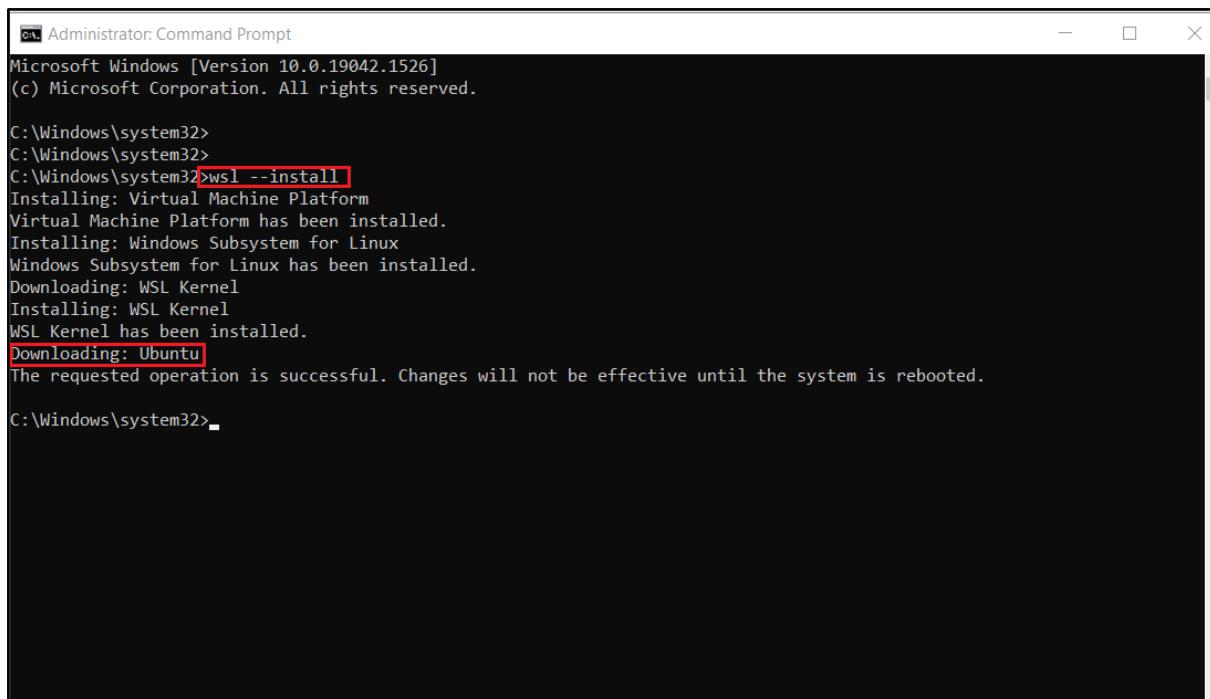
3. Run the installation commands

```
Administrator: Command Prompt - wsl --install
Microsoft Windows [Version 10.0.19042.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
C:\Windows\system32>
C:\Windows\system32>wsl --install
Installing: Virtual Machine Platform
Virtual Machine Platform has been installed.
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Downloading: WSL Kernel
Installing: WSL Kernel
WSL Kernel has been installed.
Downloading: Ubuntu
[===== 38.8%]
```

Figure 249. Run the installation command

4. Restart Windows machine

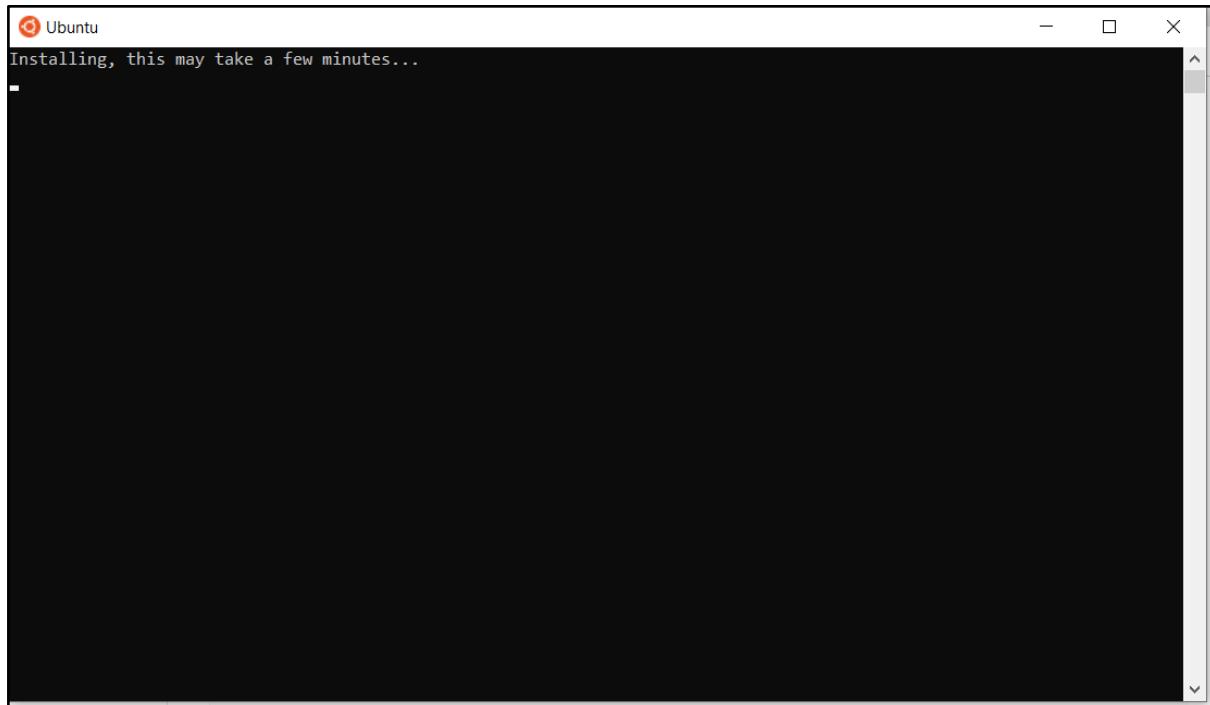


```
C:\> Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
C:\Windows\system32>
C:\Windows\system32>wsl --install
Installing: Virtual Machine Platform
Virtual Machine Platform has been installed.
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Downloading: WSL Kernel
Installing: WSL Kernel
WSL Kernel has been installed.
Downloading: Ubuntu
The requested operation is successful. Changes will not be effective until the system is rebooted.

C:\Windows\system32>
```

Figure 250. Require reboot system



```
Ubuntu
Installing, this may take a few minutes...
```

Figure 251. Reboot system

5. Enter username and password

```
zaira@Zaira: ~
Enter new UNIX username: zaira
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.10.16.3-microsoft-standard-WSL2 x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 System information as of Mon Feb 14 15:33:29 PKT 2022

 System load: 0.0 Processes: 8
 Usage of /: 0.4% of 250.98GB Users logged in: 0
 Memory usage: 2% IPv4 address for eth0: 172.20.230.64
 Swap usage: 0%

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

Figure 252. Enter username and password

6. Launch Ubuntu by searching from the start menu

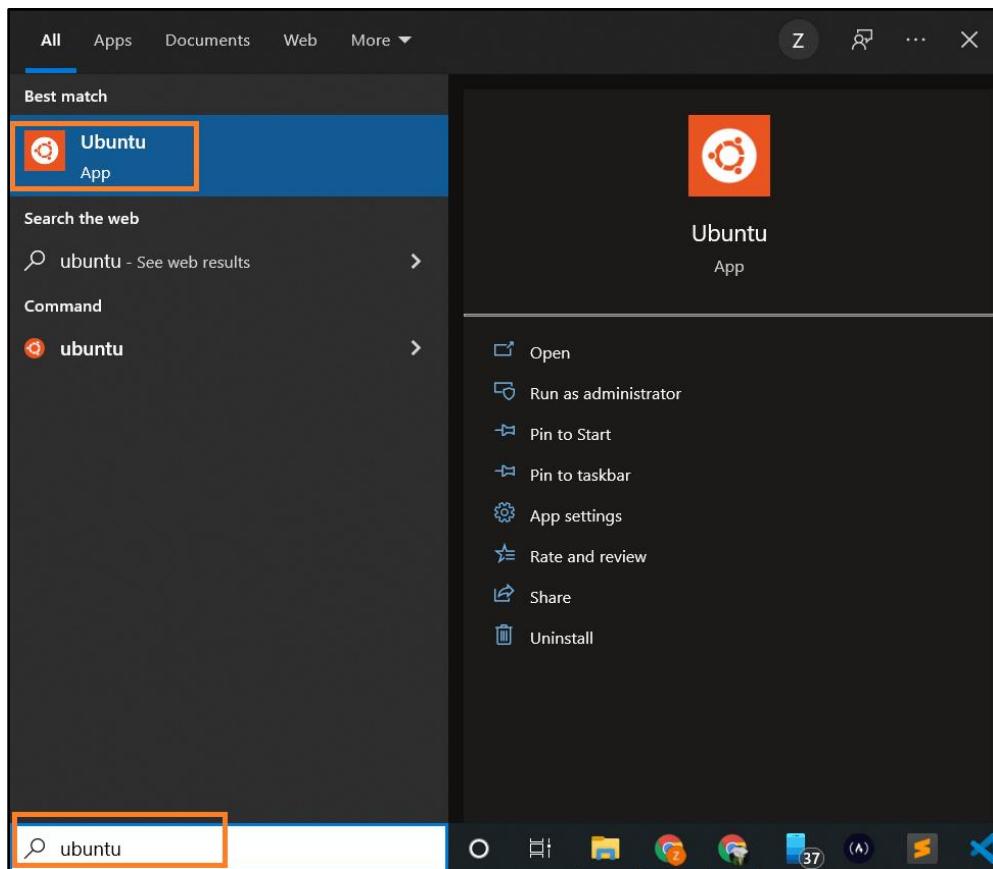
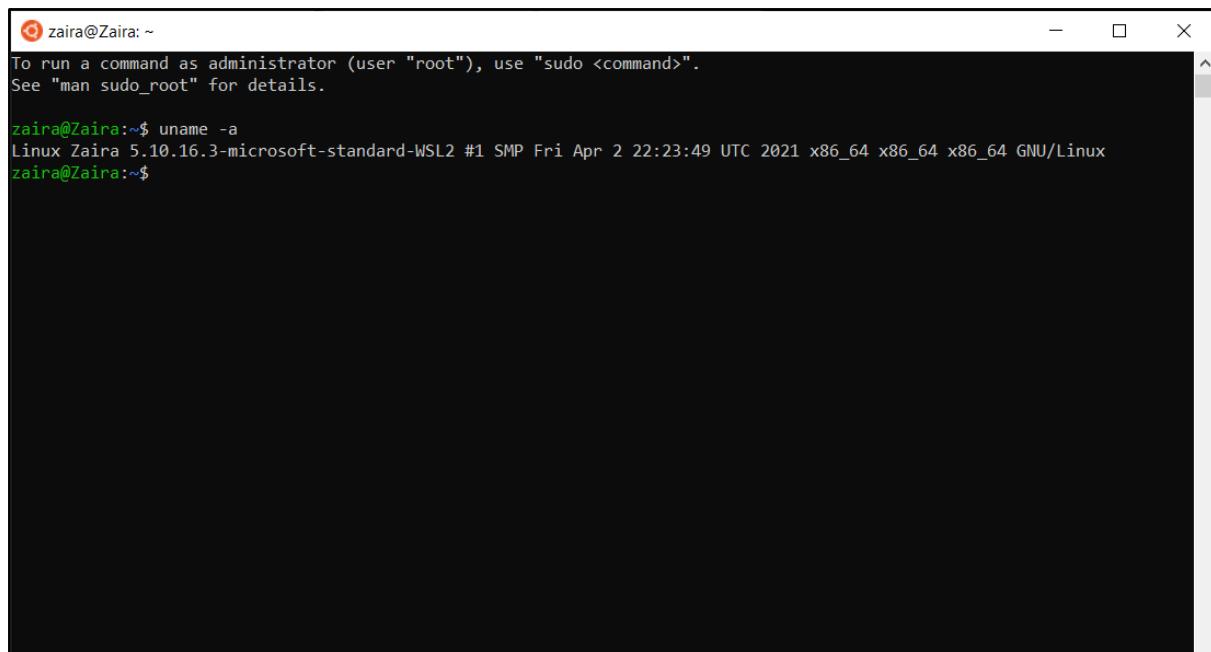


Figure 253. Open Ubuntu Application

7. Ubuntu instance launched



```
zaira@Zaira: ~
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

zaira@Zaira:~$ uname -a
Linux Zaira 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
zaira@Zaira:~$
```

Figure 254. Launch Ubuntu

2.2.3 Install CUDA Toolkit on Windows and WSL

1. Installing the Latest NVIDIA Display Driver
 - a. Search the appropriate driver for NVIDIA product

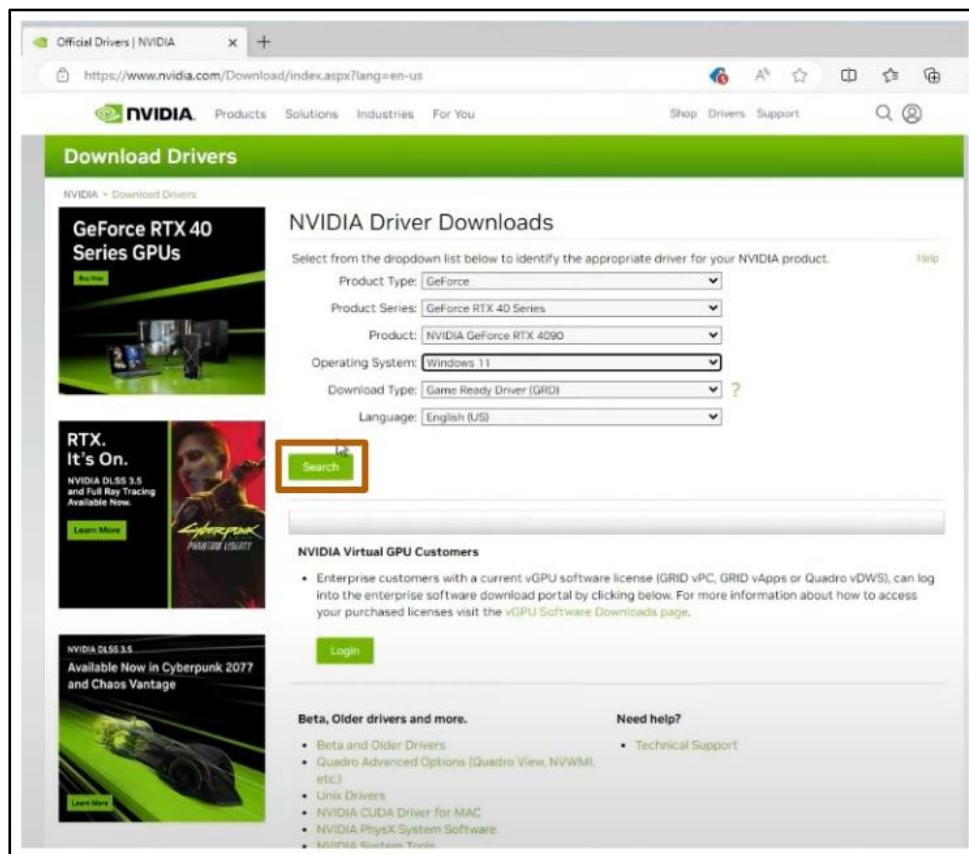


Figure 255. Choose the appropriate NVIDIA driver

b. Download the latest version of the NVIDIA driver

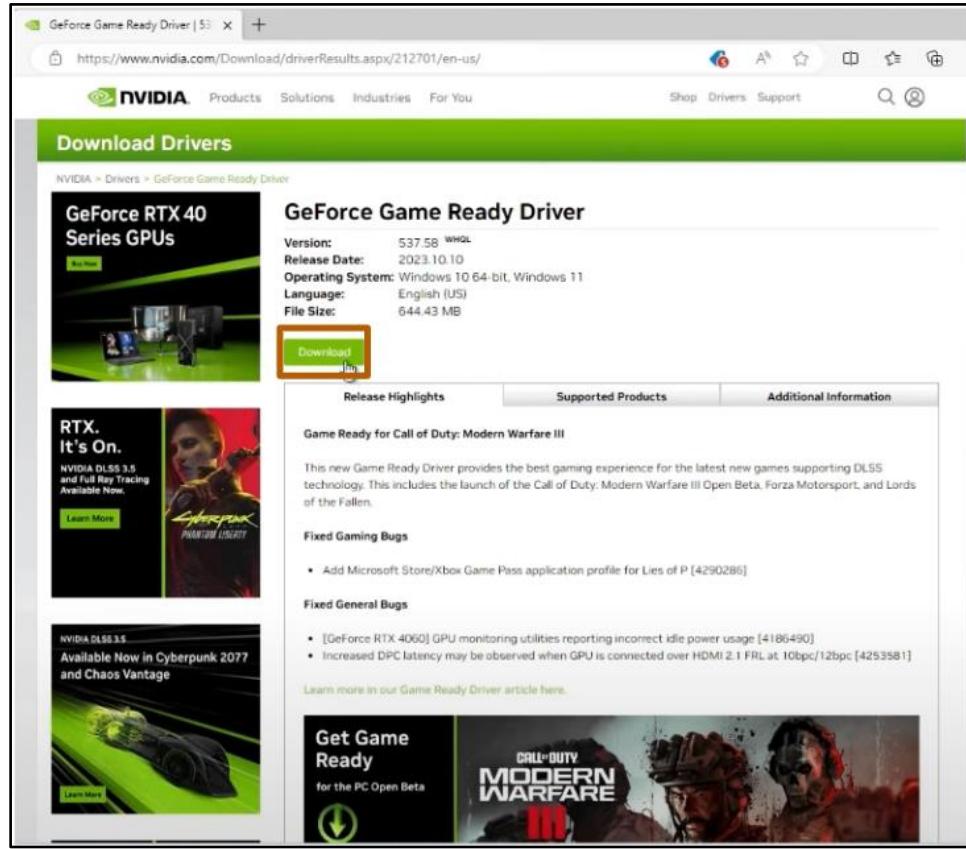


Figure 256. Download the NVIDIA driver

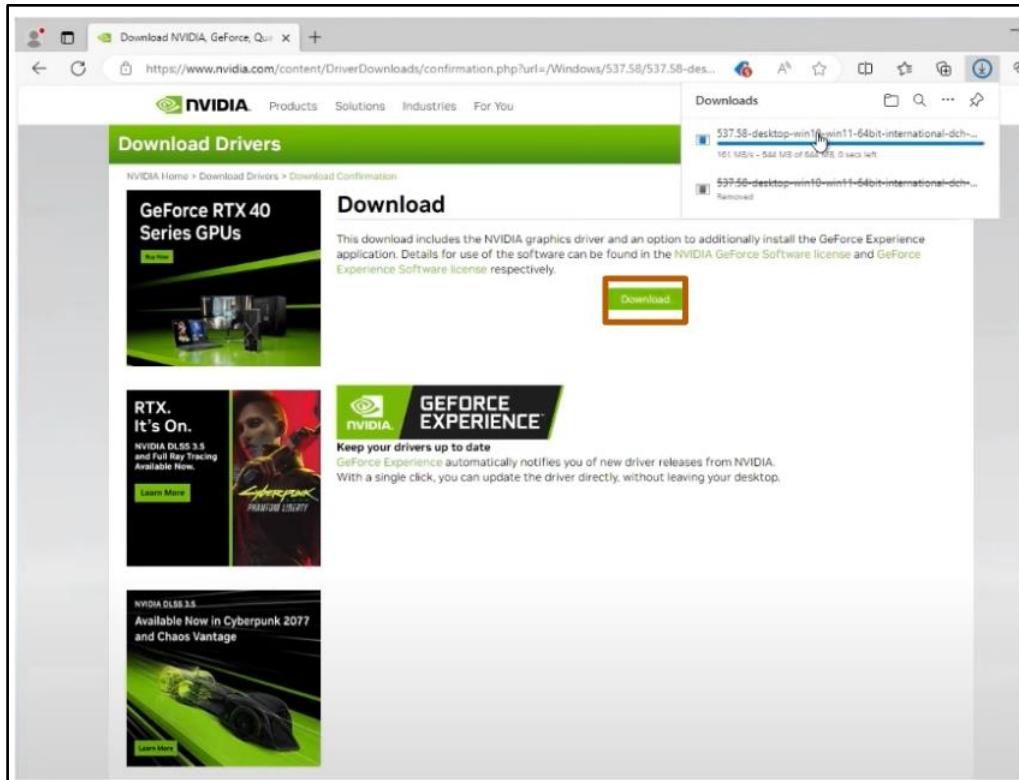


Figure 257. Download the NVIDIA driver

- c. Open **File Explorer** and double-click on file has end path .exe
- d. Click on **AGREE AND CONTINUE**

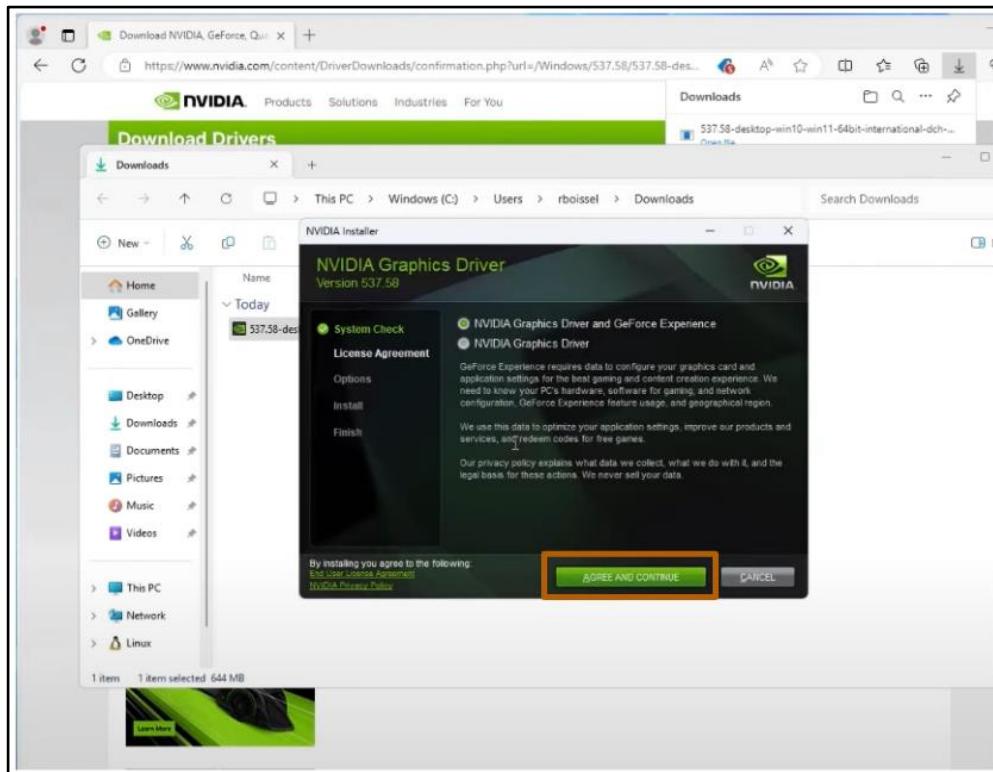


Figure 258. Install the NVIDIA driver

- e. Click on **Next**

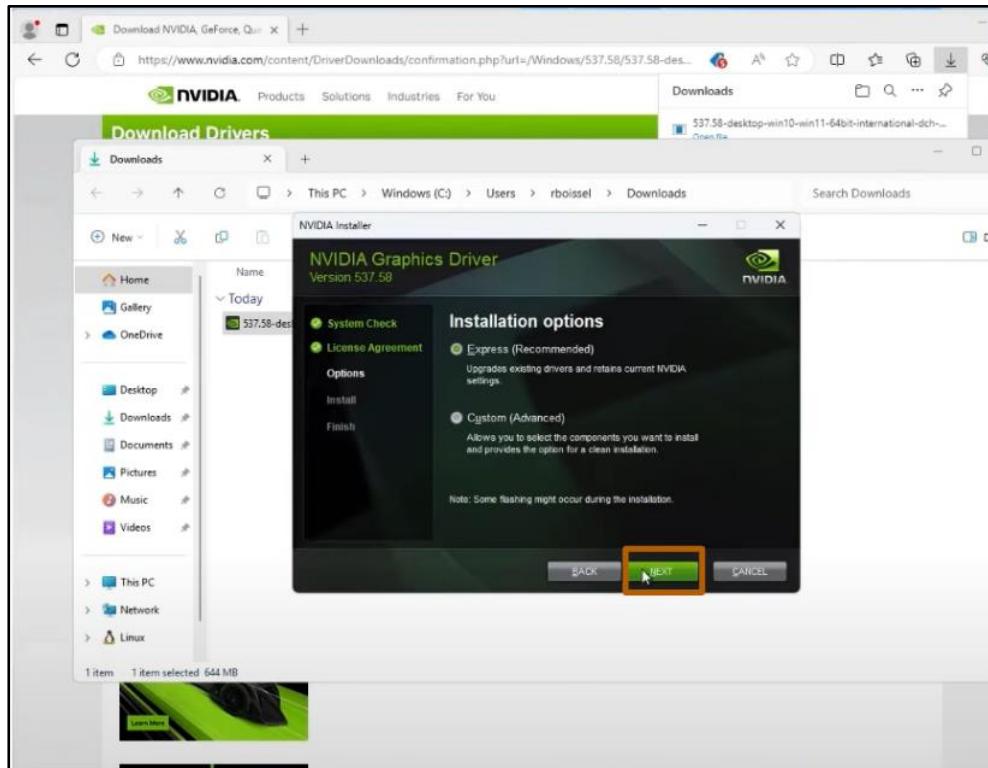


Figure 259. Install the NVIDIA driver

f. The driver is installed, click **Close**

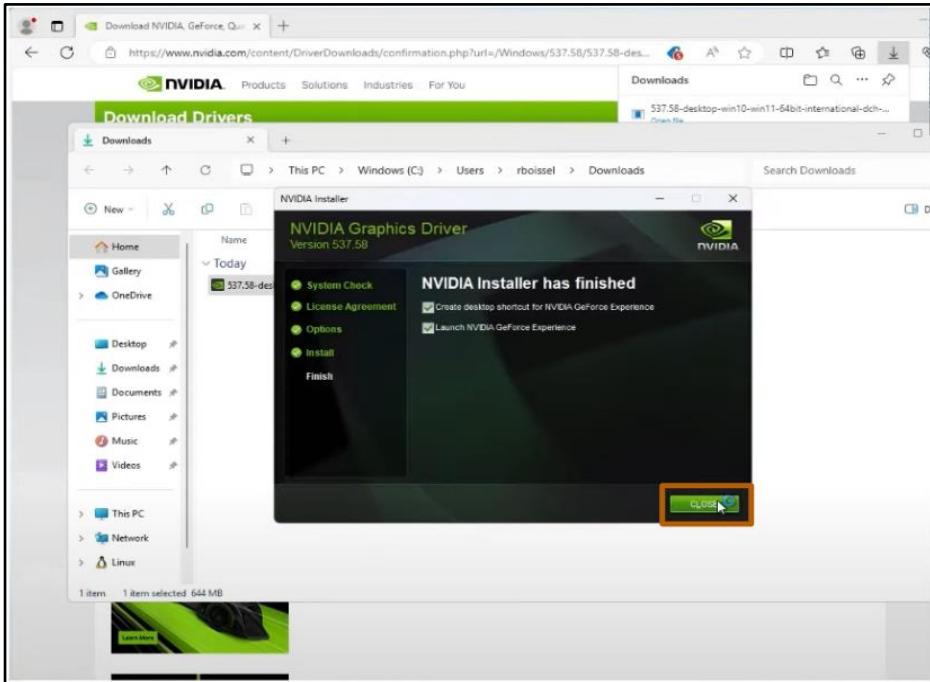


Figure 260. Install the NVIDIA driver finished

2. Installing CUDA Toolkit on WSL

- a. Open **Ubuntu Promt**, type “`sudo apt update`” and press Enter

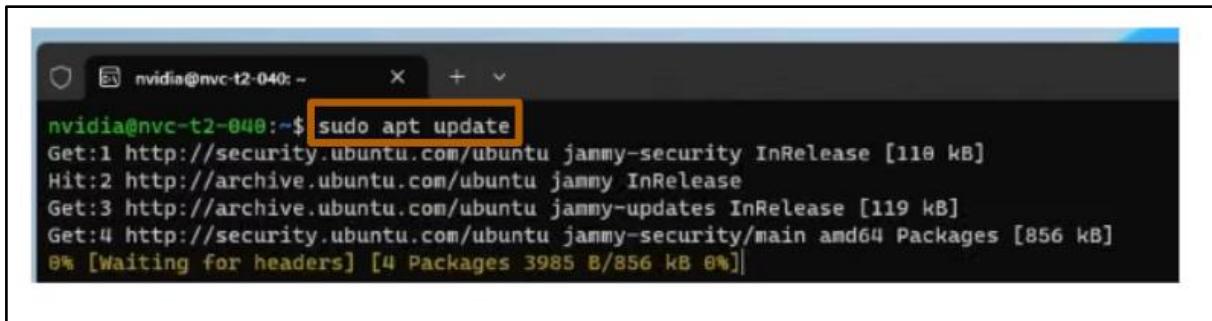


Figure 260. Update available packages

- b. Next, type “`sudo apt install build-essential`” and press Enter



Figure 261. Install build-essentials packages

- c. Next, download the CUDA toolkit, choose attribute like below and run the Installation Instructions

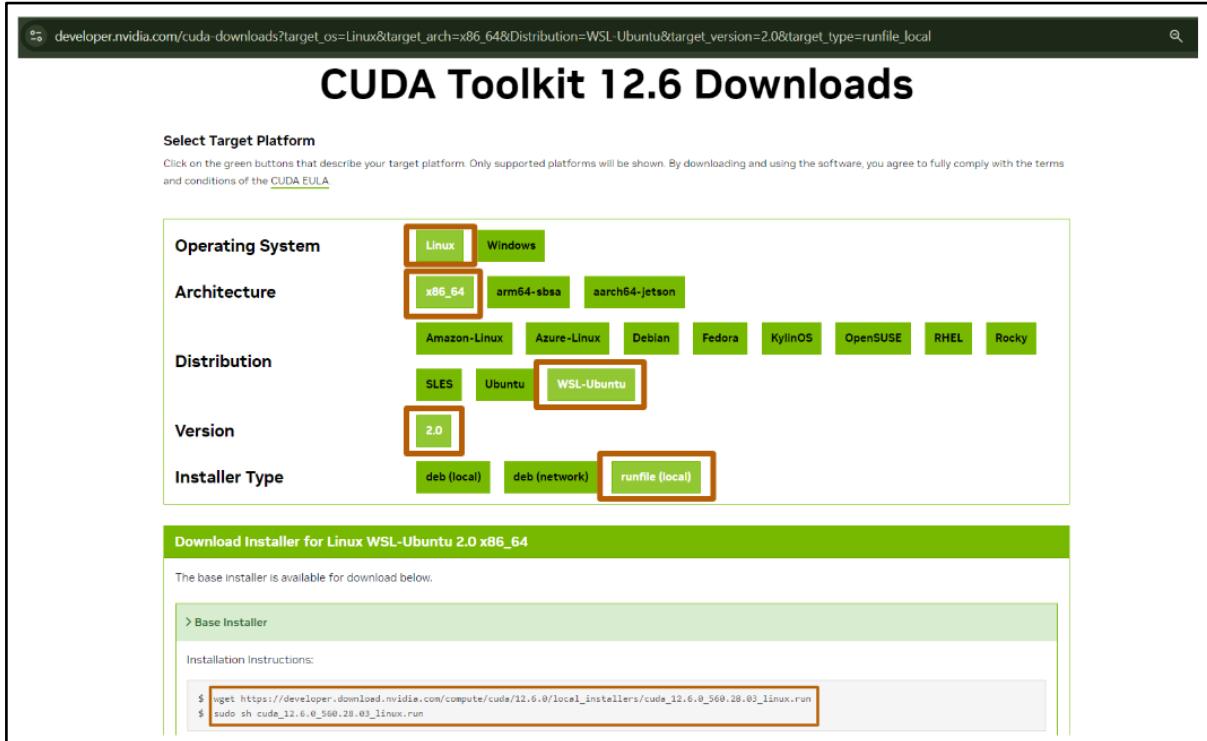


Figure 262. Download CUDA Toolkit

```
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
/sbin/ldconfig.real: /usr/lib/wsl/!libcuda.so.1 is not a symbolic link
nvidia@nvc-t2-040:~$ wget https://developer.download.nvidia.com/compute/cuda/12.2.2/local_installers/cuda_12.2.2_535.104.05_linux.run
2023-10-11 10:31:48  https://developer.download.nvidia.com/compute/cuda/12.2.2/local_installers/cuda_12.2.2_535.104.05_linux.run
Resolving developer.download.nvidia.com (developer.download.nvidia.com)... 152.195.19.142
Connecting to developer.download.nvidia.com (developer.download.nvidia.com)|152.195.19.142|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4344134690 (4.0G) [application/octet-stream]
Saving to: 'cuda_12.2.2_535.104.05_linux.run'

cuda_12.2.2_535.104.05_linux.run    100%[=====] 4.04G  202MB/s   in 23s
2023-10-11 10:52:04 (177 MB/s) - 'cuda_12.2.2_535.104.05_linux.run' saved [4344134690/4344134690]
nvidia@nvc-t2-040:~$ sudo sh cuda_12.2.2_535.104.05_linux.run
```

Figure 263. Download CUDA Toolkit

3. User Manual

3.1 Terms and Definitions

Table 237. Terms and Definitions

No.	Term And Definitions	Descriptions
1	Admin	A user who is allowed to use all features of the application.
2	Member	A user who is allowed to use only some view features of the application.
3	Bounding Box	A rectangular outline produced by the object detection model to highlight an object within a frame. Different colors represent various object types in the debug live view.
4	Event	A duration beginning when a tracked object appears in the frame and ending when it exits, including any stationary time. Events are logged if they are true positives and meet the criteria for capturing snapshots or recordings.

No.	Term And Definitions	Descriptions
5	Motion Mask	Motion masks prevent detection of motion in masked areas from triggering object detection but do not prevent detection of objects due to motion in nearby areas, such as camera timestamps, skies, or tree tops.
6	Object Mask	Object filter masks disregard any bounding boxes where the bottom center overlaps the masked area, marking them as false positives to be ignored.
7	Motion	Occurs when pixels in the current camera frame differ from previous frames. A cluster of differing pixels is shown by a red motion box in the live debug view.
8	Region	A segment of the camera frame sent for object detection due to motion, active objects, or sometimes stationary objects. Represented by green boxes in the debug live view.
9	Zone	Designated areas of interest used for notifications and to limit the areas where Vigision generates events.

3.2 System Requirements

Recommended System Requirements:

- Operating system: Windows 11
- Software: Docker Desktop v4.29.0, WSL 2
- Storage: 100GB SSD available
- RAM: 8GB or higher
- CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz (8 CPUs), ~2.4GHz or higher.
- GPU: NVIDIA GeForce GTX 1660 Ti
- Database: SQLite v3.34.1
- Browser: Google Chrome 90, Safari 14.0.3 (macOS), Microsoft Edge 89.0 or newer.

3.3 Application Usage

3.3.1 Overview

Our system has 2 actors: Admin and Member

Admin has the following features:

1. Login
2. Logout
3. Manage users
4. Get shareable link
5. View live cameras
6. Manage cameras
7. Manage camera groups
8. Edit camera group layout
9. View in full screen
10. View detail of live camera
11. Enable/disable detection
12. Enable/disable recording
13. Enable/disable snapshots
14. View defined masks and zones
15. Manage motion masks
16. Manage zones

17. Manage object masks
18. Fine-tune motion detection
19. Debug
20. View alerts/detections
21. View motion events
22. View event recording
23. Mark events as reviewed
24. Delete events
25. View exports
26. View export detail
27. Export event recording
28. Export camera recording
29. Download export
30. Rename export
31. Delete export
32. Config
33. View system metrics
34. Restart application
35. Change application theme
36. Forgot password

Member has the following features:

1. Login
2. Logout
3. View live cameras
4. View in full screen
5. View detail of live camera
6. View system metrics
7. Change application theme
8. Forgot password

3.3.2 User guide for admin

3.3.2.1 Login

- Step 1: Open your preferred web browser and navigate to the Vigision application.

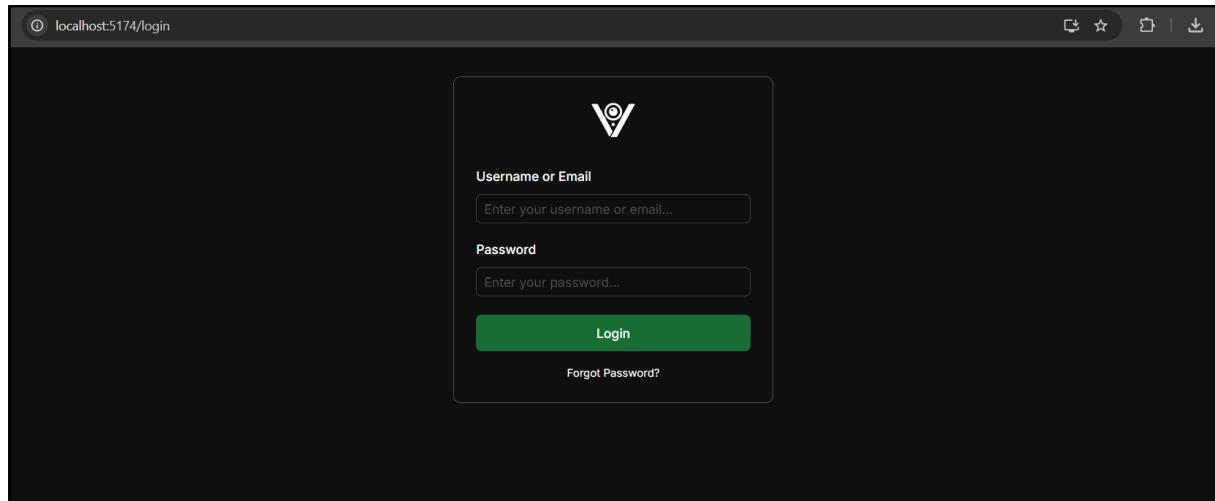


Figure 264. Login

- Step 2: Enter Your Username or Email Address. (If first time, the default Username of admin is “admin”)

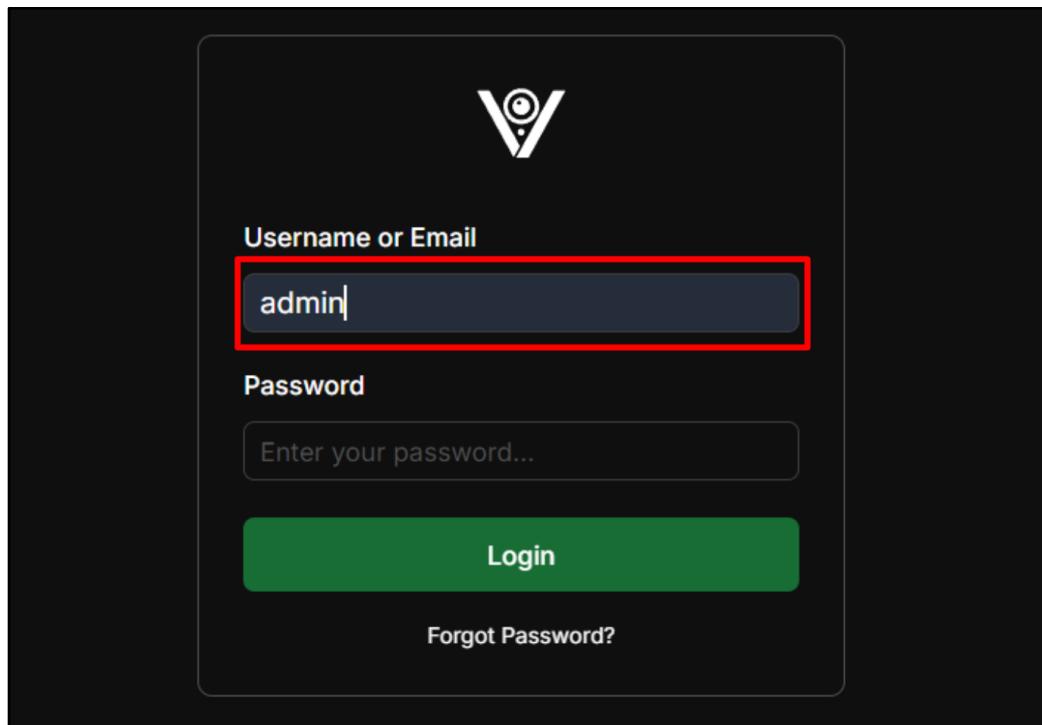


Figure 265. Login

- Step 3. Enter Your Password. (If first time, the default Password of admin is “admin”)

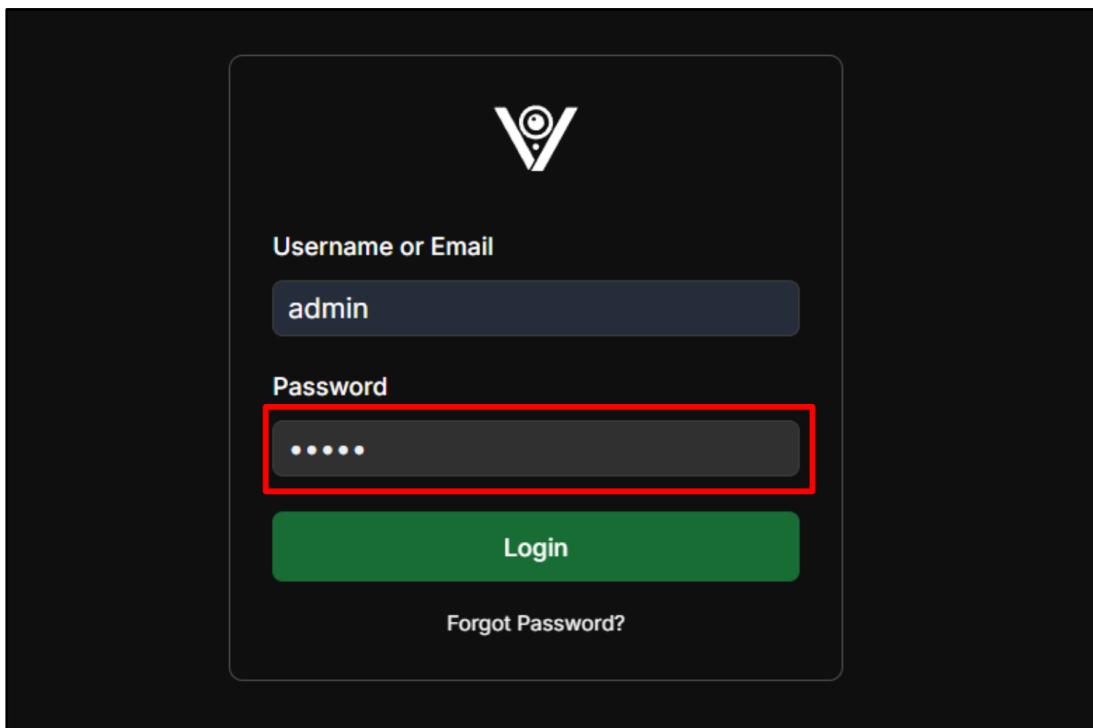


Figure 266. Login

- Step 4. Click on the Login Button. You will be logged in successfully and see the “Live Dashboard” page.

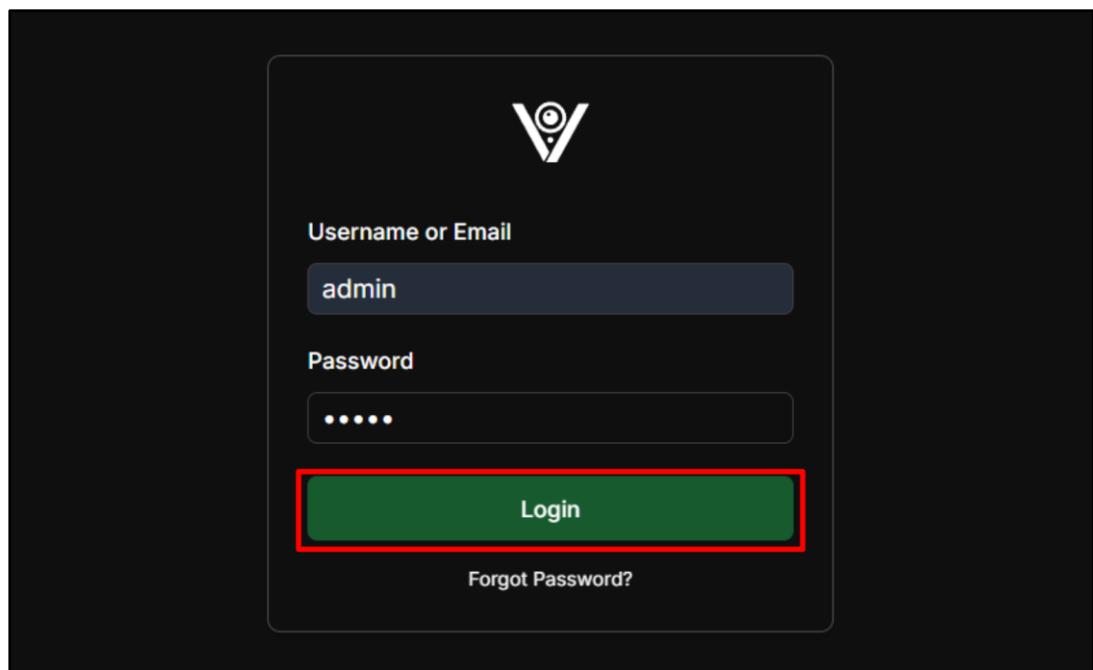


Figure 267. Login

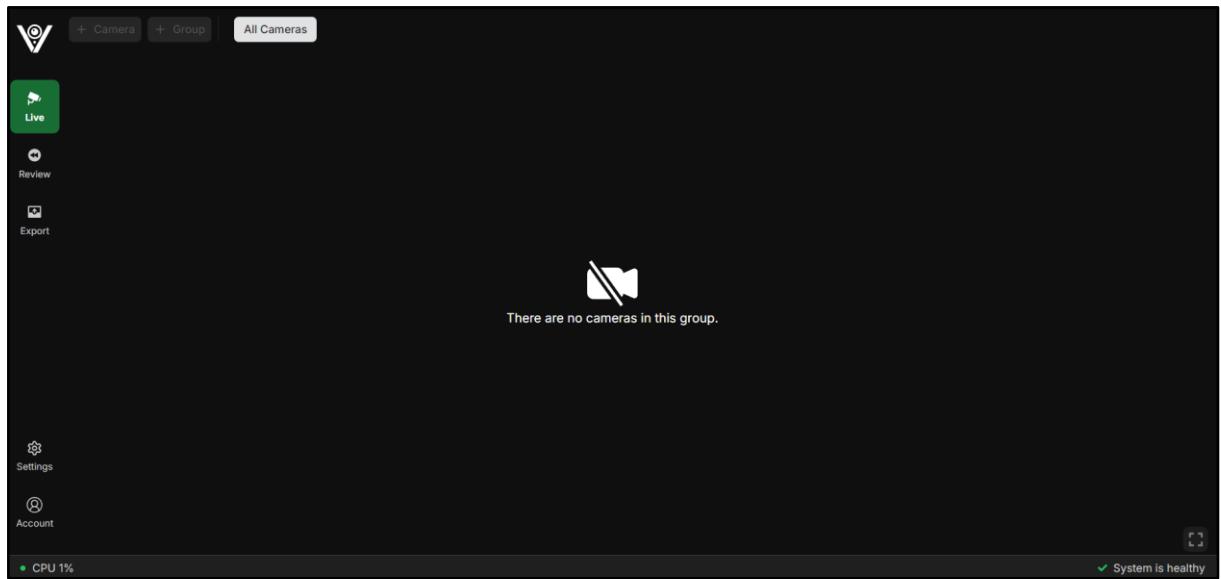


Figure 268. Login successfully

3.3.2.2 Logout

To log out, click on the button “Account” at the bottom left corner.

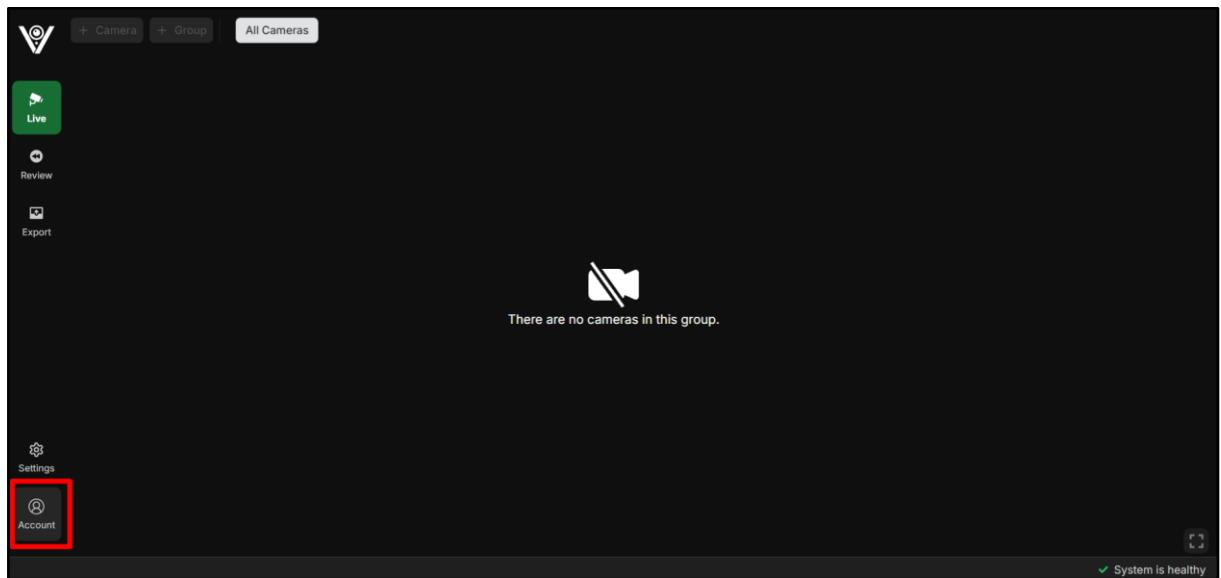


Figure 269. Logout

Next, click on the button “Logout”

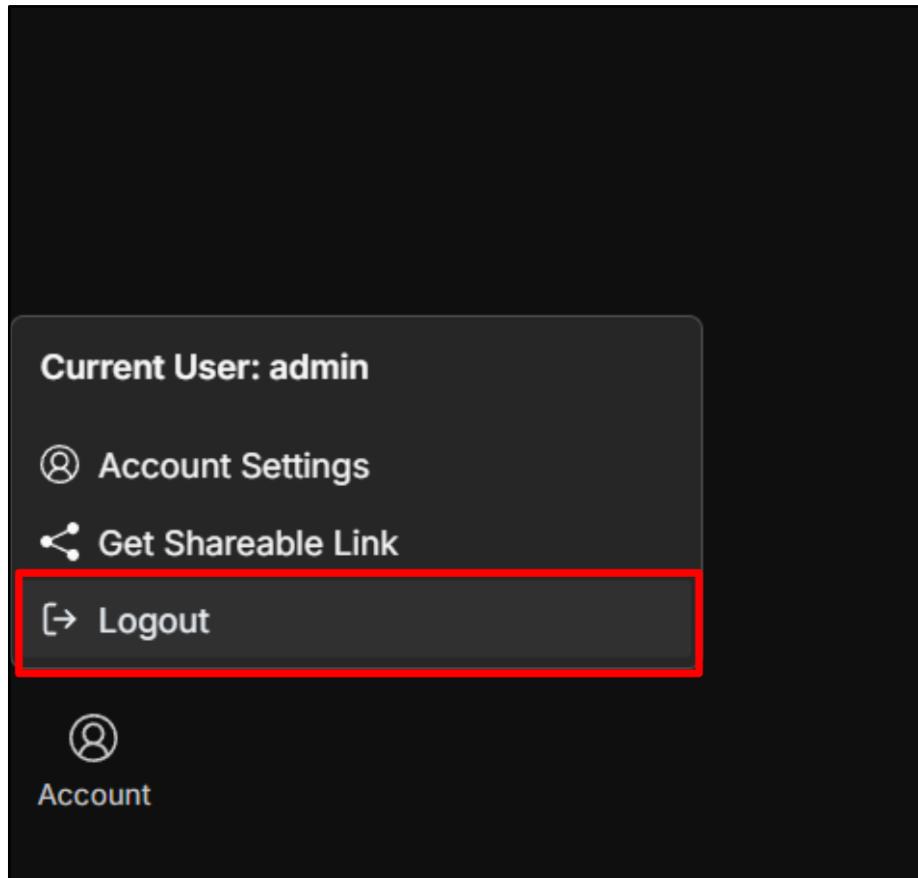


Figure 270. Logout

You have logged successfully.

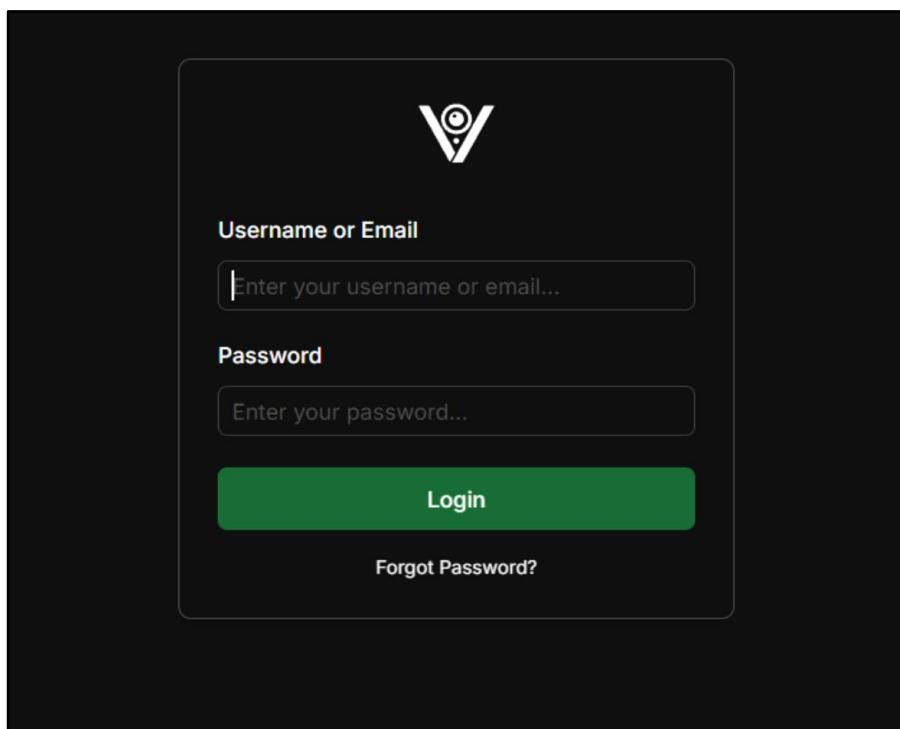


Figure 271. Logout successfully

3.3.2.3 Manage users

3.3.2.3.1 View list of users

Way 1:

First, click on the button “Account” → “Account Settings”

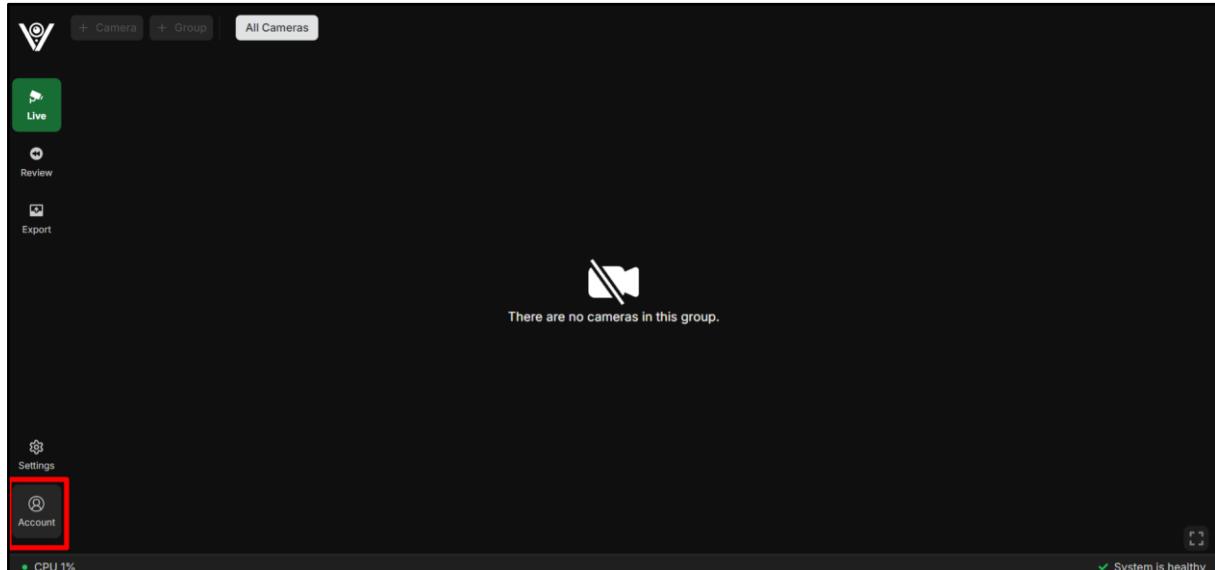


Figure 272. Manage users - View list of users

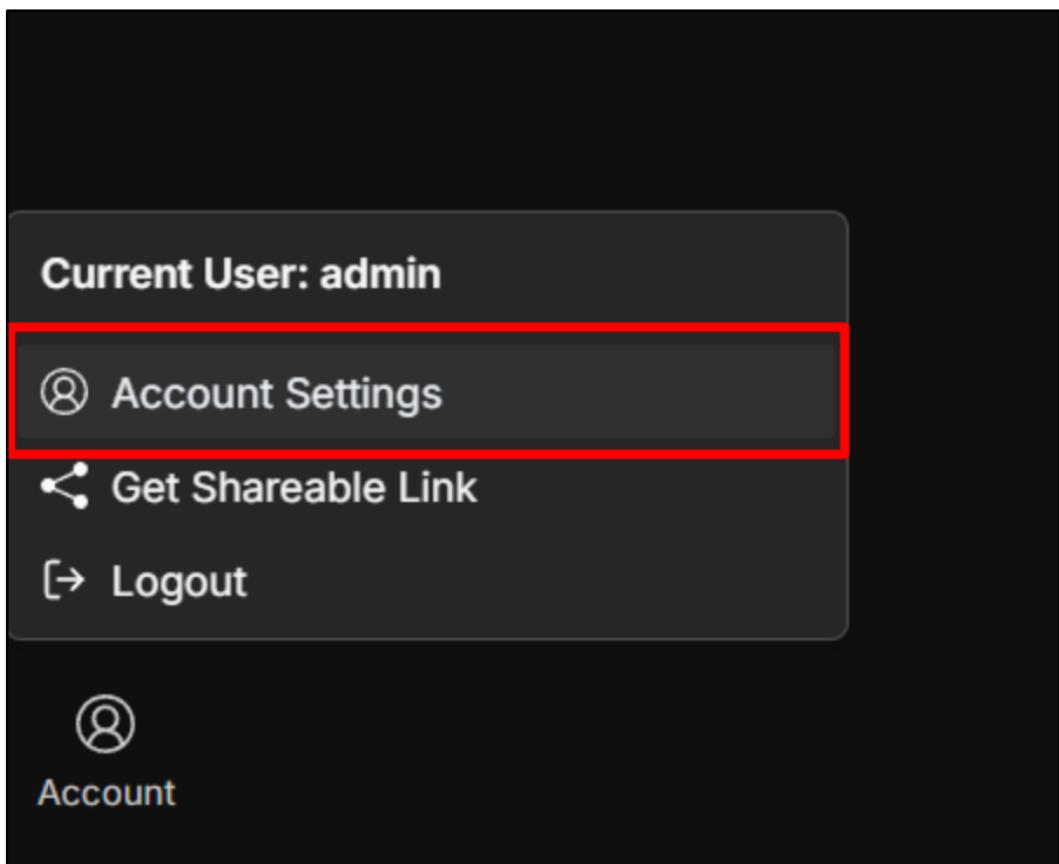


Figure 273. Manage users - View list of users

Way 2:

First, click on the button “Settings” → “Settings”

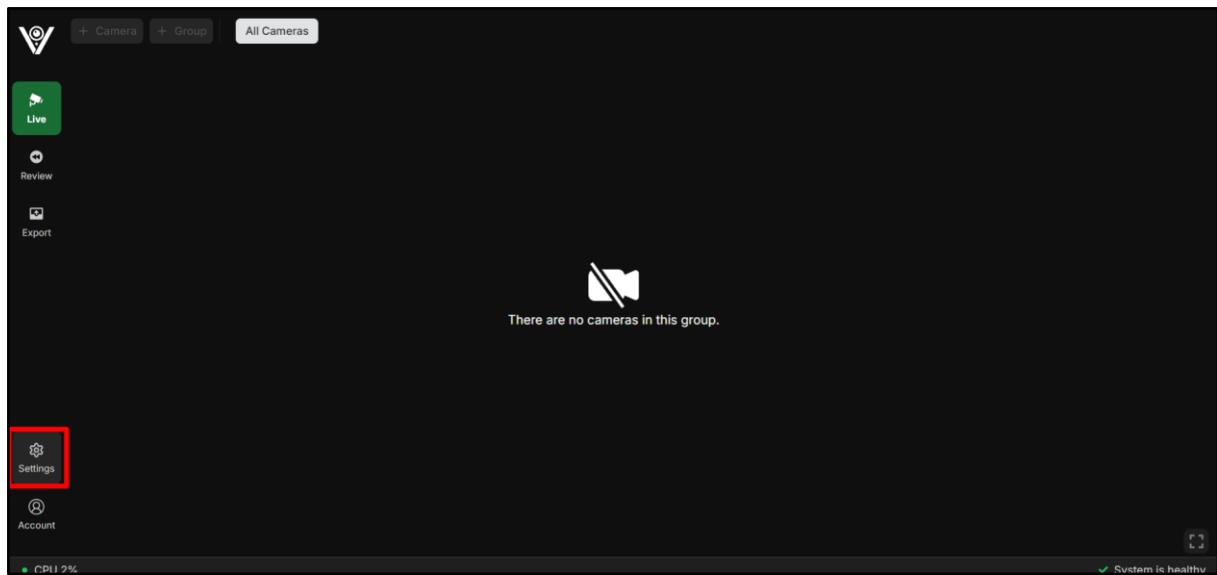


Figure 274. Manage users - View list of users

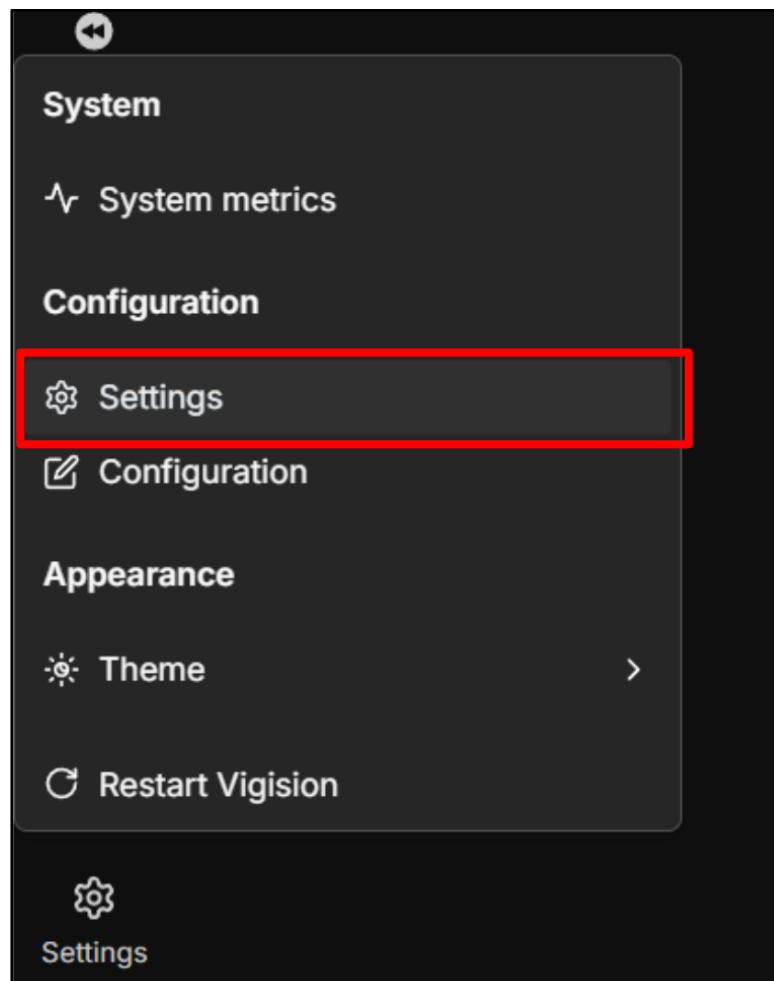


Figure 275. Manage users - View list of users

After that, open tab “Users”

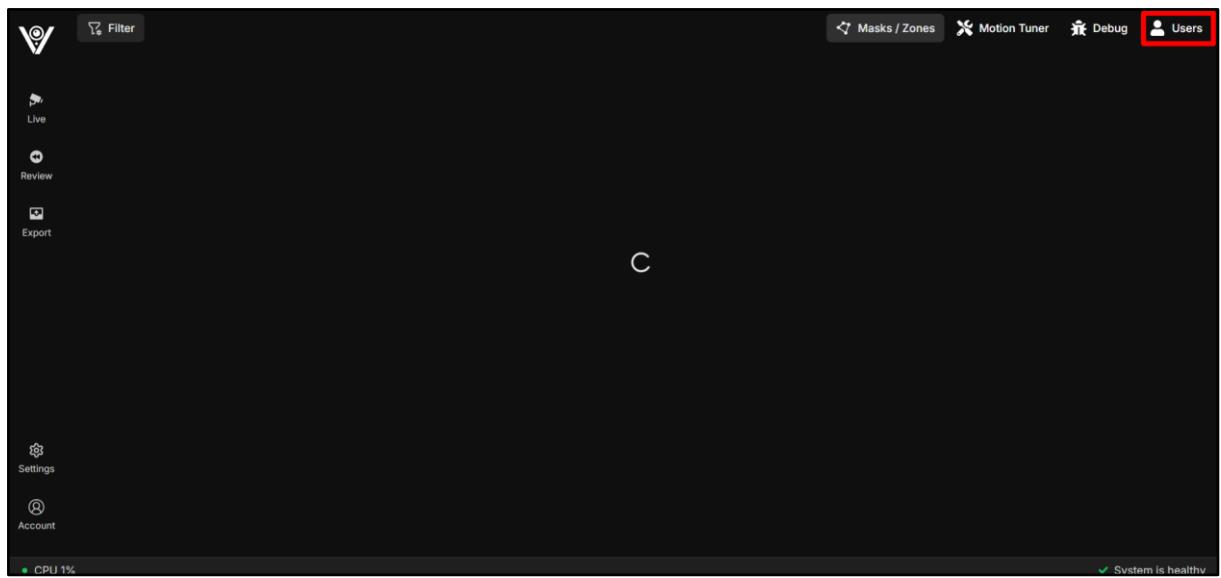


Figure 276. Manage users - View list of users

If there exists no user:

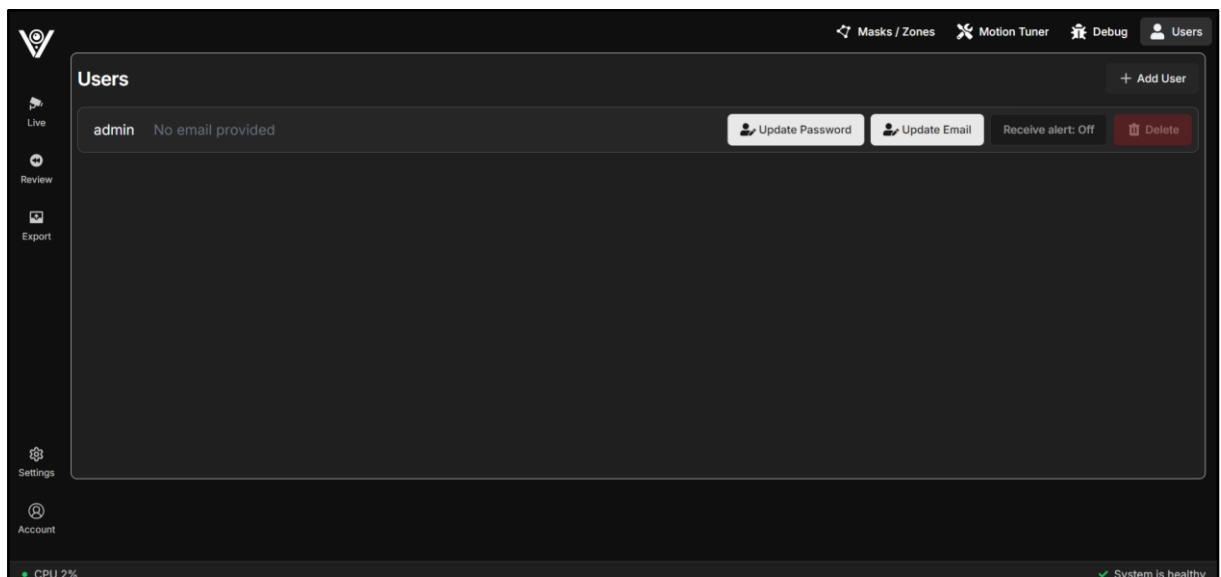


Figure 277. Manage users - View list of users

If there exists users:

The screenshot shows a dark-themed user management interface. On the left is a sidebar with icons for Live, Review, Export, Settings, and Account. The main area is titled 'Users' and lists three entries:

User	Email	Action Buttons
admin	No email provided	[Update Password] [Update Email] [Receive alert: Off] [Delete]
newname	phucphan1421@gmail.com	[Update Password] [Update Email] [Receive alert: Off] [Delete]
newname_2	phucphce171166@fpt.edu.vn	[Update Password] [Update Email] [Receive alert: Off] [Delete]

Figure 278. Manage users - View list of users

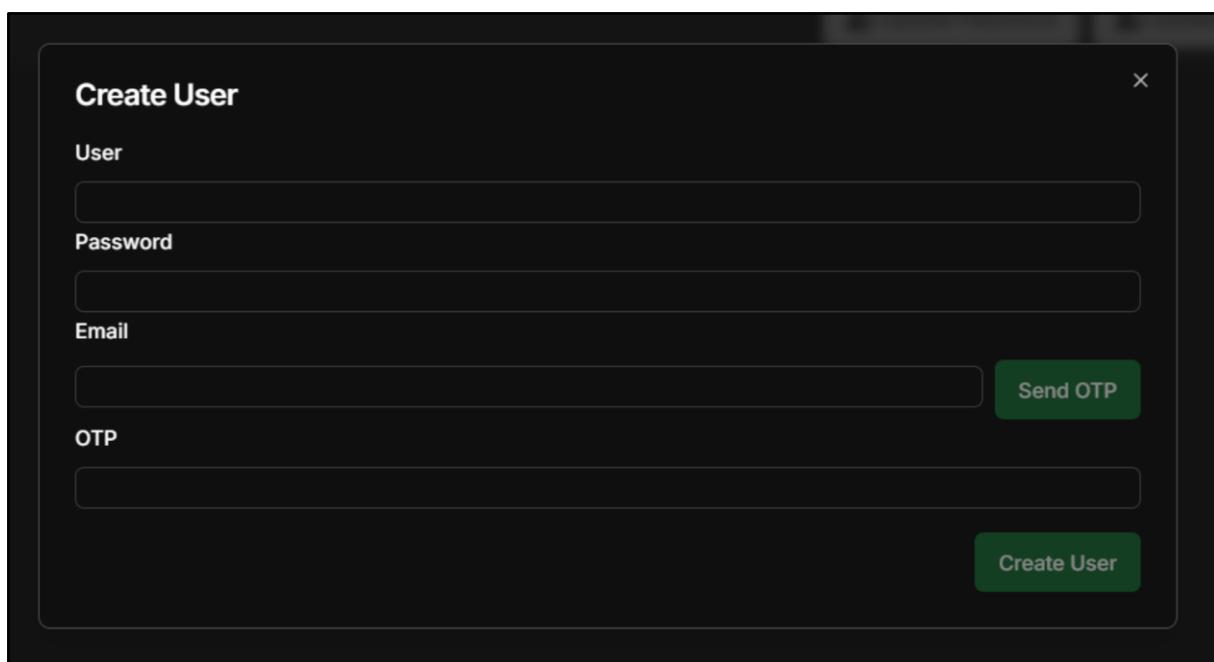
3.3.2.3.2 Create user

Assume that you are viewing list of user. To create user, click on button “+ Add user” on the top-right corner.

The screenshot shows the same dark-themed user management interface as Figure 278. The '+ Add User' button in the top right corner is highlighted with a red box.

Figure 279. Manage users - Create user

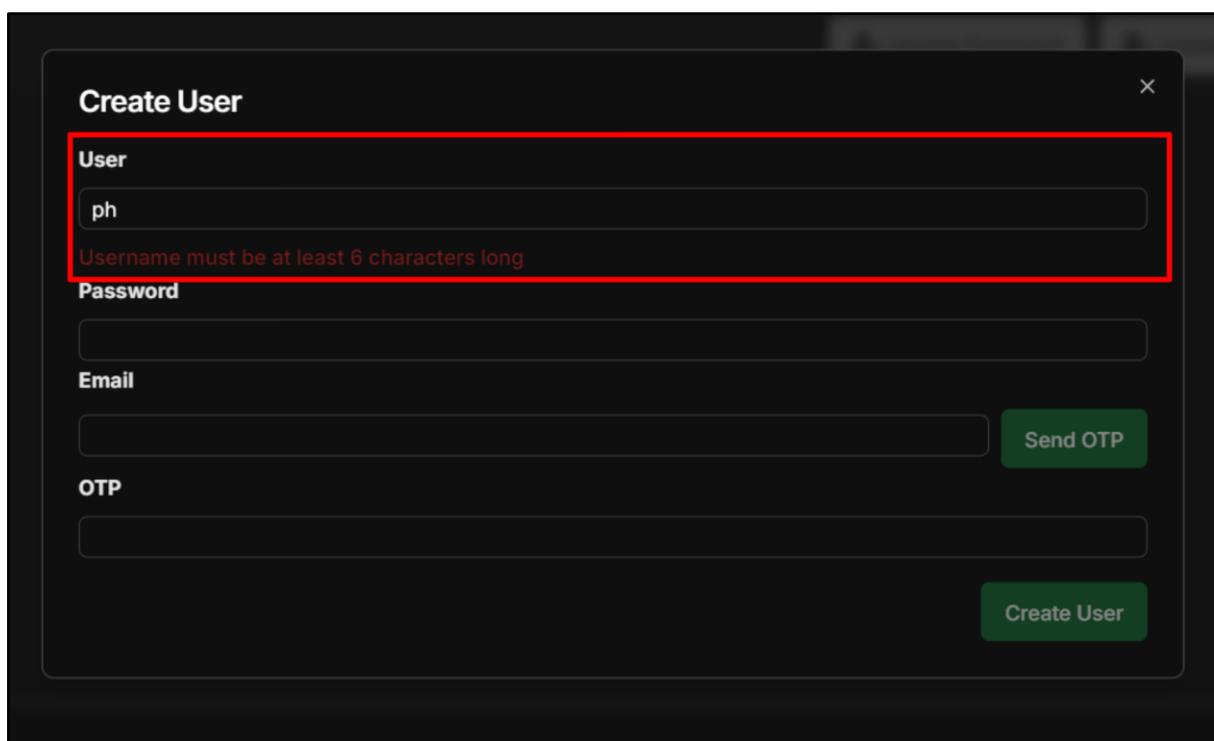
Enter user information:



The image shows a dark-themed 'Create User' form. It includes fields for User (text input), Password (text input), Email (text input), and OTP (text input). A green 'Send OTP' button is located next to the Email field. A green 'Create User' button is at the bottom right. An 'X' icon is in the top right corner.

Figure 280. Manage users - Create user

Please note that Username must be at least 6 characters long.



The image shows the same 'Create User' form as Figure 280, but with validation applied to the 'User' field. The 'User' field contains 'ph' and has a red border around it. A red box also surrounds the error message 'Username must be at least 6 characters long'. The other fields (Password, Email, OTP) and buttons (Send OTP, Create User) are visible but unaffected by the validation.

Figure 281. Manage users - Create user

...and username may only include letters, numbers, . or _

The screenshot shows a 'Create User' form on a dark background. The 'User' field is highlighted with a red box. Inside the 'User' field, the text 'new user' is entered. Below the 'User' field, a placeholder text 'Username may only include letters, numbers, . or _' is displayed. The 'Password', 'Email', and 'OTP' fields are empty. To the right of the 'Email' field is a green 'Send OTP' button. At the bottom right is a green 'Create User' button.

Figure 282. Manage users - Create user

Below is a valid username: **newuser**

The screenshot shows the same 'Create User' form as Figure 282, but with a valid username. The 'User' field is highlighted with a red box and contains the text 'newuser'. The other fields ('Password', 'Email', 'OTP') are empty. To the right of the 'Email' field is a green 'Send OTP' button. At the bottom right is a green 'Create User' button.

Figure 283. Manage users - Create user

About password, please note that:

- Password must be at least 8 characters long
- Password must include at least one uppercase letter.

- Password must contain at least one special character.
- Password must contain at least one number.

Example of invalid passwords: Cantho65, cantho@_, cantho65, C^nth0

Example of valid password: Cantho65^^

The screenshot shows a 'Create User' dialog box. The 'User' field contains 'newuser'. The 'Password' field contains '*****'. A red box highlights both the password input field and the error message 'Password must be at least 8 characters long' located below it. The 'Email' and 'OTP' fields are empty. On the right side, there is a 'Send OTP' button and a green 'Create User' button.

Figure 284. Manage users - Create user

This screenshot shows the same 'Create User' dialog box. The 'User' field contains 'newuser'. The 'Password' field contains '*****'. A red box highlights both the password input field and the error message 'Password must include at least one uppercase letter' located below it. The 'Email' and 'OTP' fields are empty. On the right side, there is a 'Send OTP' button and a green 'Create User' button.

Figure 285. Manage users - Create user

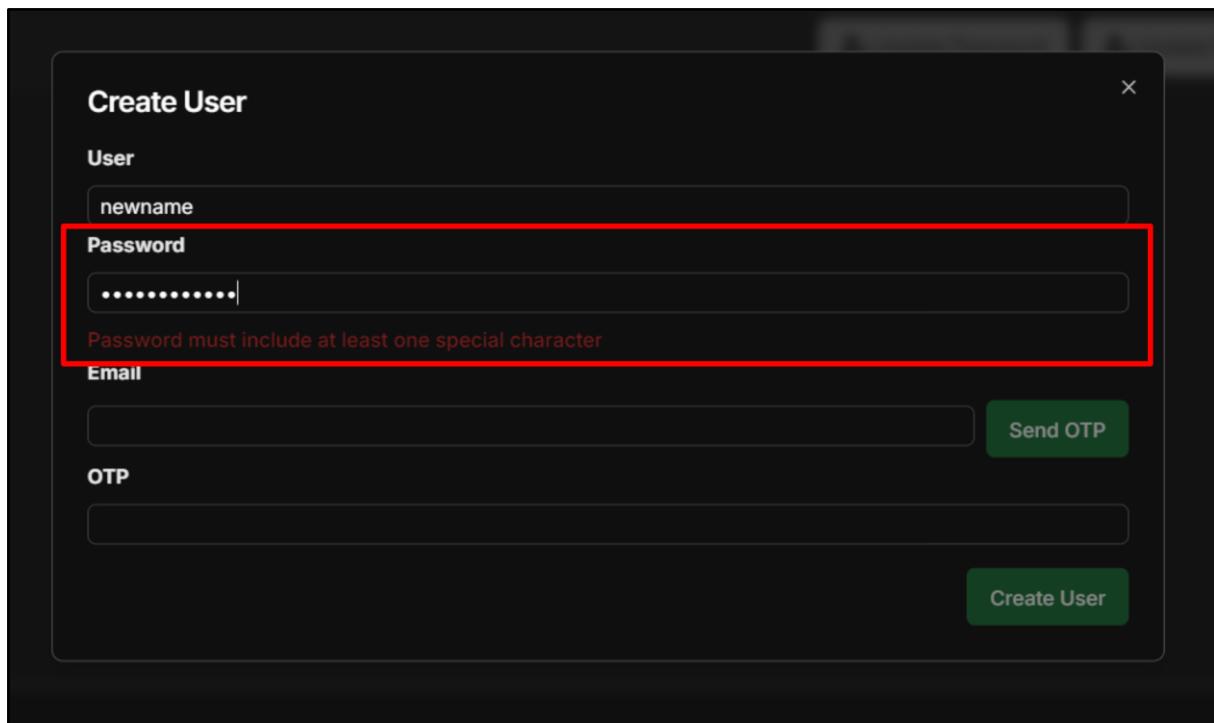


Figure 286. Manage users - Create user

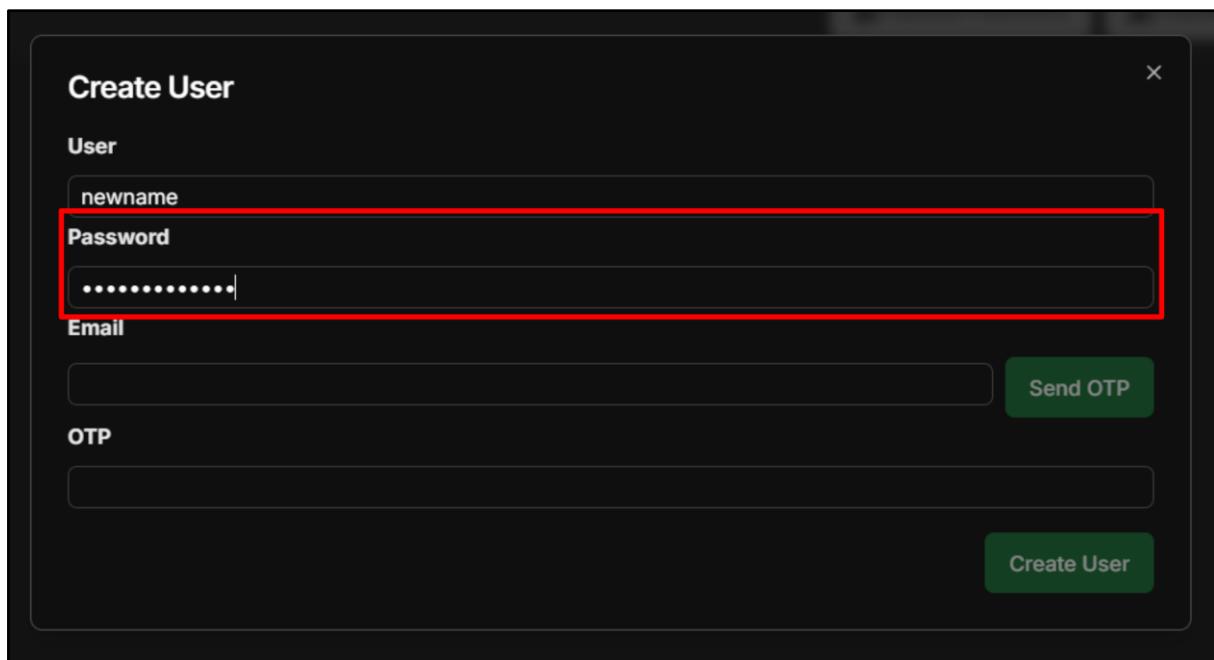


Figure 287. Manage users - Create user

Email should be valid:

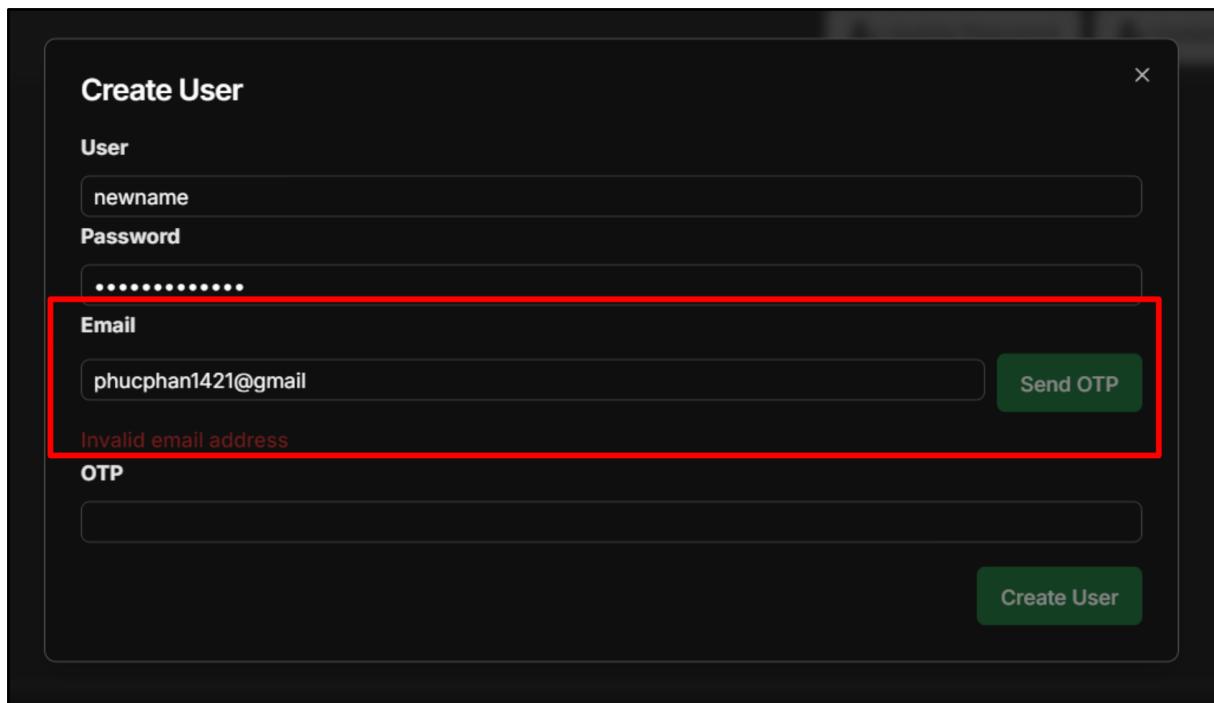


Figure 288. Manage users - Create user

If email is valid, click on button "Send OTP"

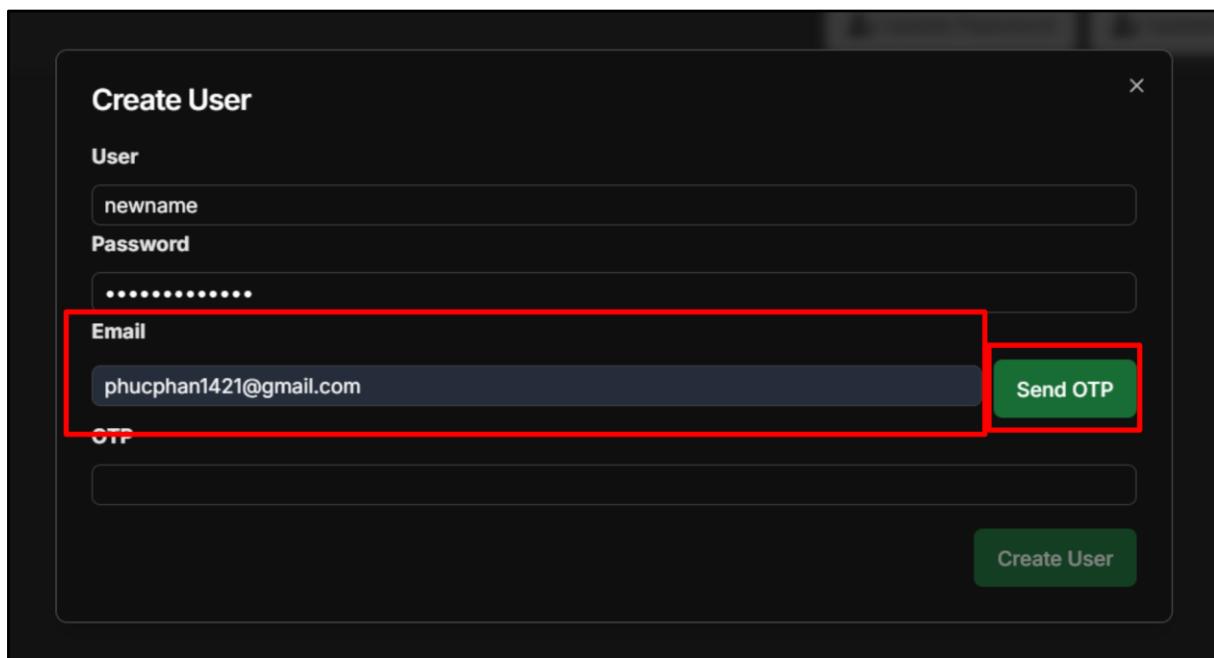


Figure 289. Manage users - Create user

At the time that the message “OTP sent successfully” pops up, check your email address and copy the OTP

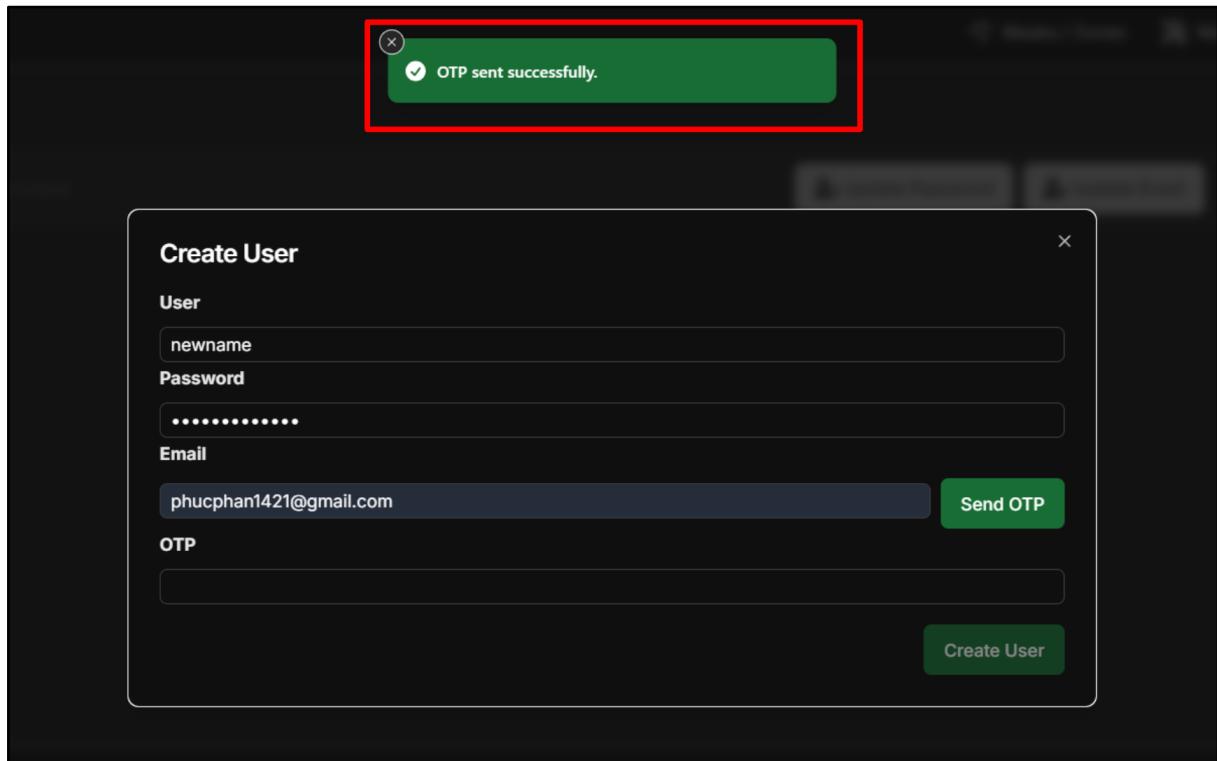


Figure 290. Manage users - Create user

Example of email receiving OTP

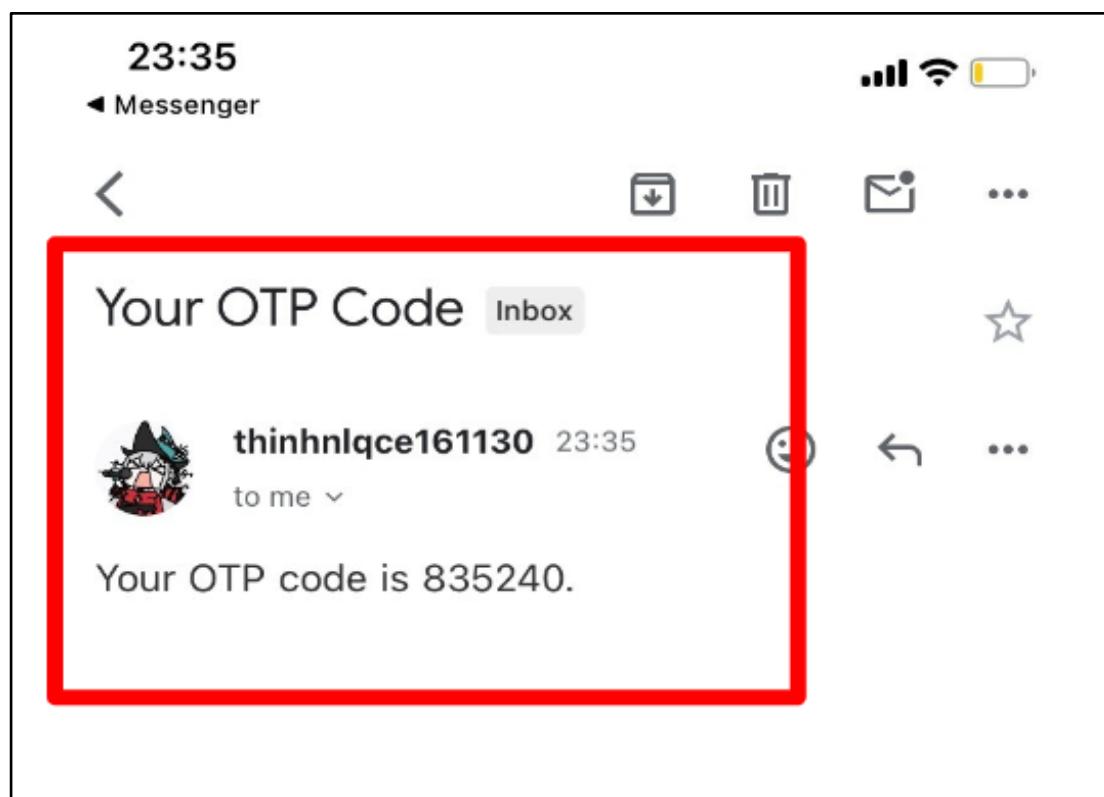


Figure 291. Manage users - Create user

OTP should be 6 characters long

The screenshot shows a 'Create User' form with a dark background. It includes fields for User (newname), Password, Email (phucphan1421@gmail.com), and OTP (3123). A red box highlights the OTP field and its validation message: 'OTP must be 6 characters long'. A green 'Send OTP' button is located to the right of the email field.

Figure 292. Manage users - Create user

After entering a 6 characters long OTP, click on the button “Create User”

The screenshot shows the same 'Create User' form as Figure 292, but with a valid OTP (312332) entered in the OTP field. A red box highlights the OTP field and the 'Create User' button. The 'Create User' button is now green and appears to be functional.

Figure 293. Manage users - Create user

If OTP is incorrect, error message is displayed.

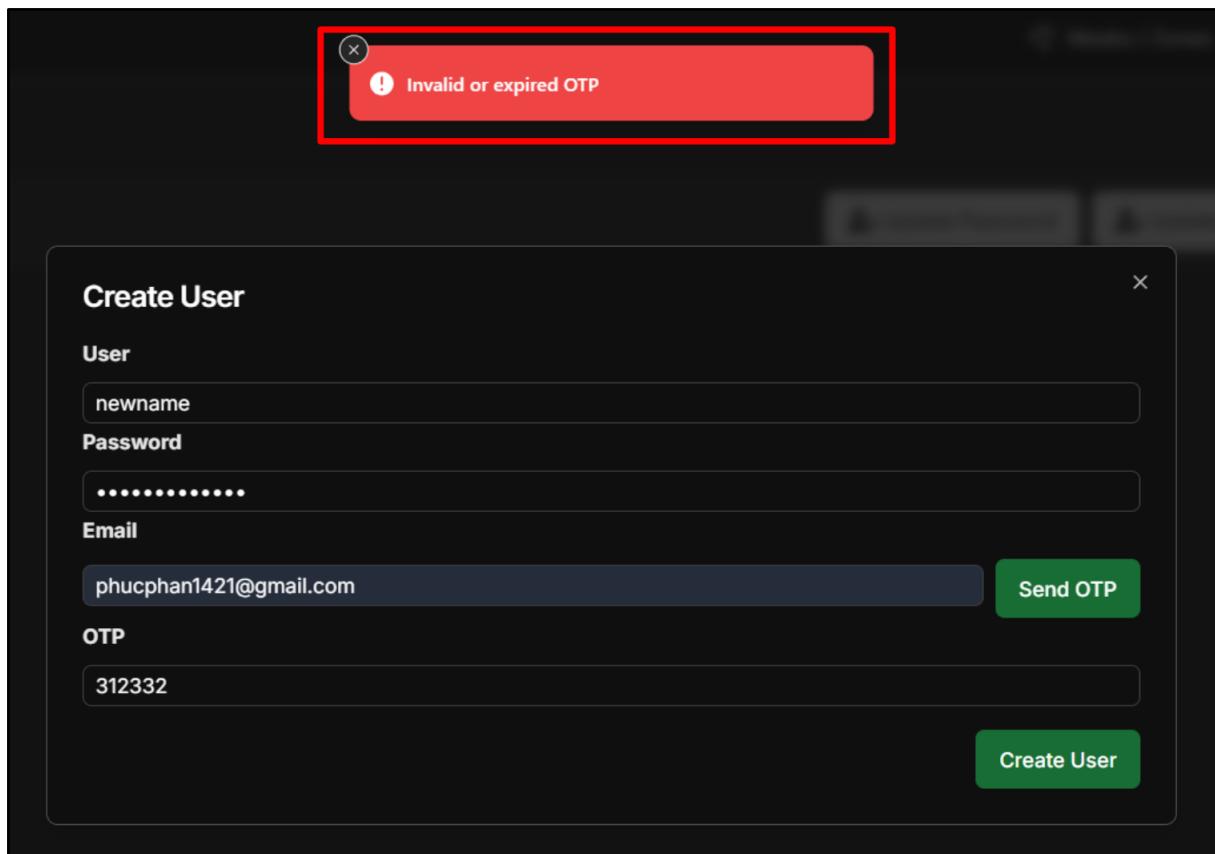


Figure 294. Manage users - Create user

If the OTP is correct, the system displays “Created successfully”

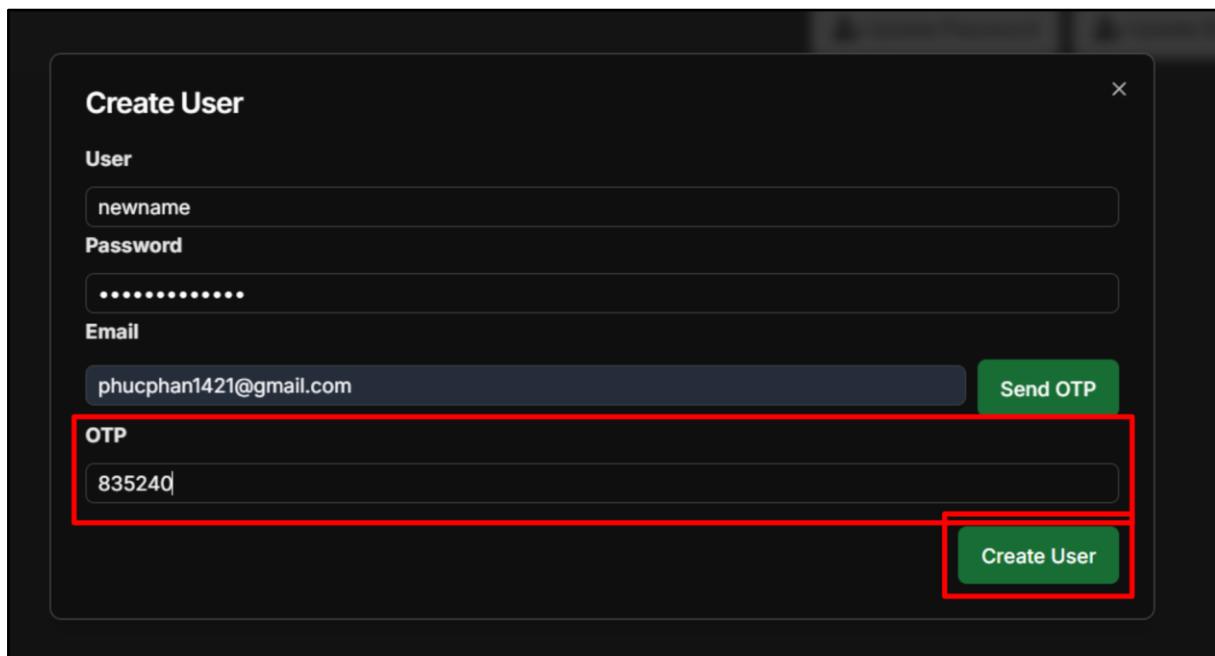


Figure 295. Manage users - Create user

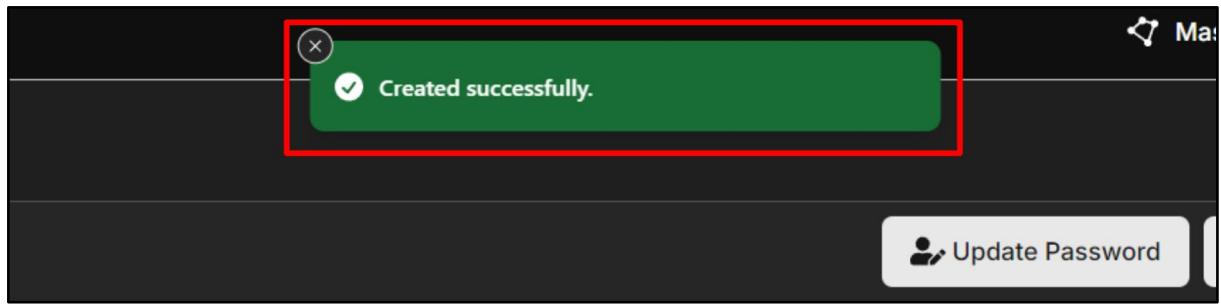


Figure 296. Manage users - Create user

After adding successfully, you will see that account is added to user list.

Users		+ Add User
admin	No email provided	Update Password Update Email Receive alert: Off Delete
newname	phucphan1421@gmail.com	Update Password Update Email Receive alert: Off Delete

Figure 297. Manage users - Create user

Another invalid case: The entered email is existed in the database.

A screenshot of a modal dialog titled "Create User". The form contains fields for "User" (with value "newuser2"), "Password" (with placeholder dots), "Email" (with value "phucphan1421@gmail.com" highlighted with a red box), and "OTP" (with value "153021"). To the right of the email field is a "Send OTP" button. At the bottom right of the modal is a large green "Create User" button.

Figure 298. Manage users - Create user

At that time, an error message was displayed.

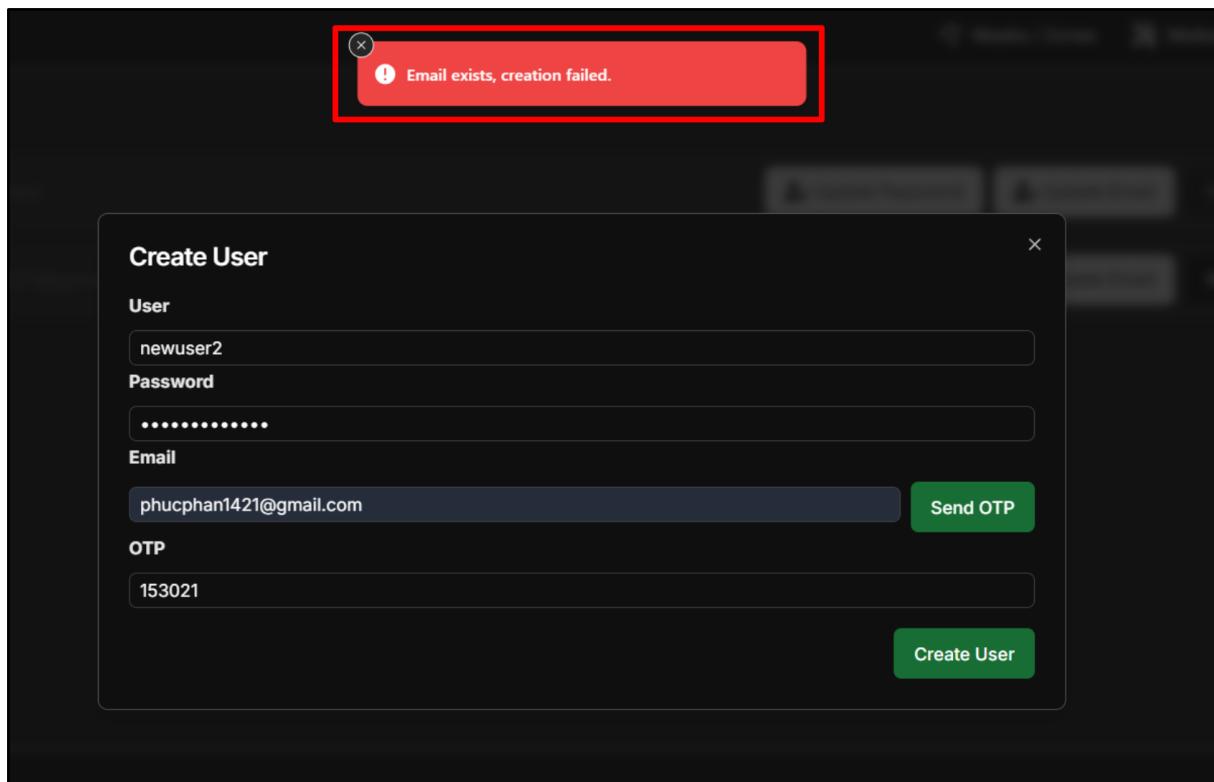


Figure 299. Manage users - Create user

Another invalid case: Username has existed in the database

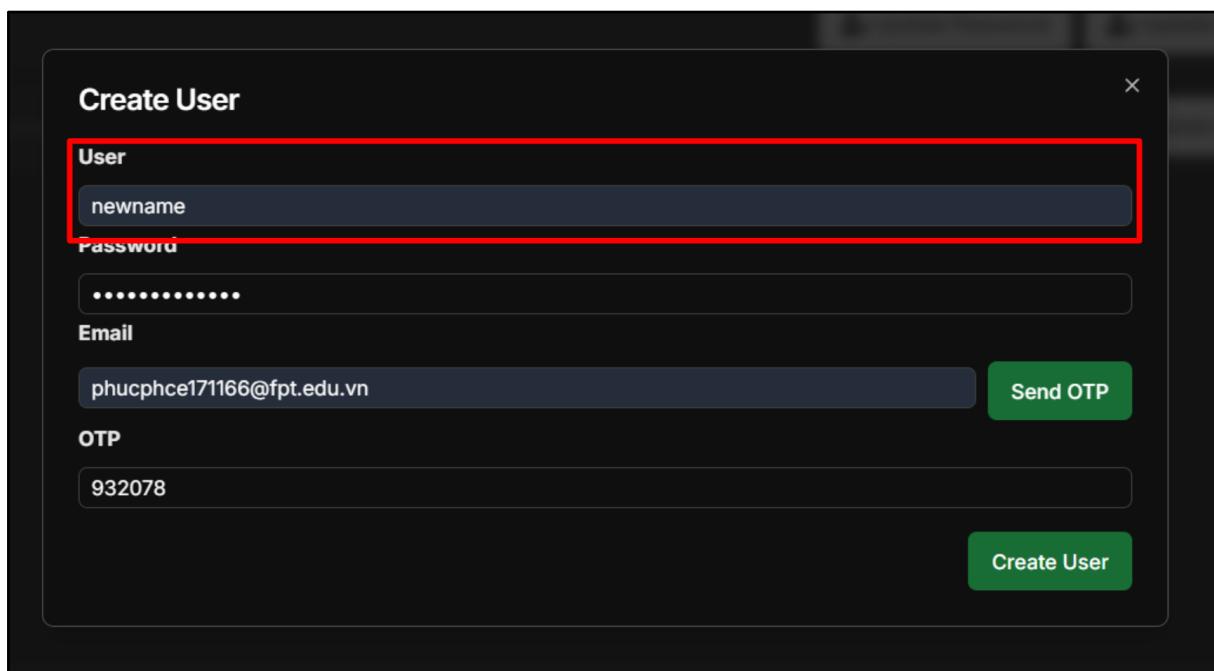


Figure 300. Manage users - Create user

Then, an error message was displayed

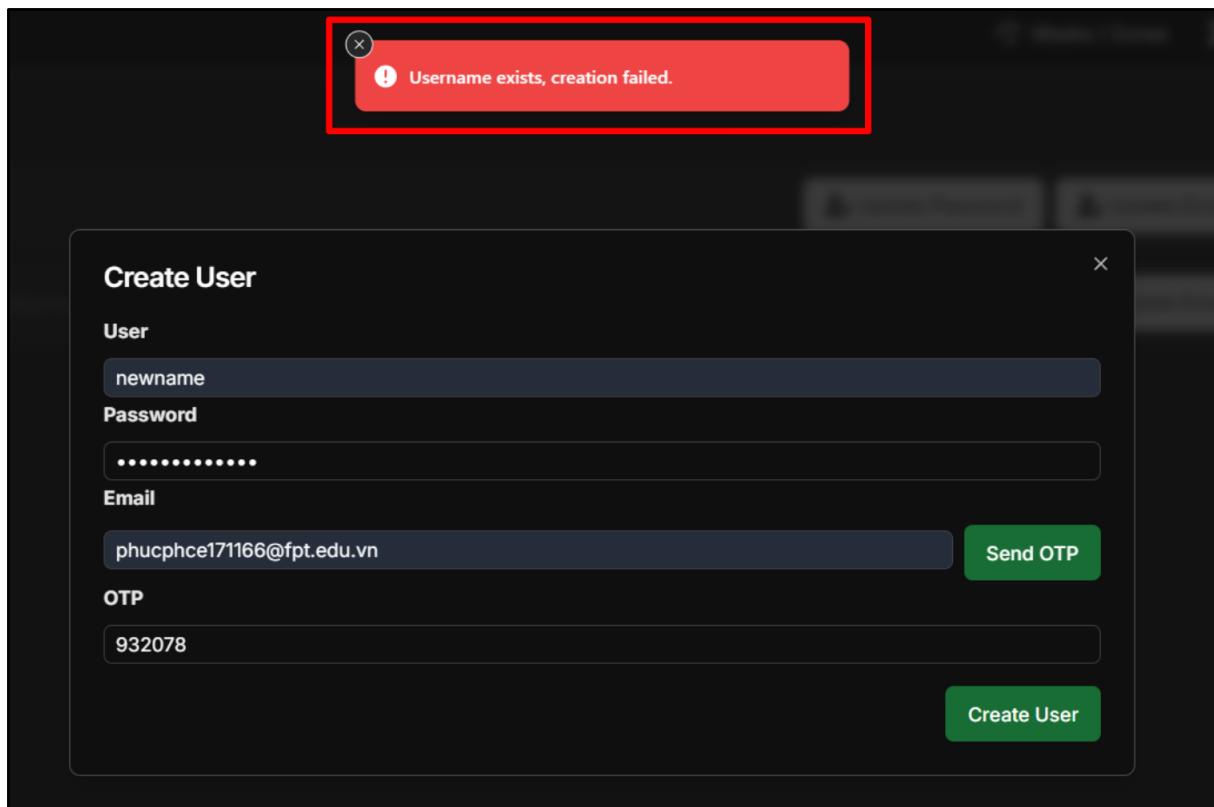


Figure 301. Manage users - Create user

Valid case: All inputs are valid

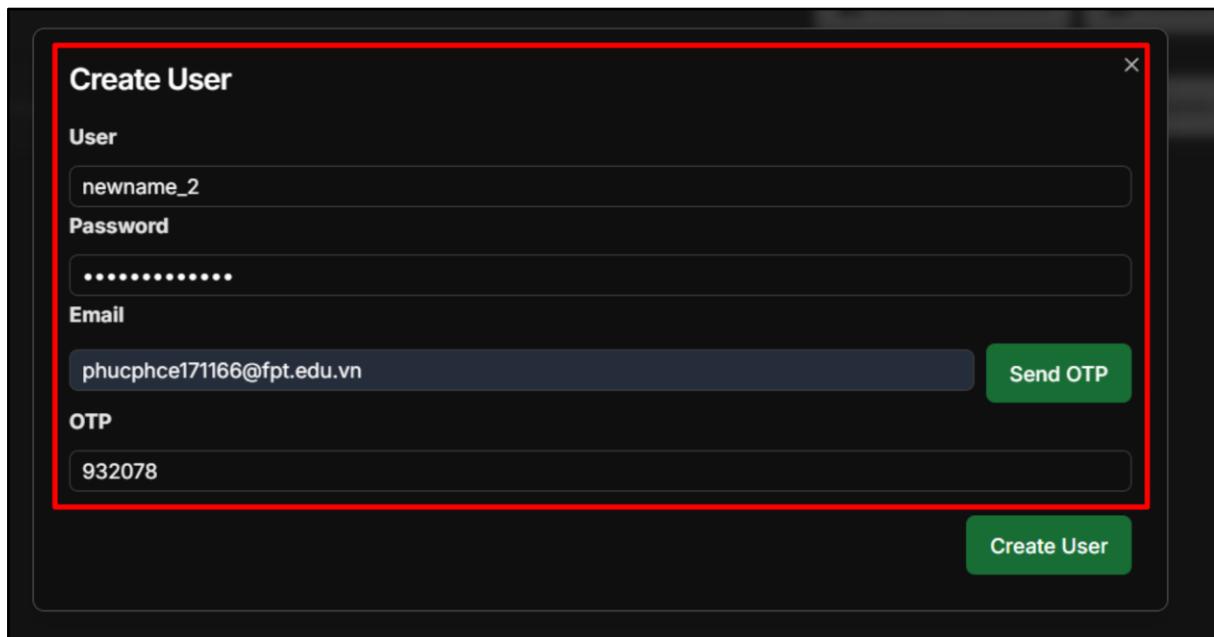


Figure 302. Manage users - Create user

New user is added

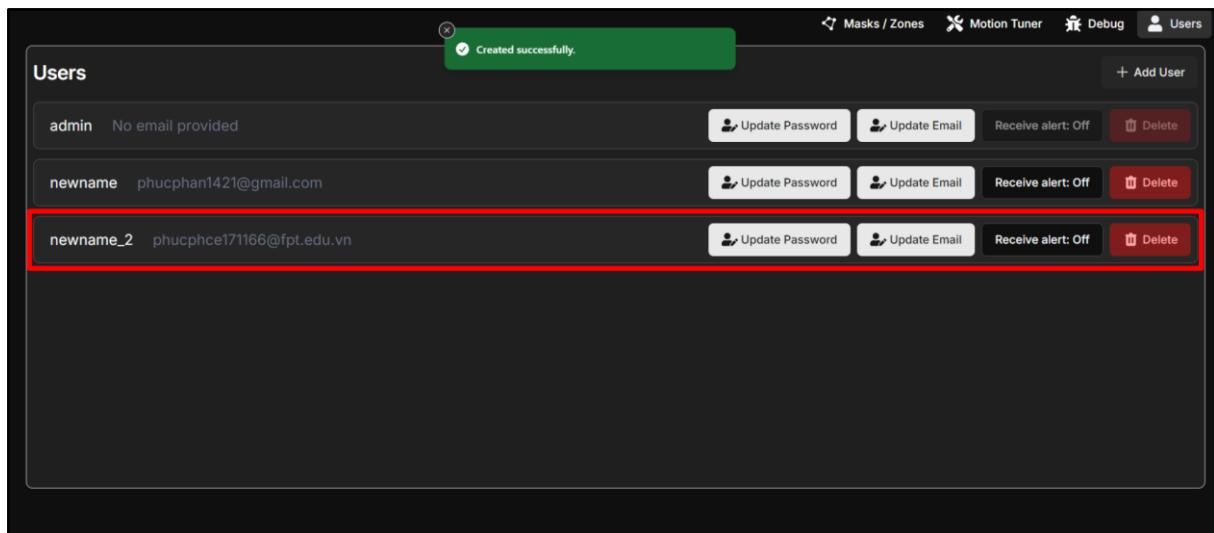


Figure 303. Manage users - Create user

3.3.2.3.3 Update email

If you want to update user email, you can click on the button “Update Email” for that user in the user list.

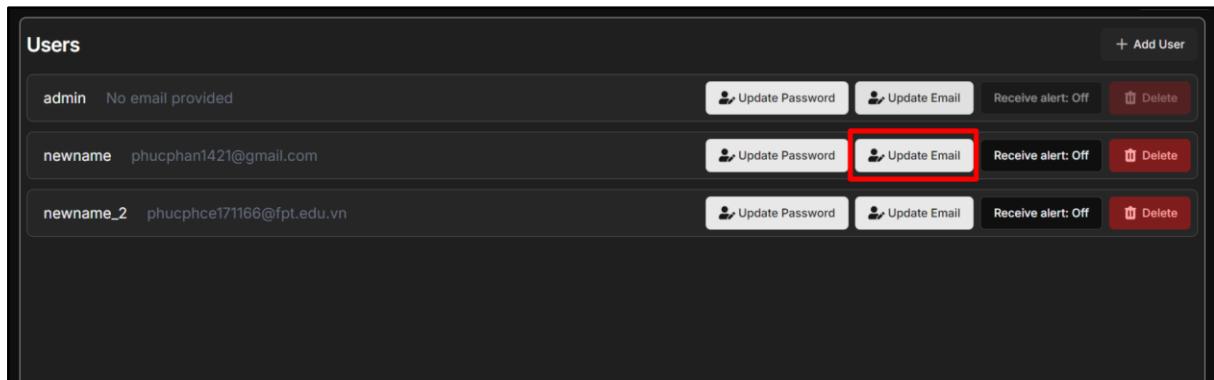


Figure 304. Manage users - Update email

Next, enter new email address, click on “Send OTP”, enter, OTP and click on the button “Update Email”

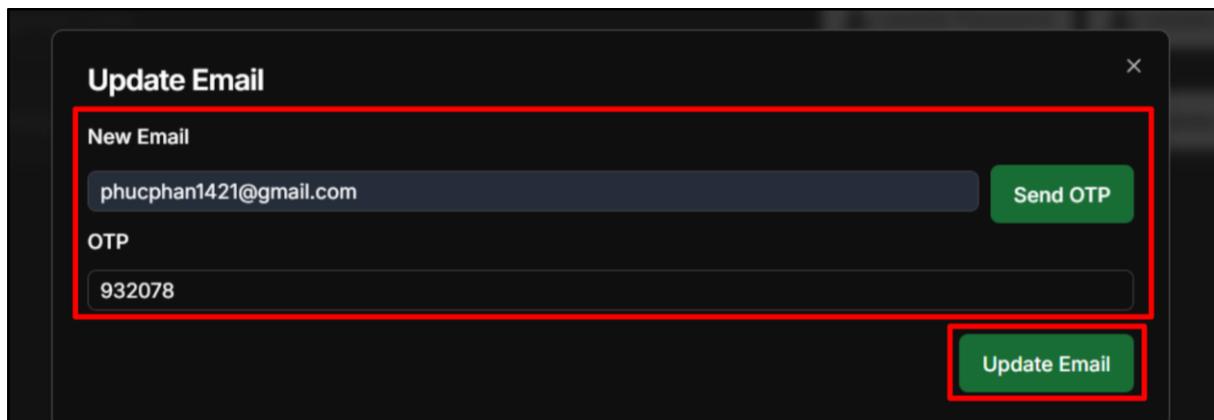


Figure 305. Manage users - Update email

If the entered email address already exists, enter another email address and try again.

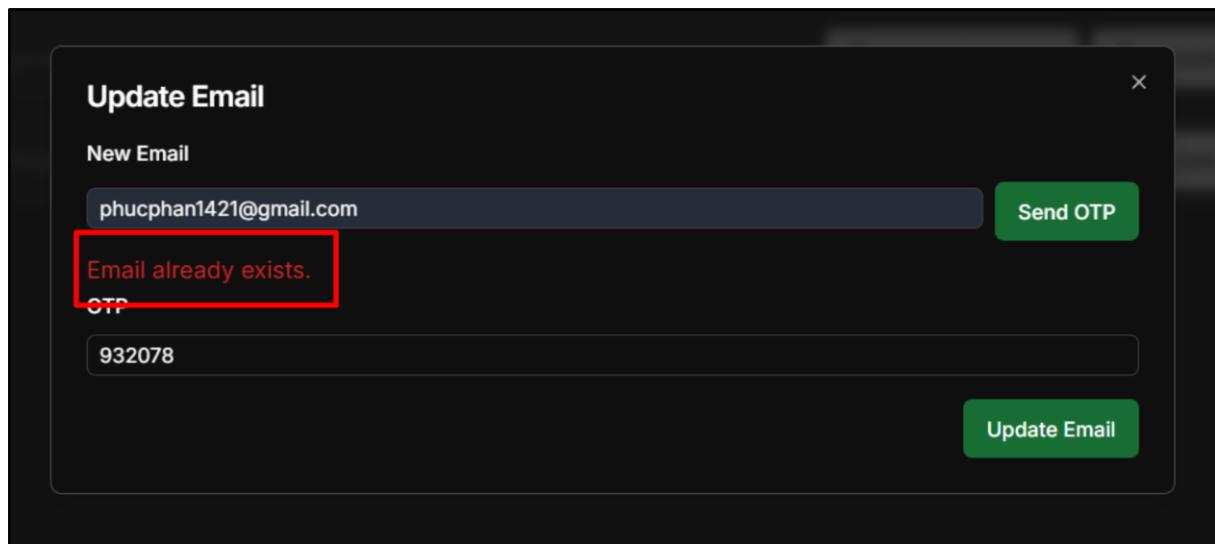


Figure 306. Manage users - Update email

Try entering another email address and send OTP

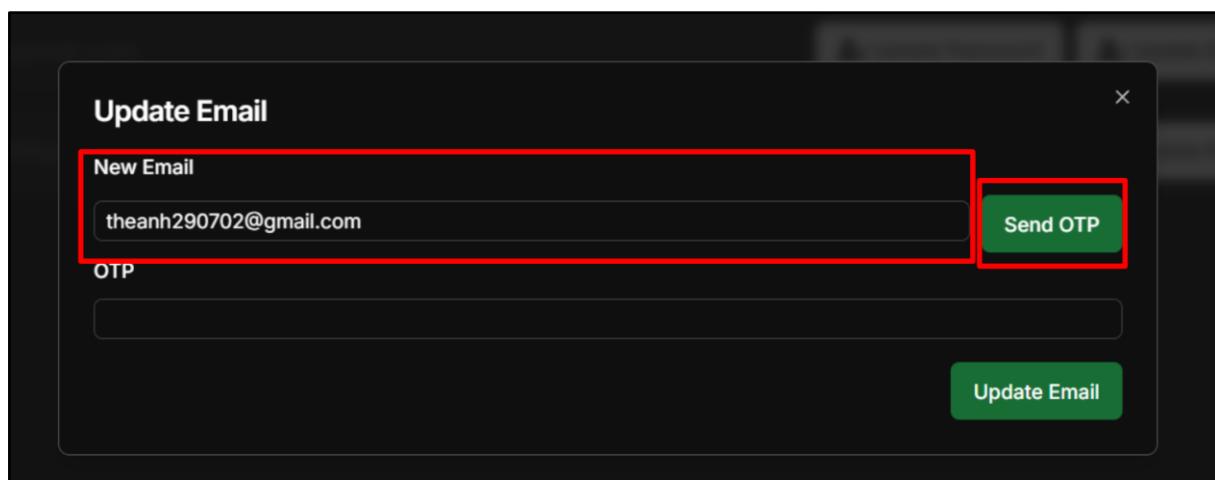


Figure 307. Manage users - Update email

Sent successfully

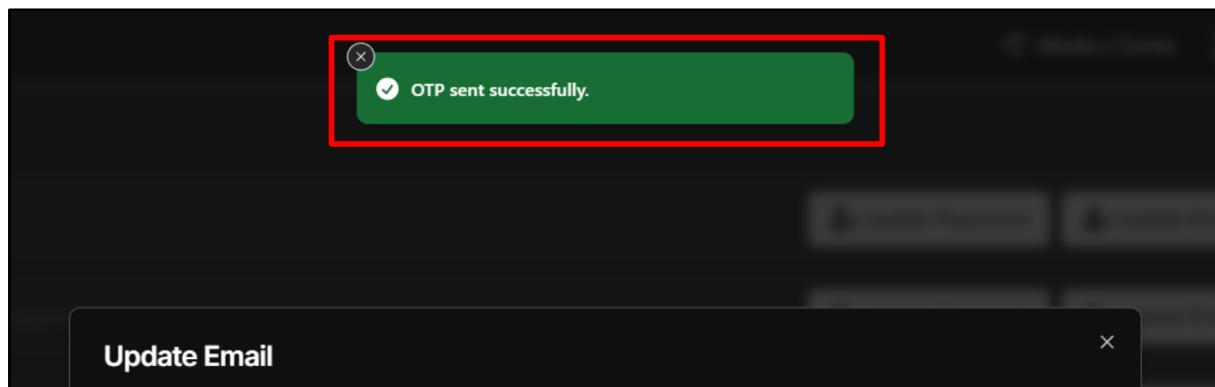


Figure 308. Manage users - Update email

Check your email and copy the OTP

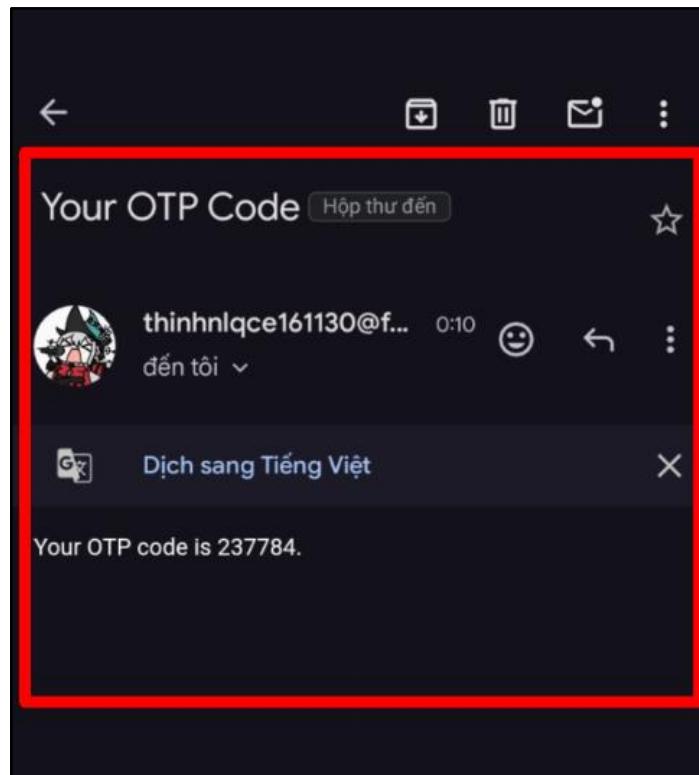


Figure 309. Manage users - Update email

Oh no, it is an incorrect OTP

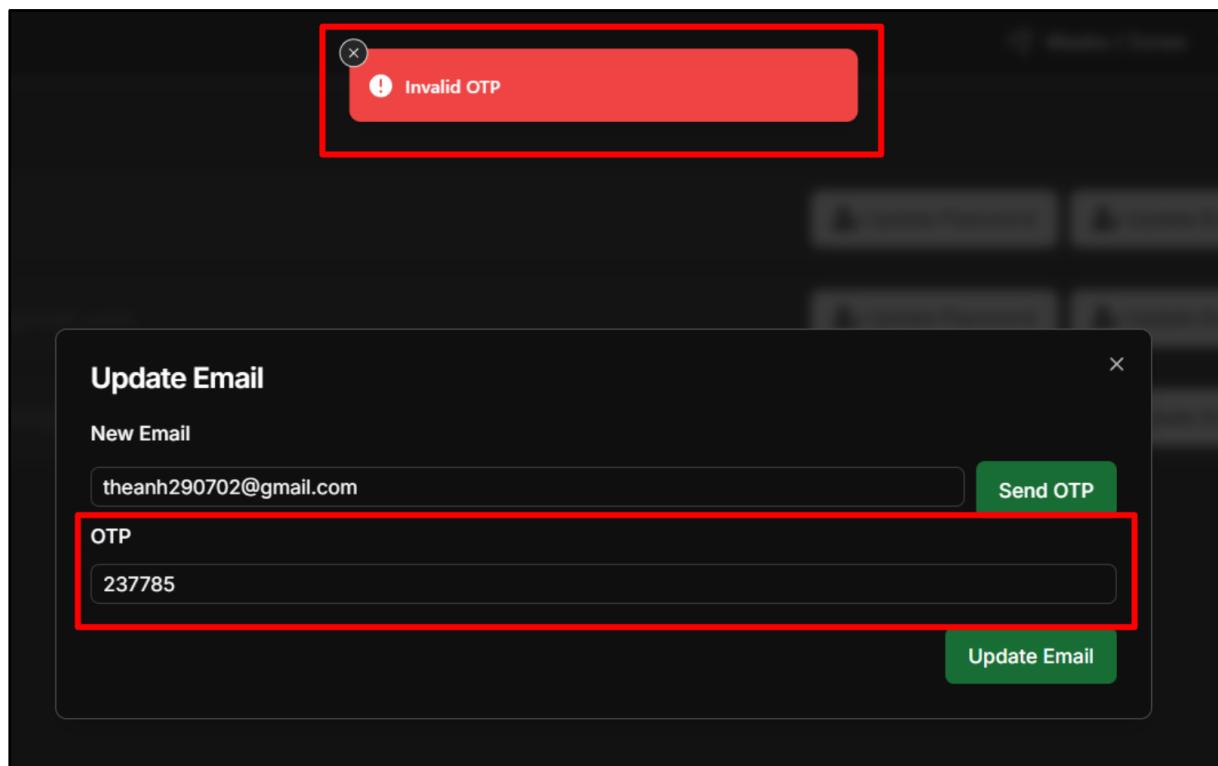


Figure 310. Manage users - Update email

Enter correct OTP and click on the button “Update Email”

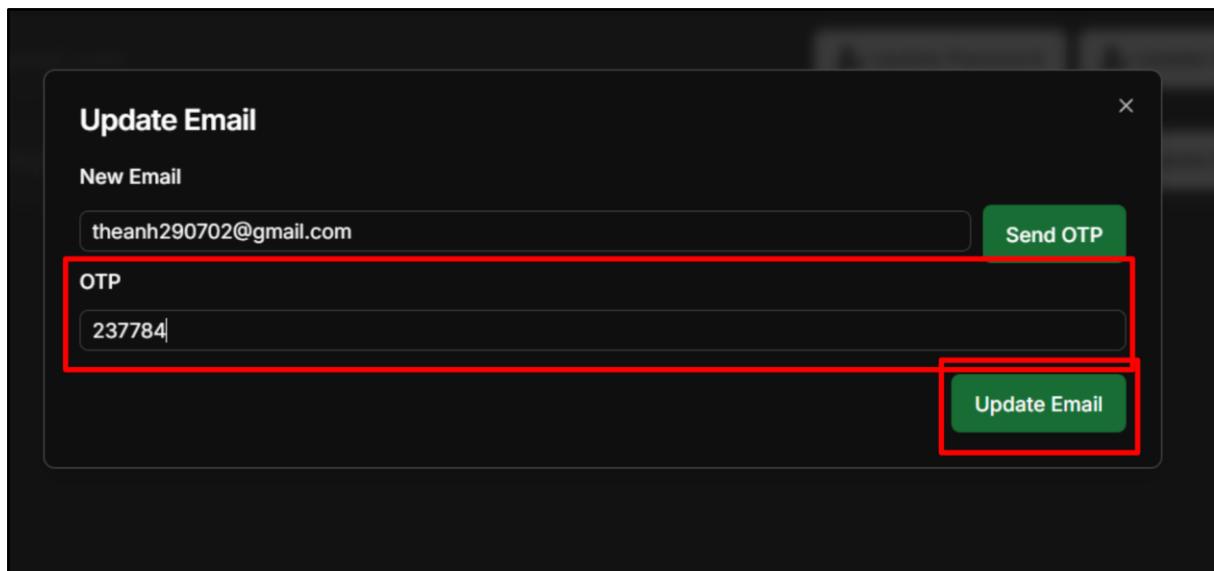


Figure 311. Manage users - Update email

Email updated successfully

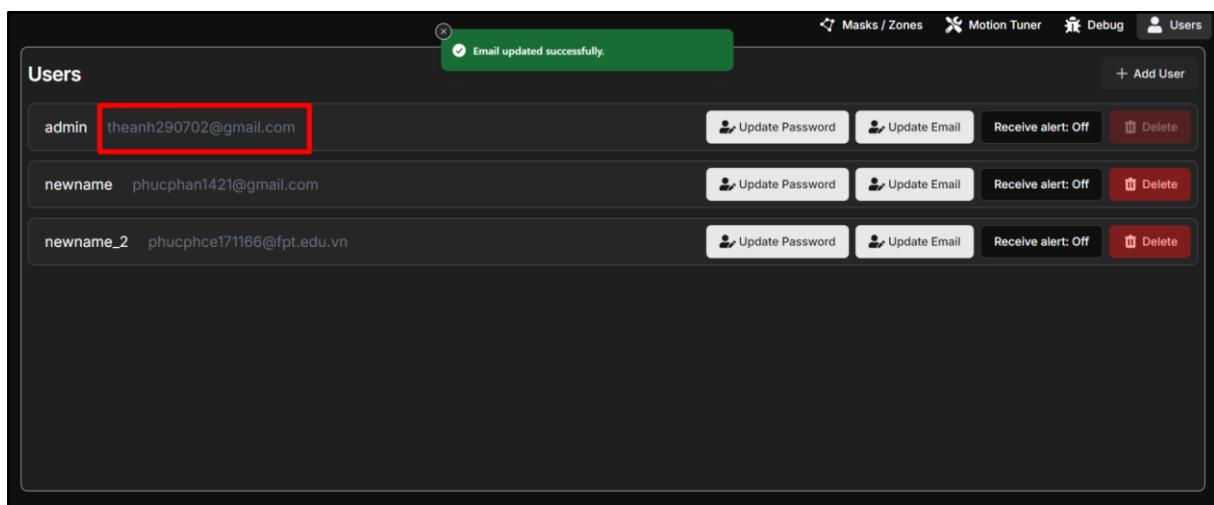


Figure 312. Manage users - Update email

3.3.2.3.4 Update password

For security, you should change the default password immediately after your first login (as admin). To update user password, click on the button “Update Password” of the account you want to update

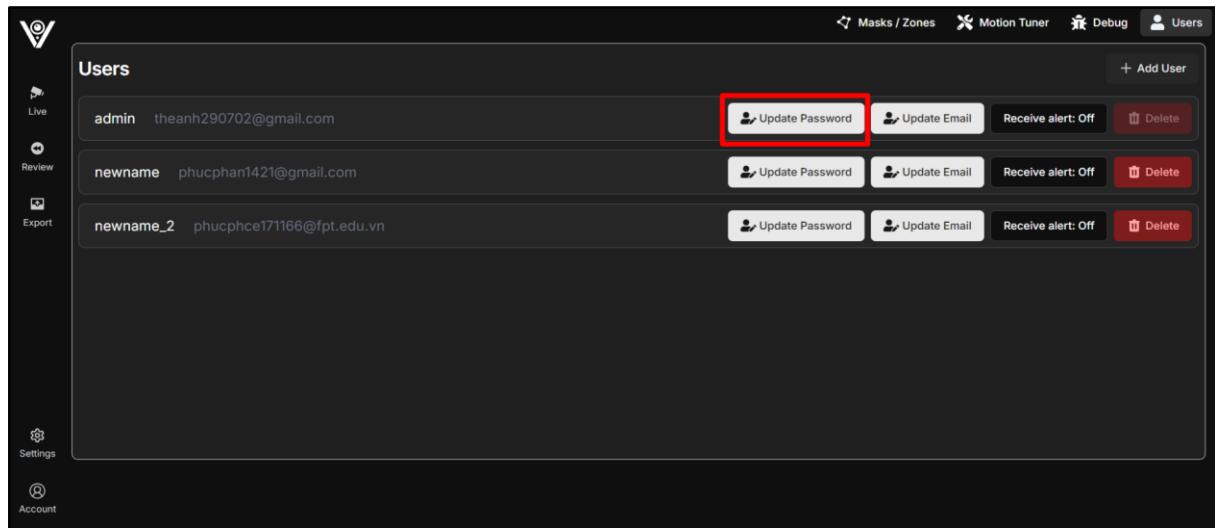


Figure 313. Manage users - Update password

“Set password” form appears.

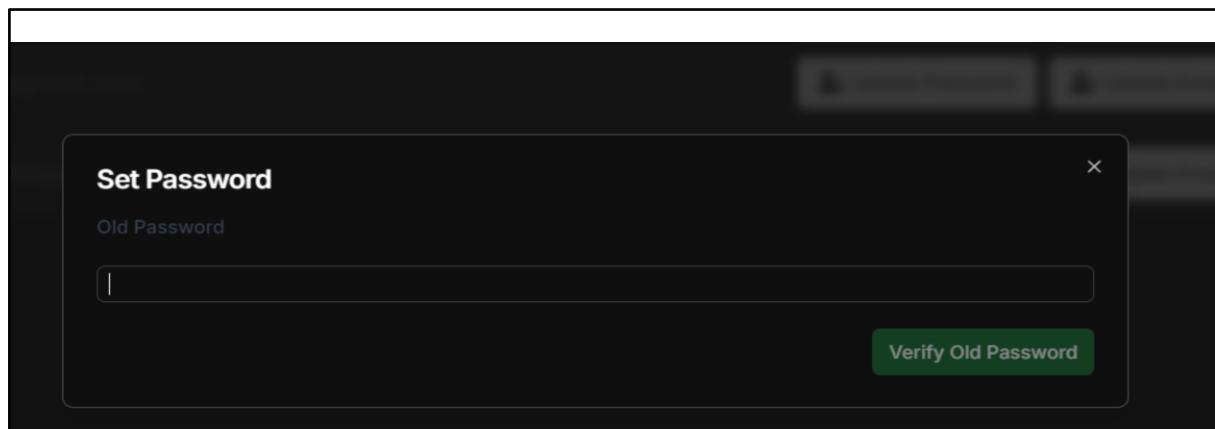


Figure 314. Manage users - Update password

Enter **old password** and click on the button “Verify Old Password”

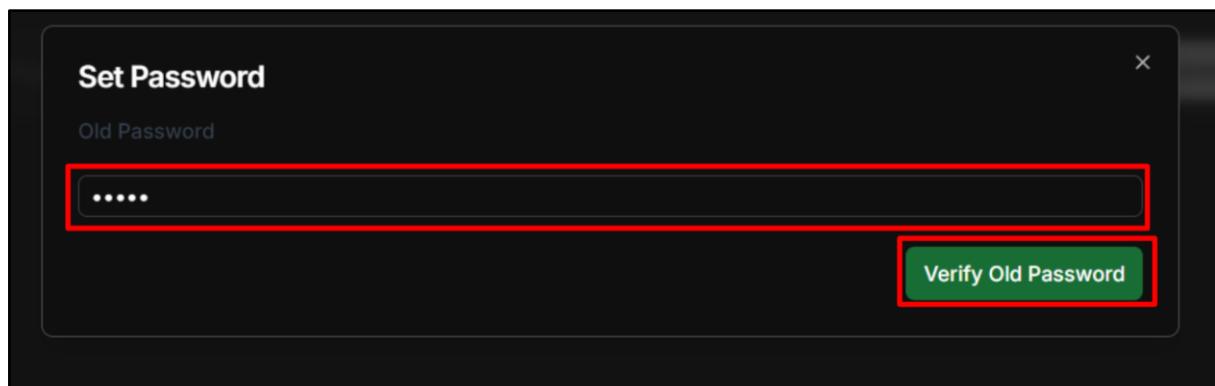


Figure 315. Manage users - Update password

In this case, old password is incorrect

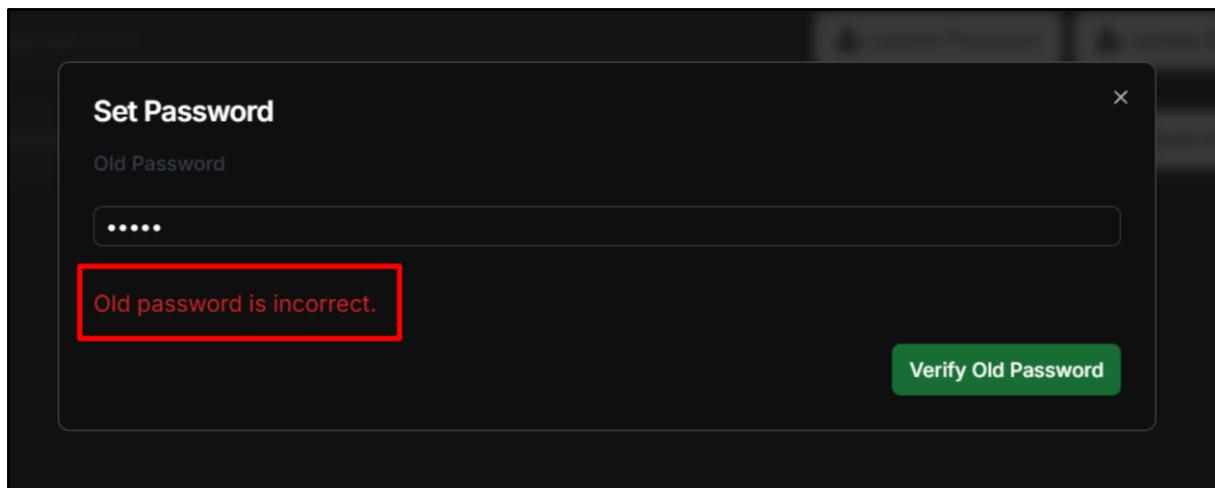


Figure 316. Manage users - Update password

If password is correct, enter **new password**. Please note that password must satisfy the following conditions:

- Password must be at least 8 characters long.
- Password must contain at least one number.
- Password must contain at least one uppercase letter.
- Password must contain at least one special character.

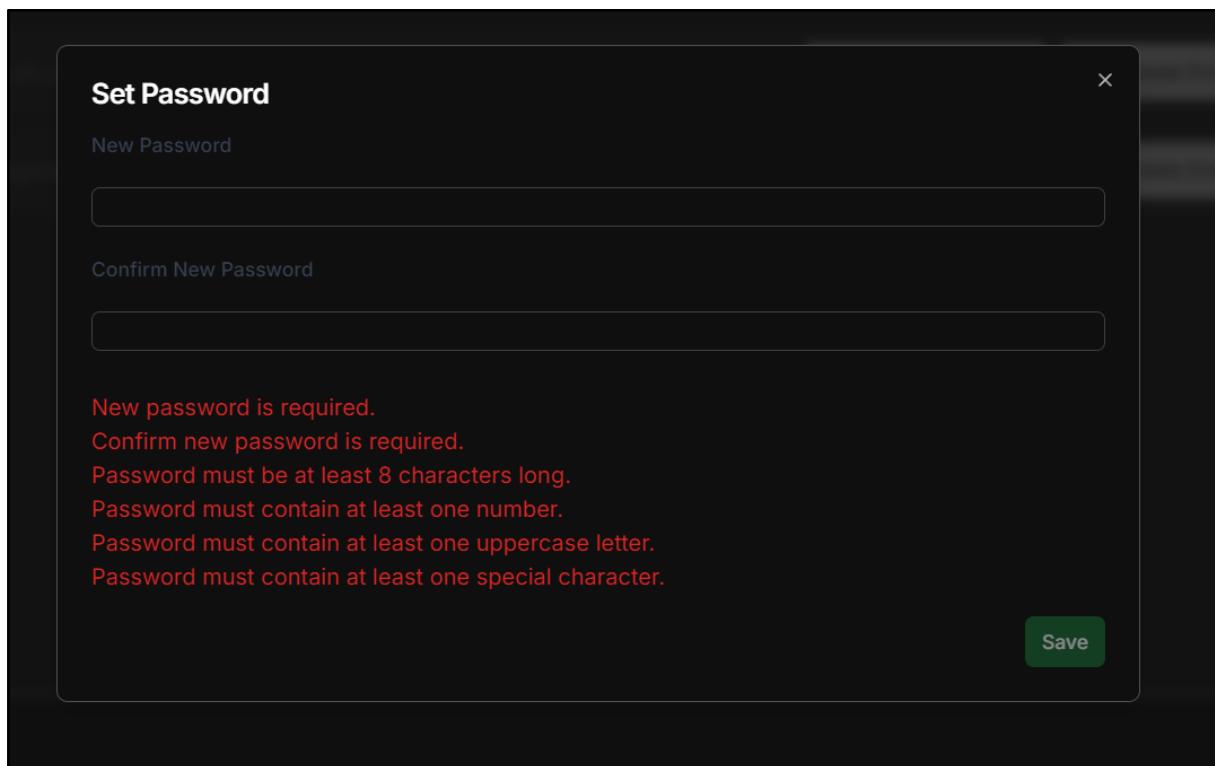


Figure 317. Manage users - Update password

Example of an error case:

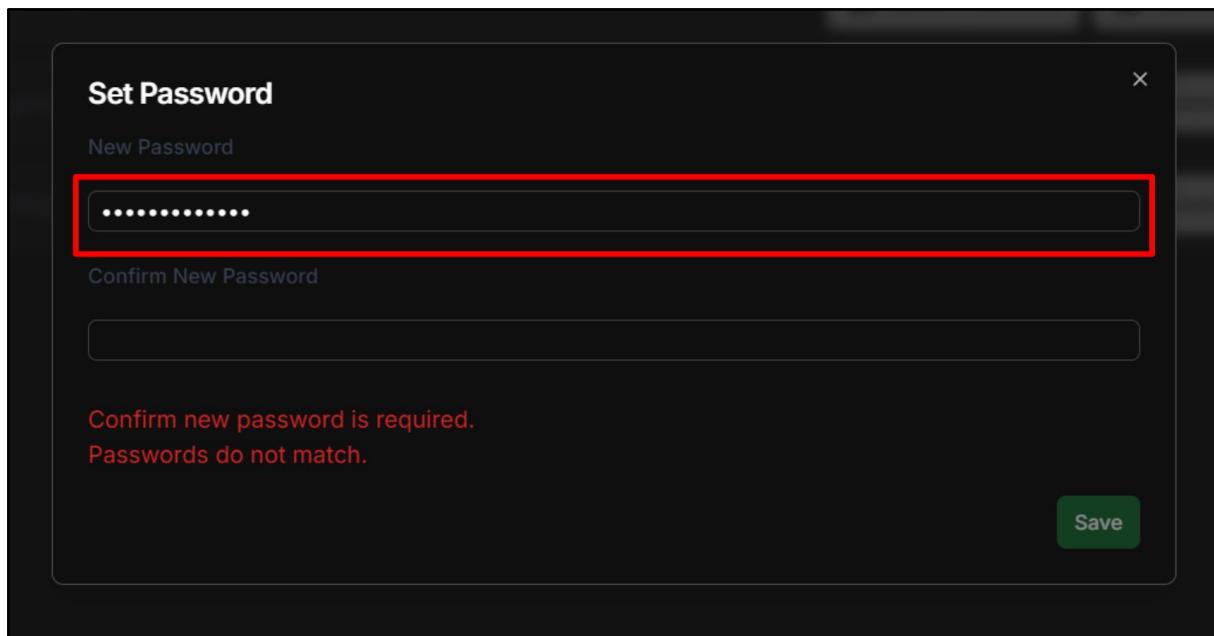


Figure 318. Manage users - Update password

Example of another error case:

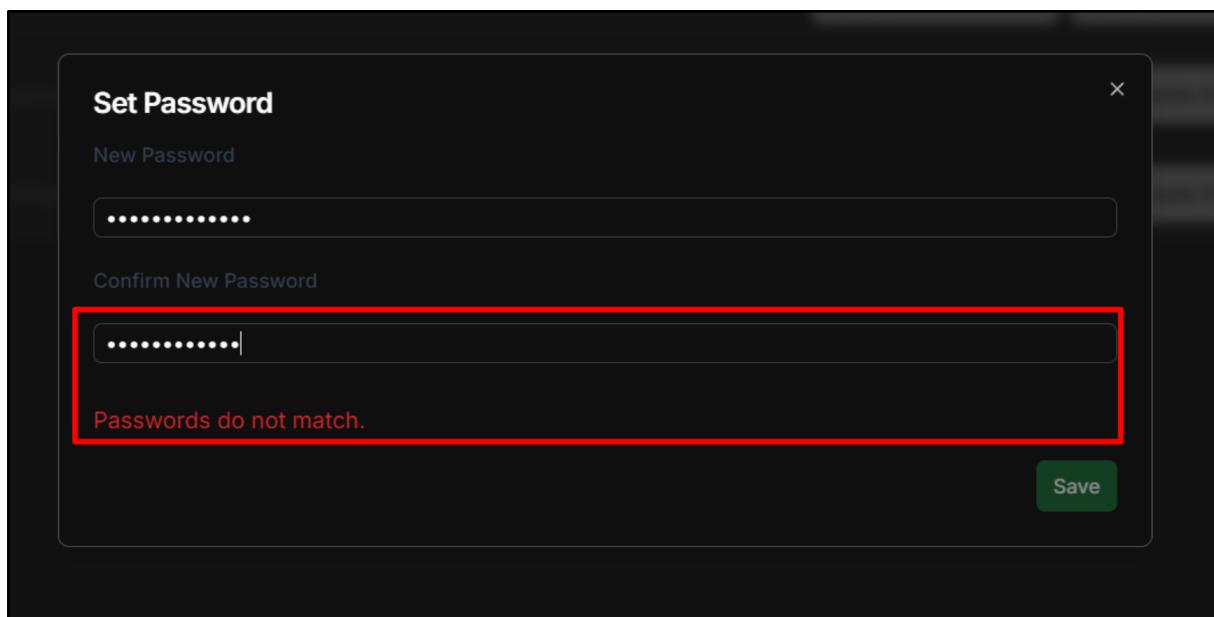


Figure 319. Manage users - Update password

Normal/valid case: Update user password successfully

Enter valid password and confirm correctly, click **Save**. Choose a strong password that includes a mix of letters, numbers, and symbols to enhance your system's security.

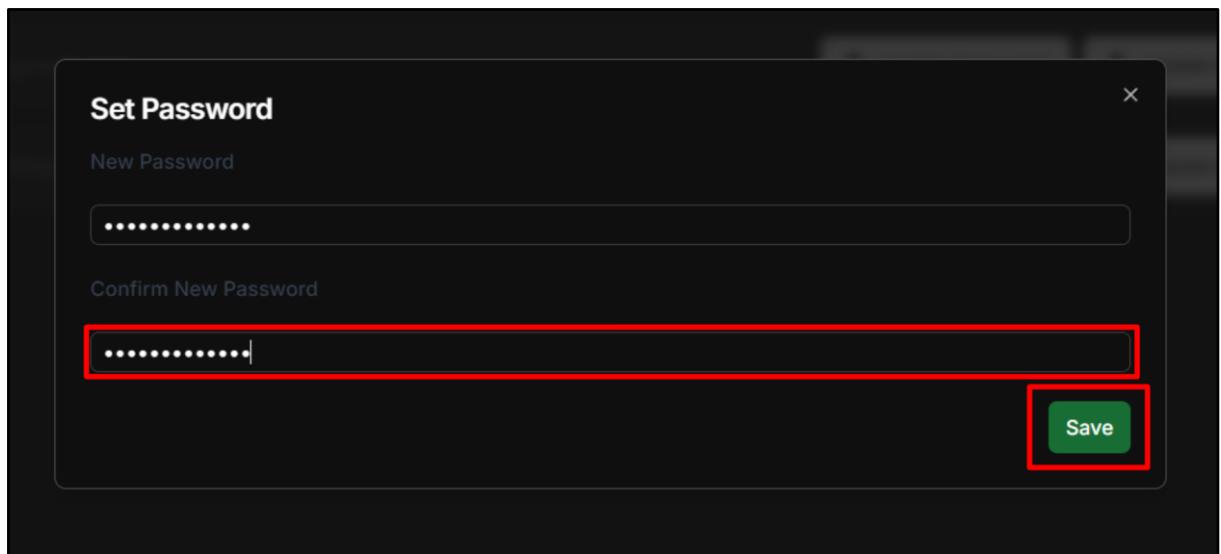


Figure 320. Manage users - Update password

The system inform “Password updated successfully”

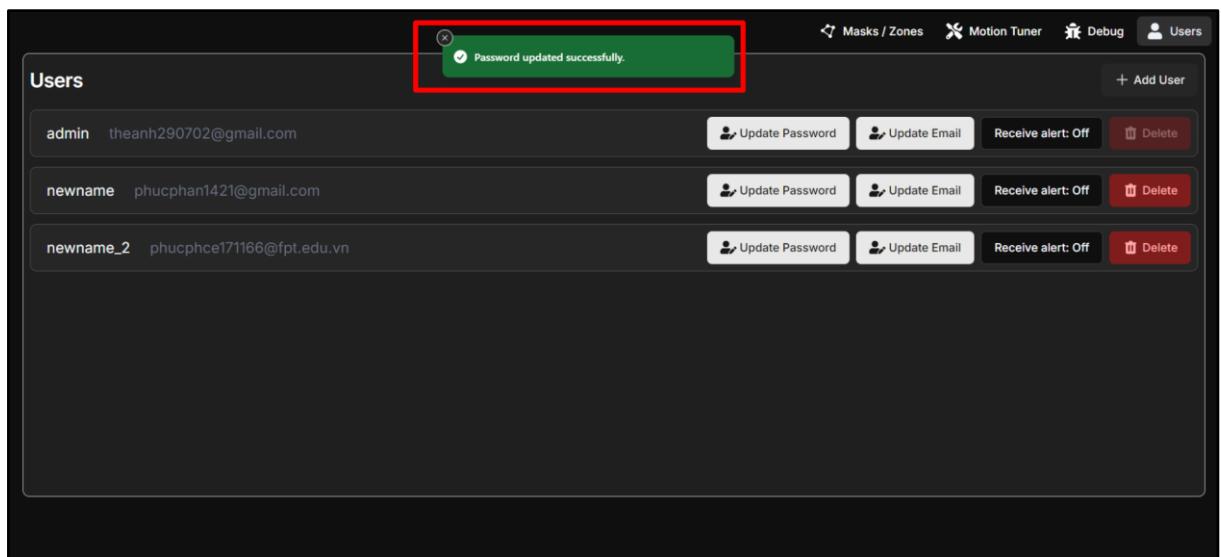


Figure 321. Manage users - Update password

3.3.2.3.5 Delete user

To delete a user, click on the button “Delete” corresponding to that user account.

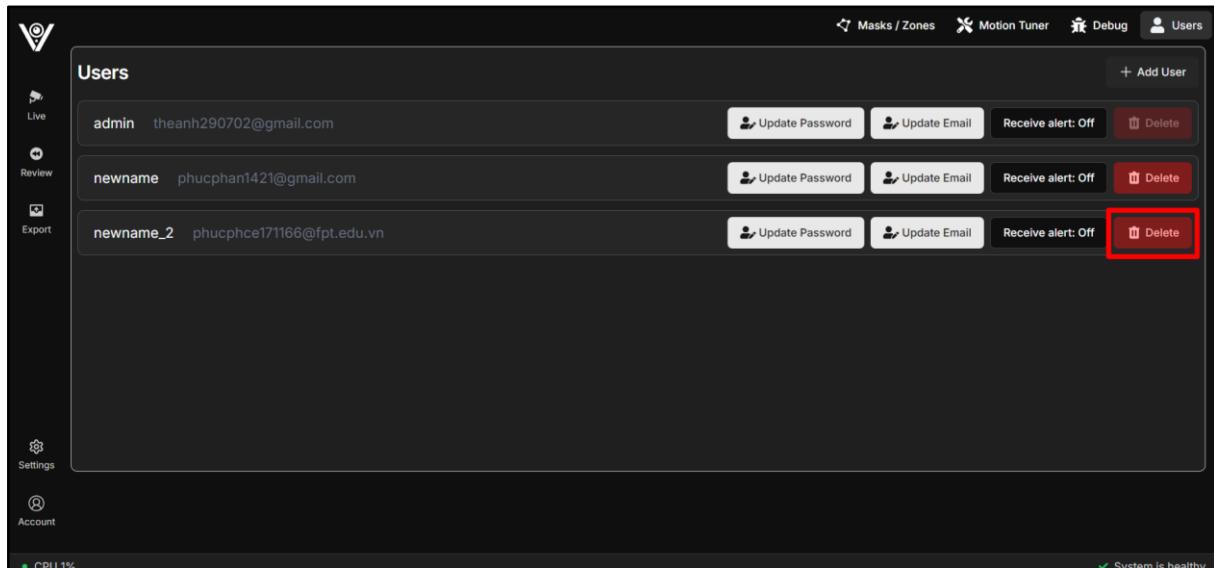


Figure 322. Manage users - Delete user

Confirm delete user

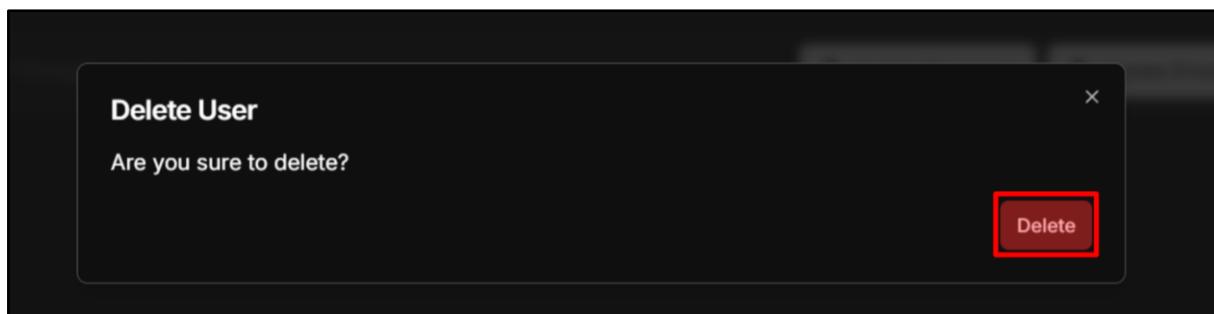


Figure 323. Manage users - Delete user

Delete user successfully

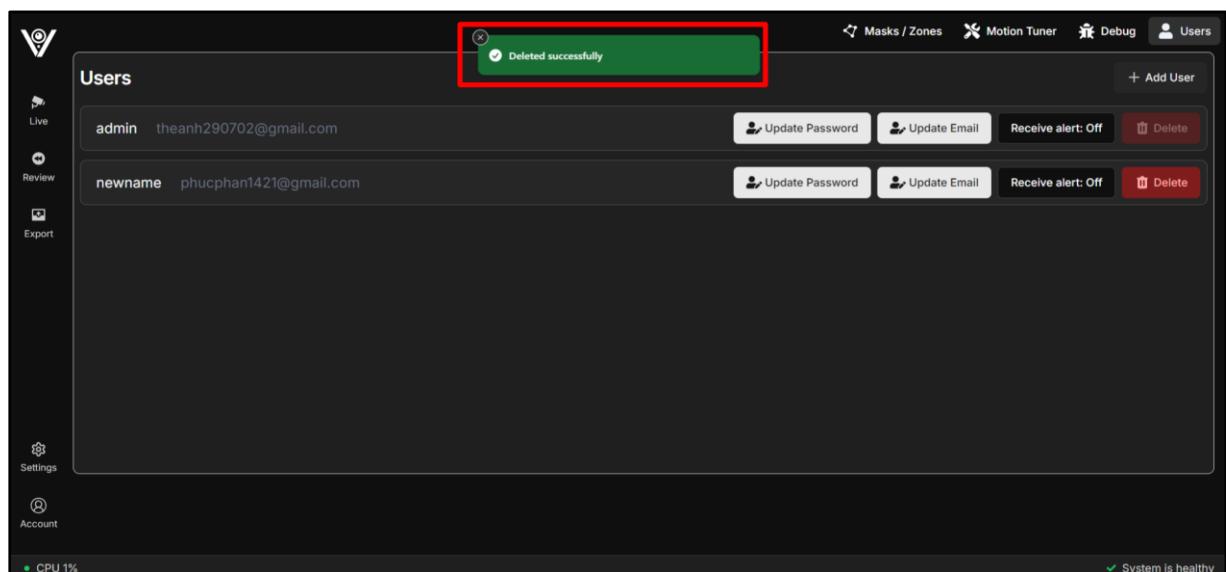


Figure 324. Manage users - Delete user

3.3.2.3.6 Enable/disable receive alert by email

To enable receive alert by email, click on the button **Receive alert: Off**

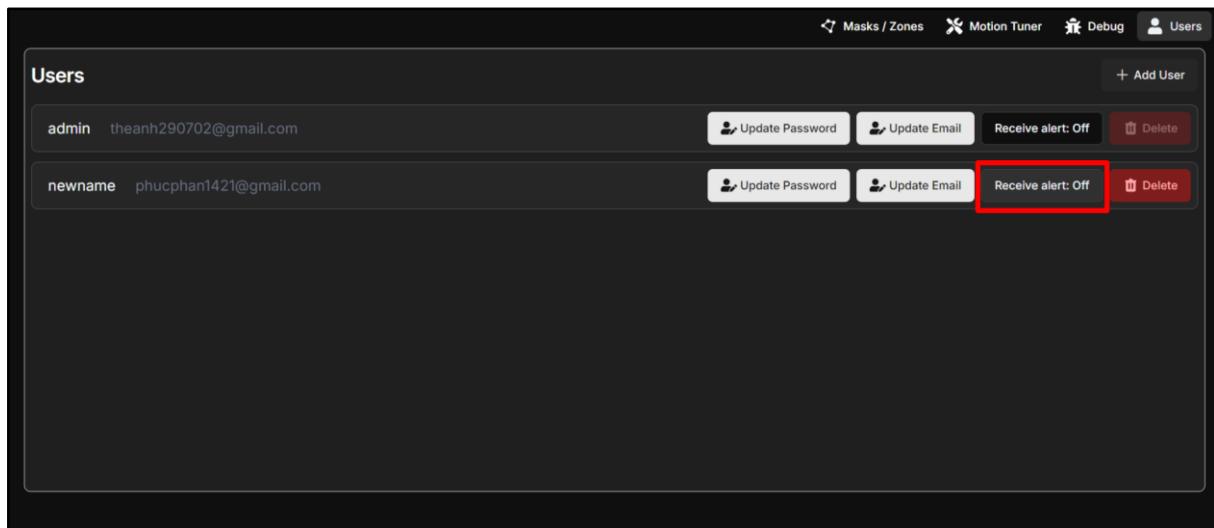


Figure 325. Manage users - Enable/disable receive alert by email

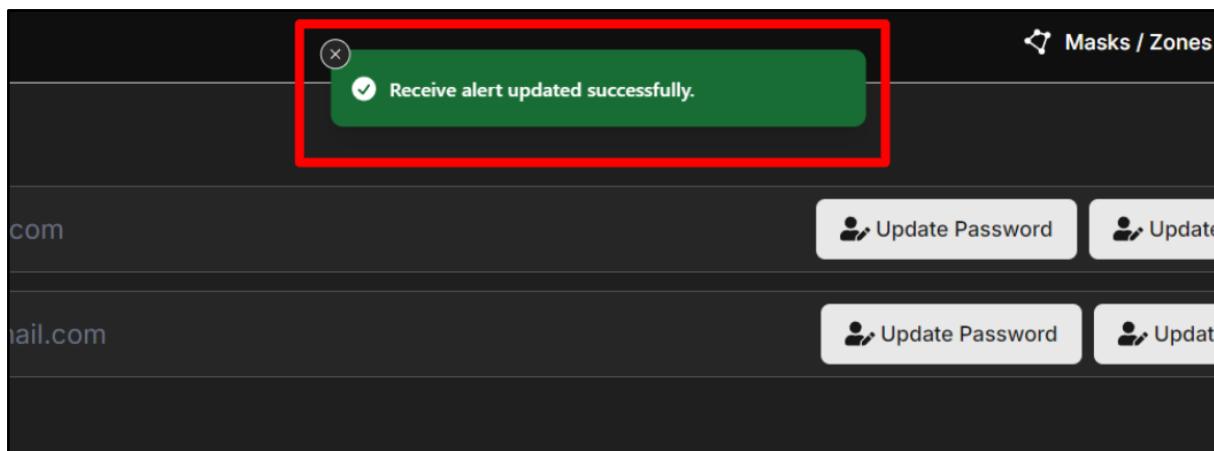


Figure 326. Manage users - Enable/disable receive alert by email

To disable receive alert by email, click on the button **Receive alert: On**

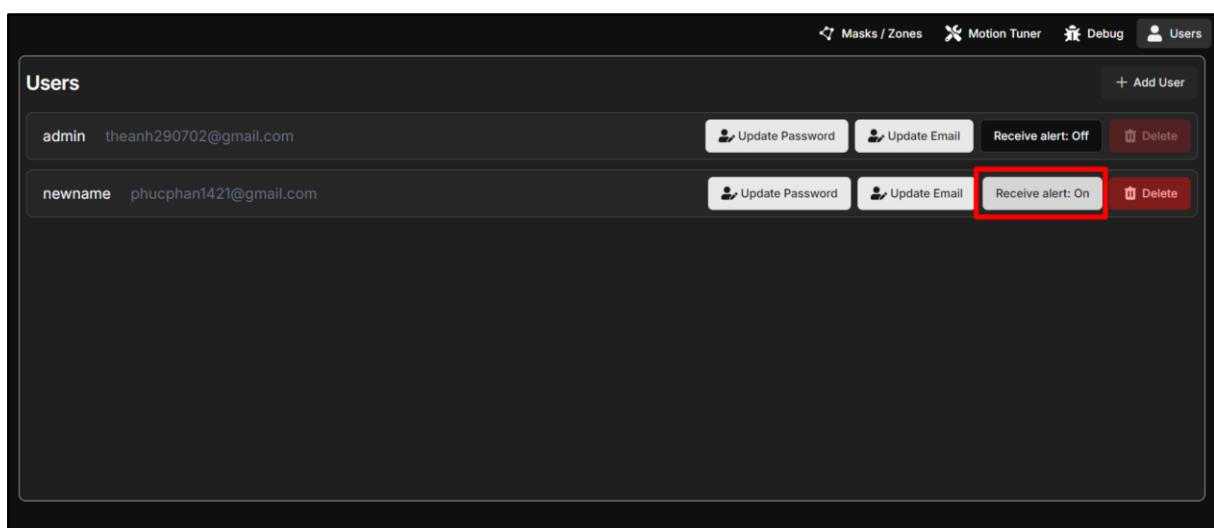


Figure 327. Manage users - Enable/disable receive alert by email

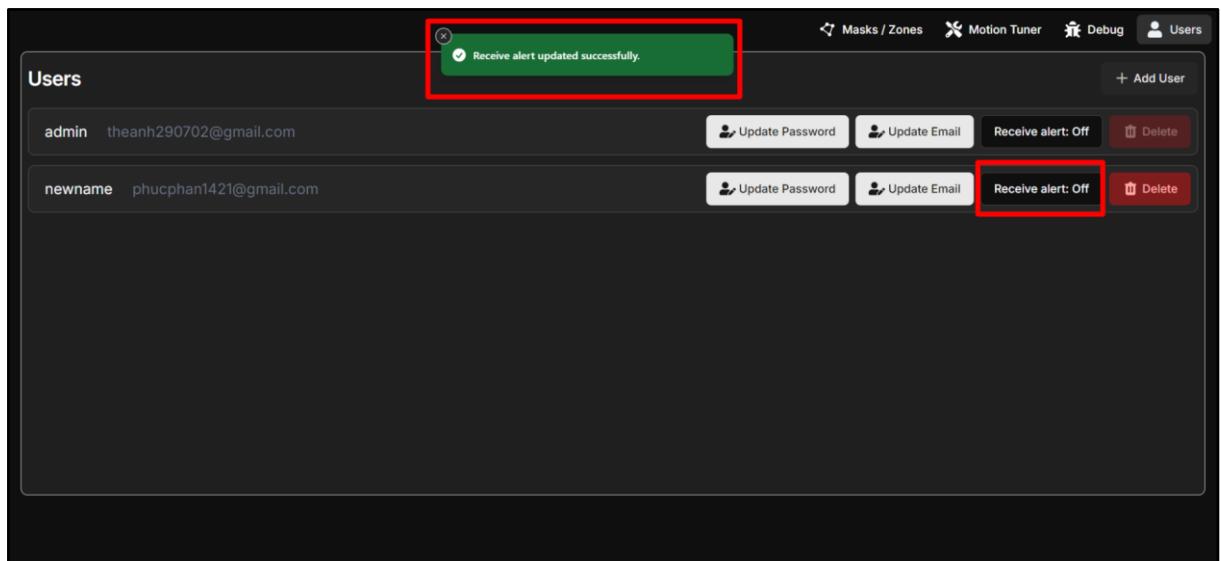


Figure 328. Manage users - Enable/disable receive alert by email

3.3.2.4 Get shareable link

To get shareable link, click on “Account”

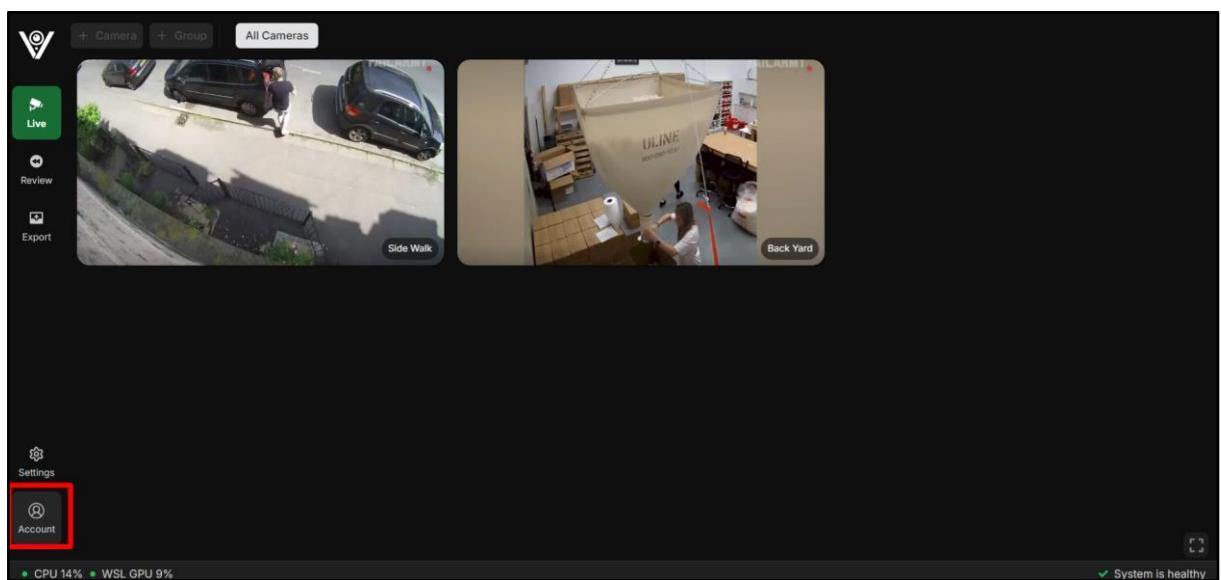


Figure 329. Get shareable link

A pop up is show as below:

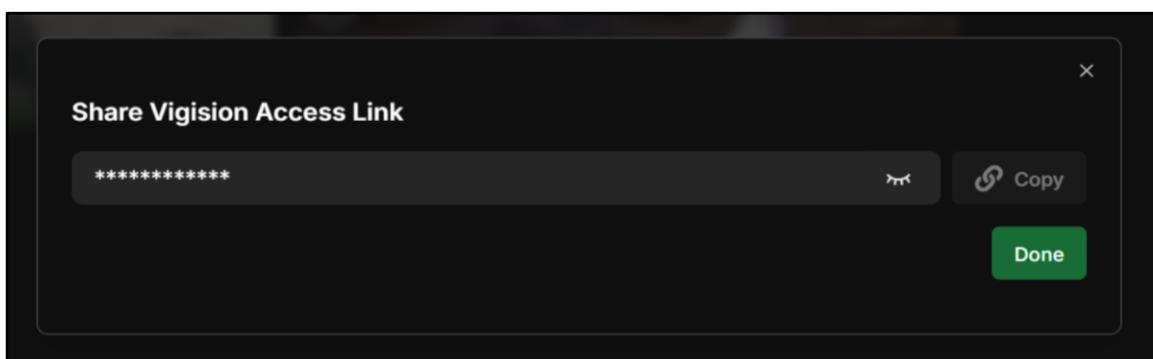


Figure 330. Get shareable link

You can click on the icon “Eye” to show the link:

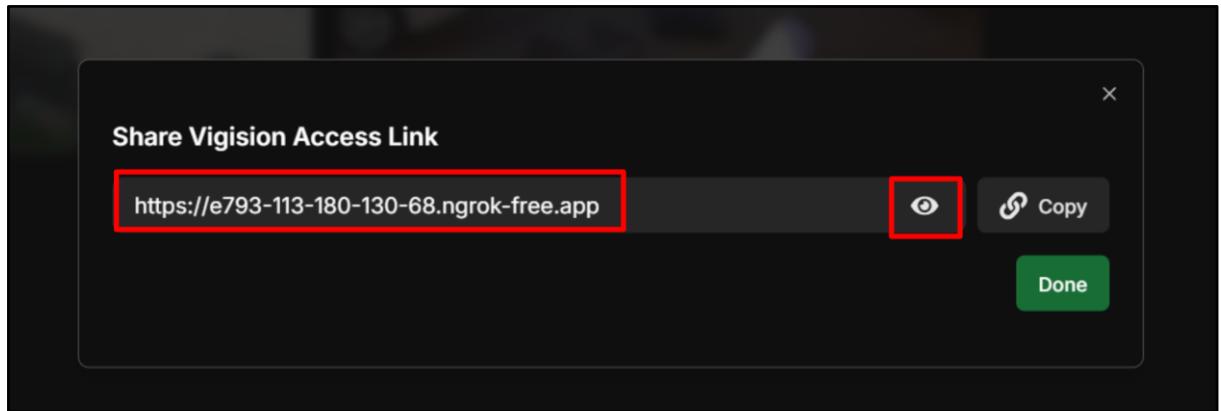


Figure 331. Get shareable link

Click “Copy” to copy the link

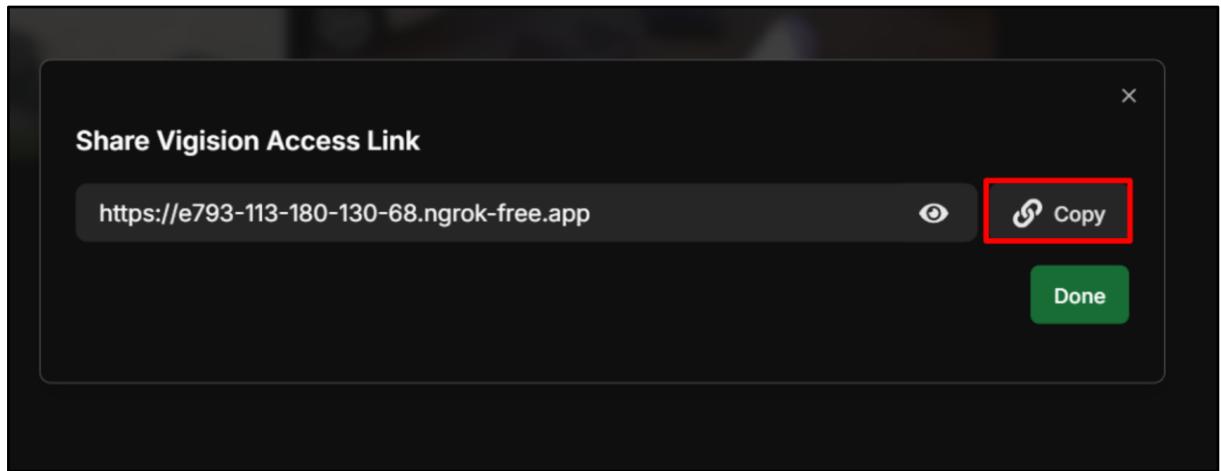


Figure 332. Get shareable link

After that, a message will displayed like below:

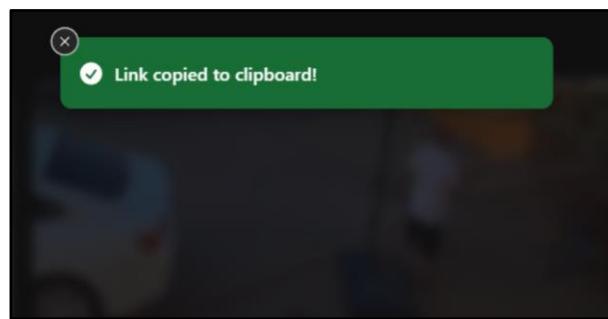


Figure 333. Get shareable link

Finally, click on the button “Done” to close the pop-up.

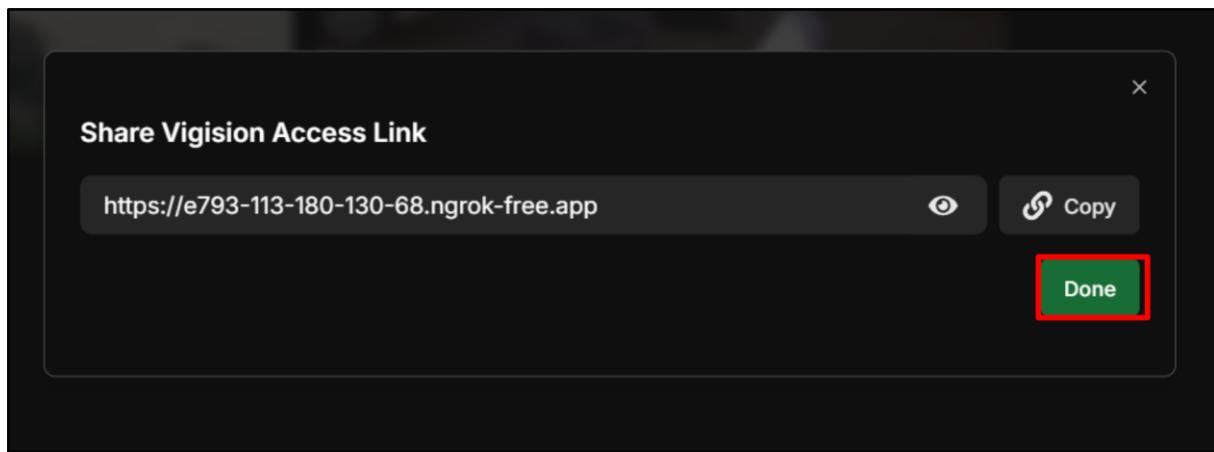


Figure 334. Get shareable link

3.3.2.5 View live cameras

- Step 1: After logged in successfully, the app will automatically navigate to the “Live Dashboard” page. This can also be done by clicking the “Live” option in the sidebar. If there are no cameras, there will be the message “There are no cameras in this group”. If there are cameras, the “Live Dashboard” page will display a grid of available camera feeds. Each feed might show a live video stream, camera detection status and the name of the camera.

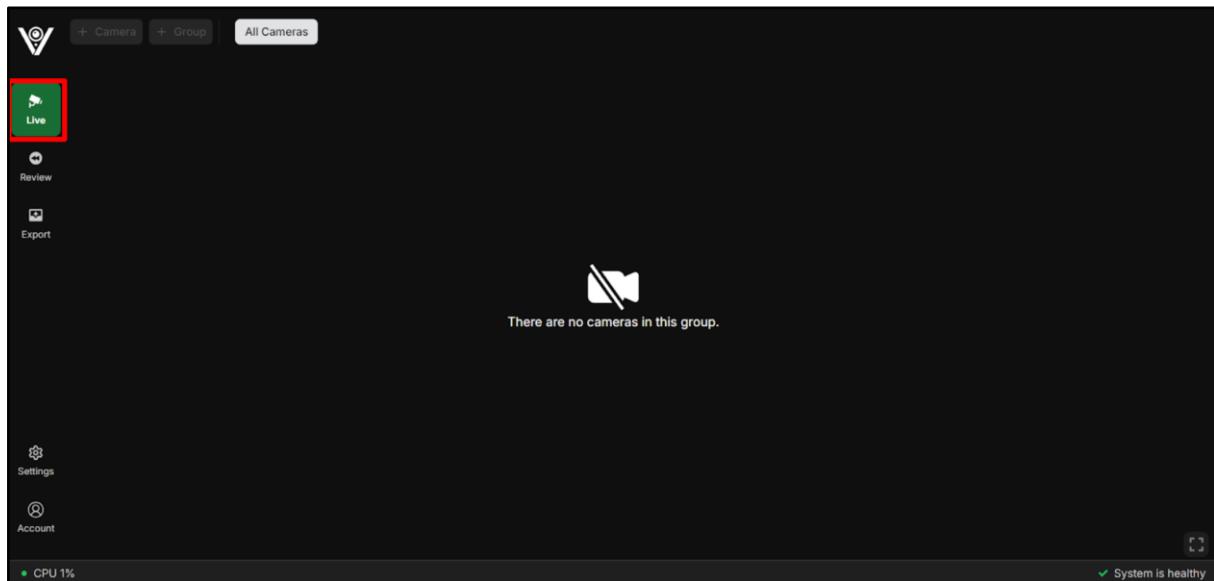


Figure 335. View live cameras

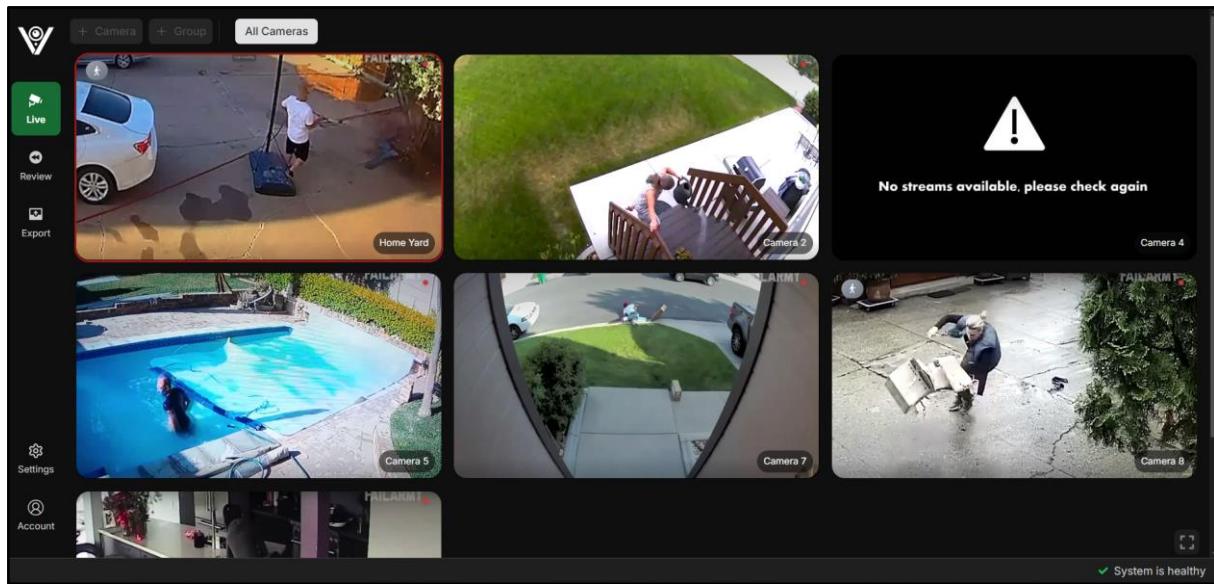


Figure 336. View live cameras

- Step 2: Clicking on a specific camera group will typically navigate you to a sub-dashboard specific to that group. This view might display the live video streams from all cameras within the selected groups if available.

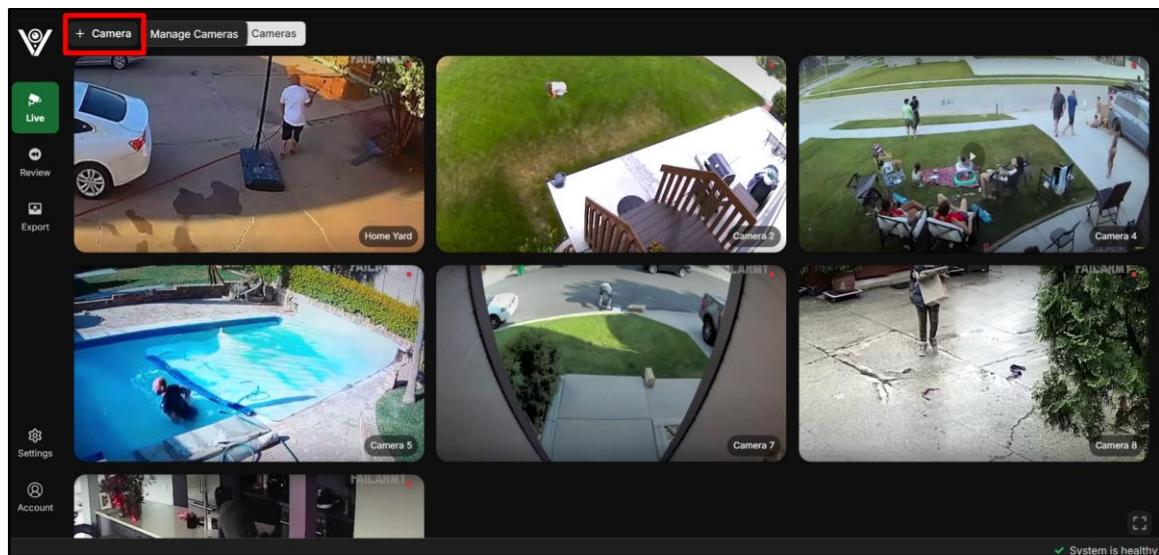


Figure 337. View live cameras

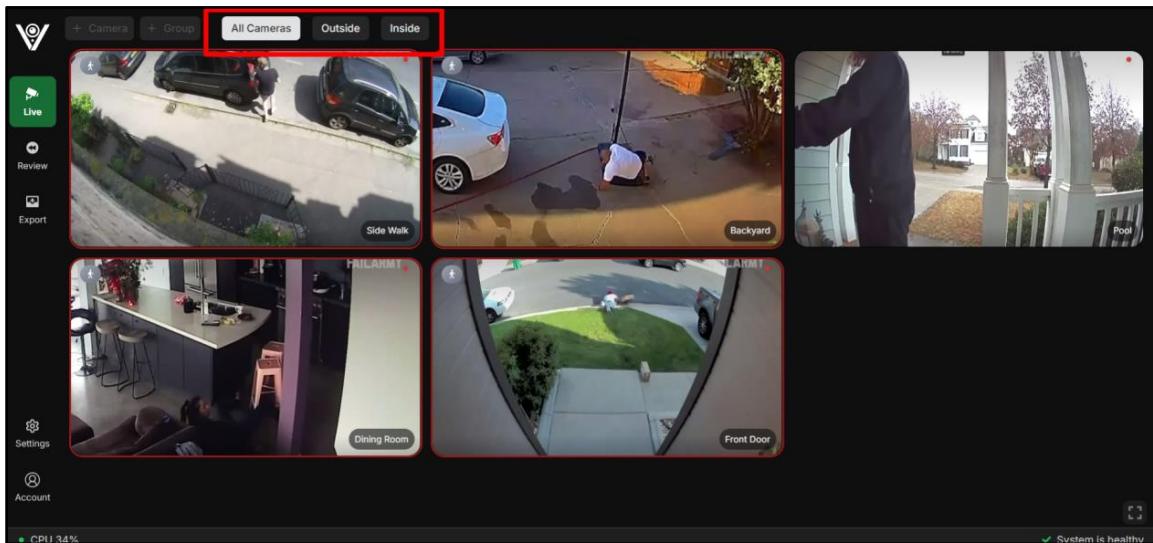


Figure 338. View live cameras

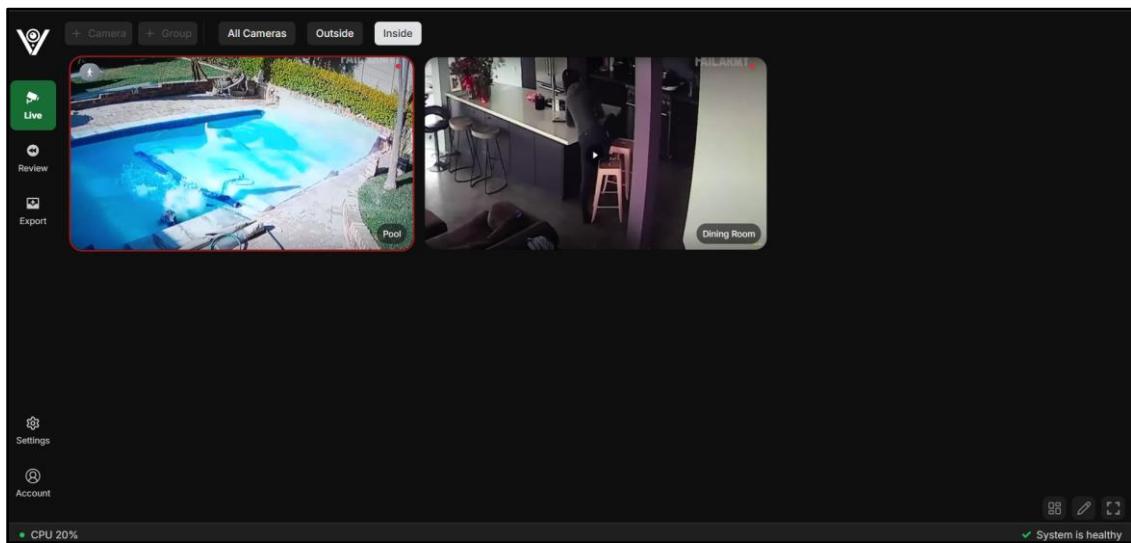


Figure 339. View live cameras

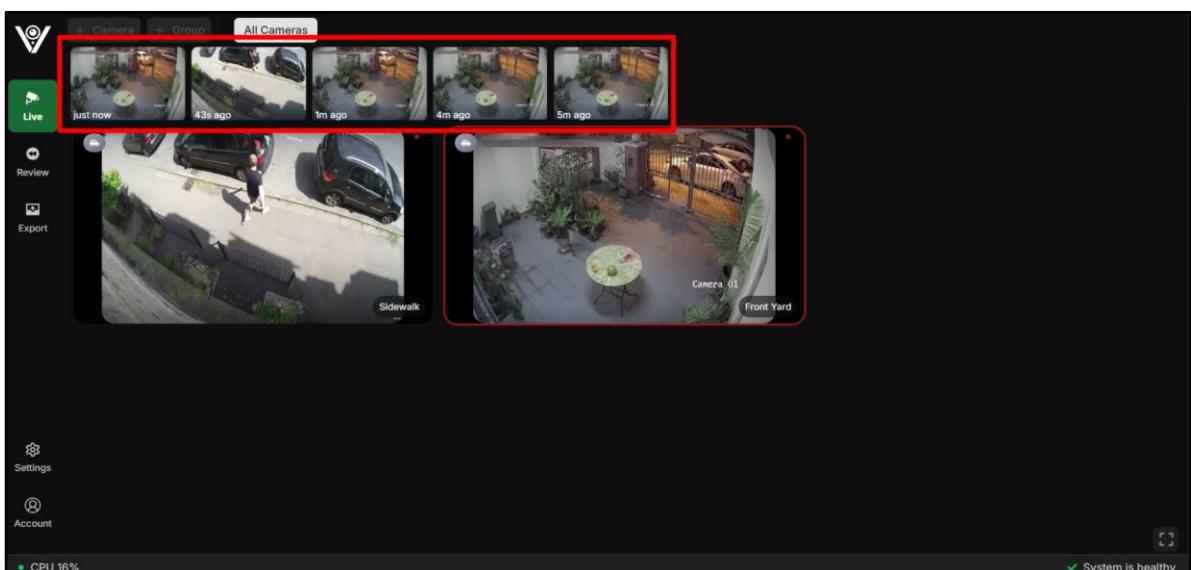


Figure 340. View live cameras

3.3.2.6 Manage cameras

3.3.2.6.1 View list of cameras

- Step 1: On the “Live Dashboard” page, click on button “+Camera” as shown below

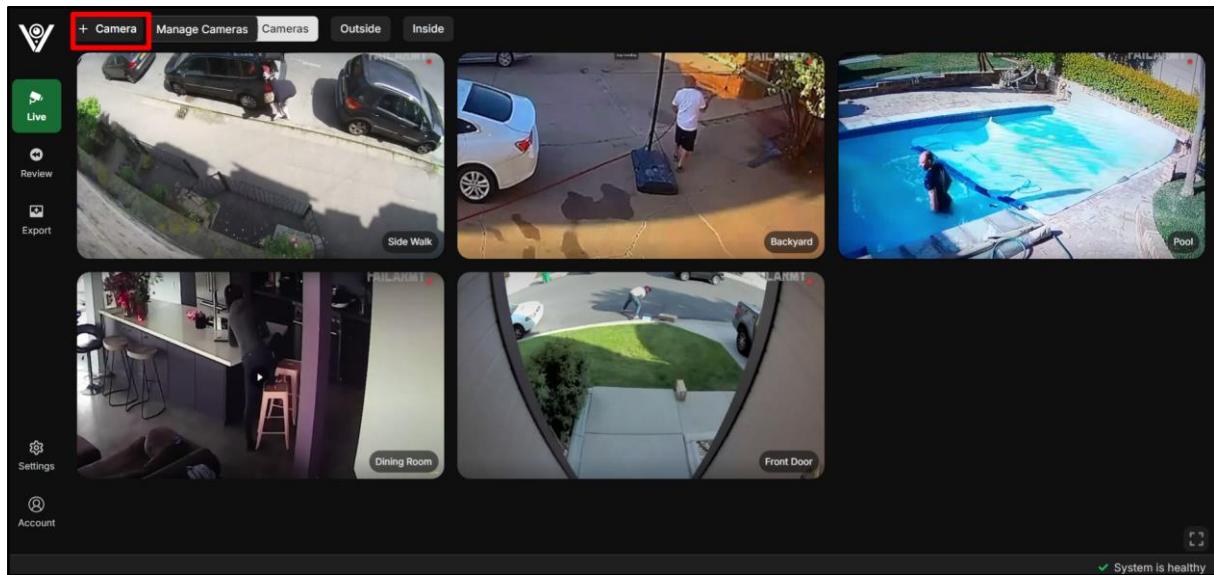


Figure 341. Manage cameras - View list of cameras

- Step 2: The camera list will appear right immediately.

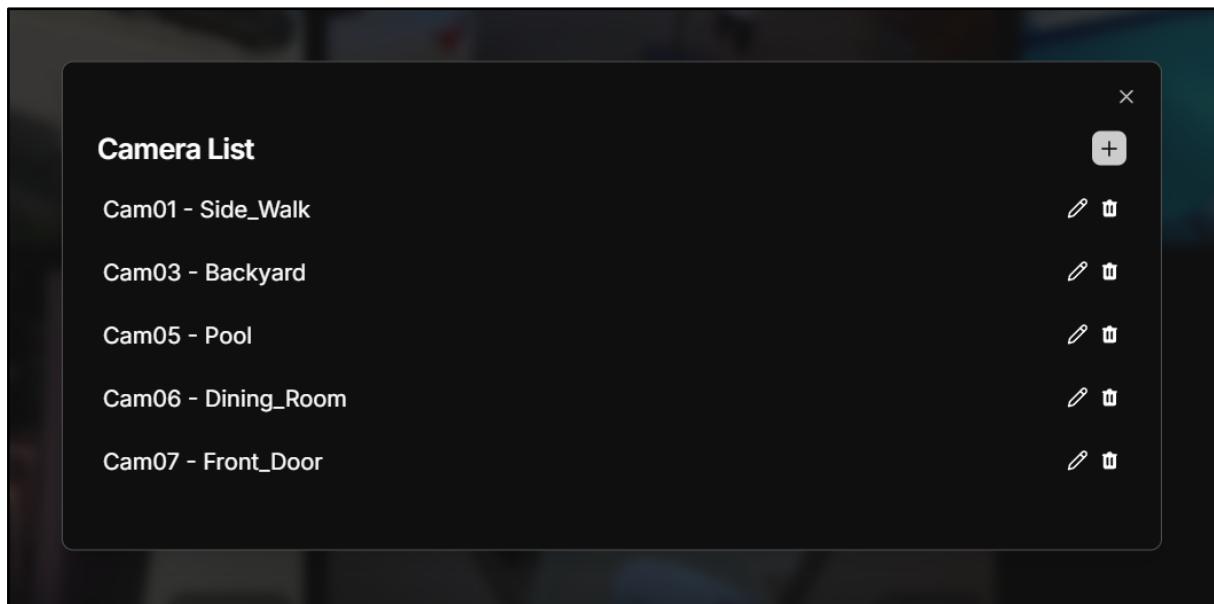


Figure 342. Manage cameras - View list of cameras

3.3.2.6.2 Create camera

- Step 1: On the camera list pop up as below, click on the button “Add” to add new camera

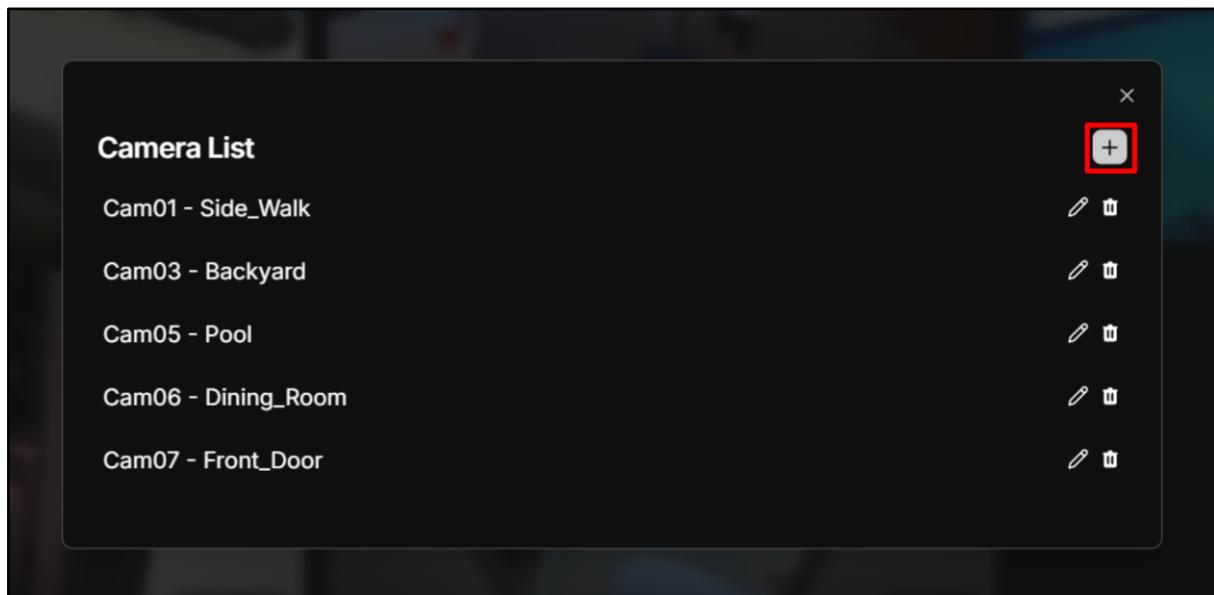


Figure 343. Manage cameras - Create camera

- Step 2: Enter camera configuration

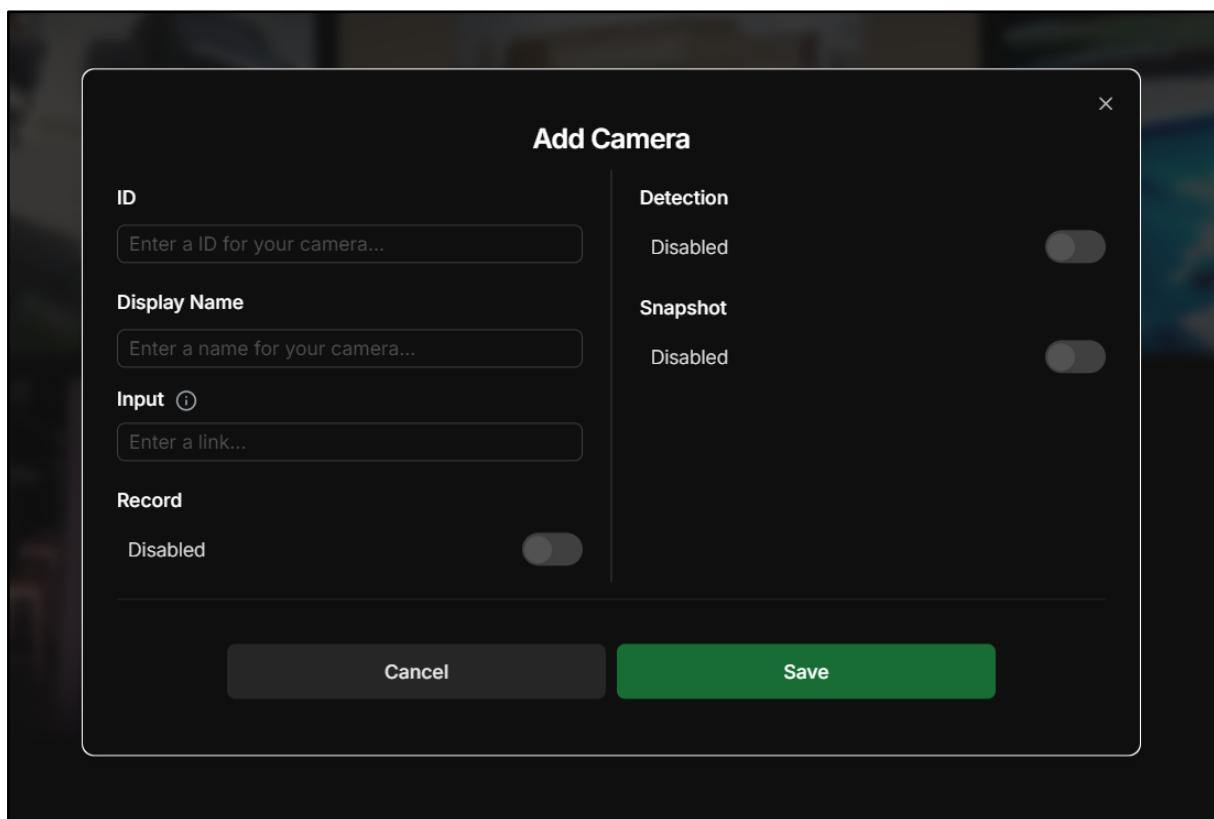


Figure 344. Manage cameras - Create camera

- Step 3: Enter camera ID. The ID must be unique for each camera.

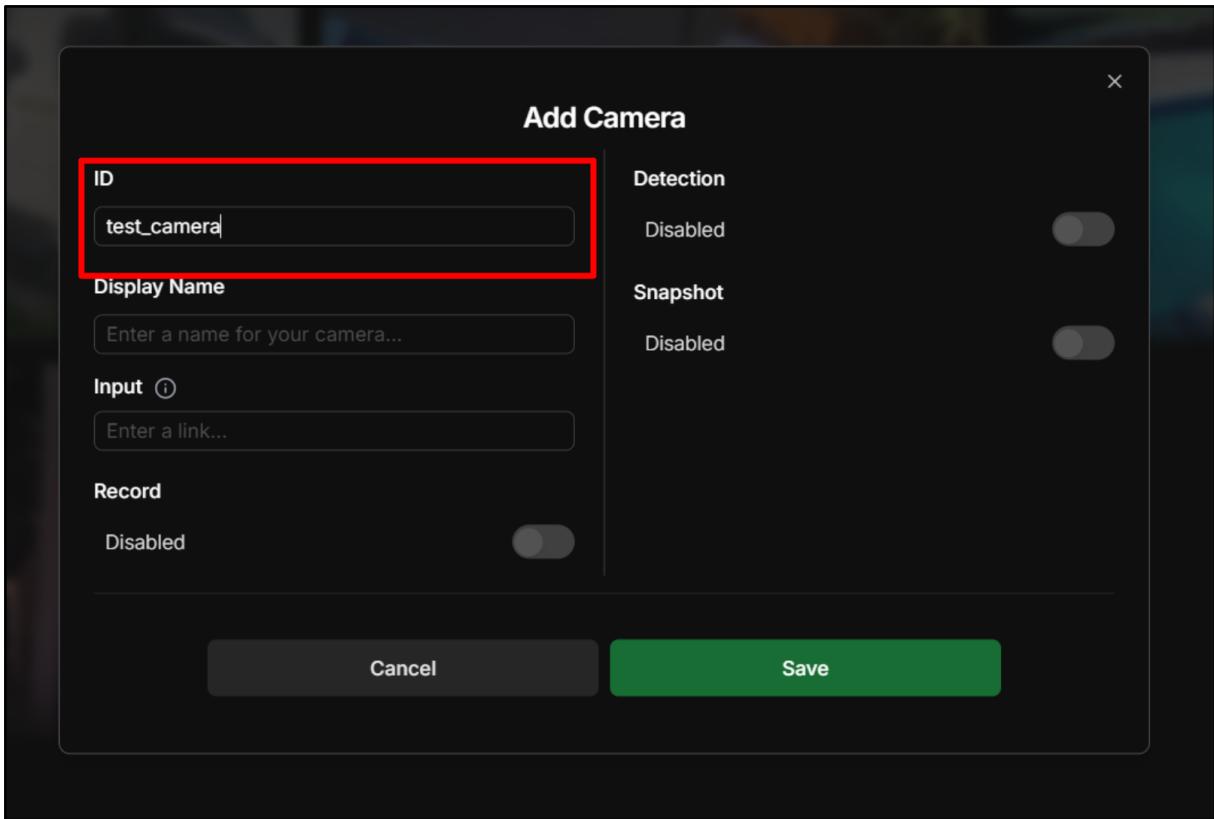


Figure 345. Manage cameras - Create camera

- Step 3: Enter camera Display Name. Display Name is the name you want to see it in Live Dashboard and other place. This allow you to distinguish with other cameras.

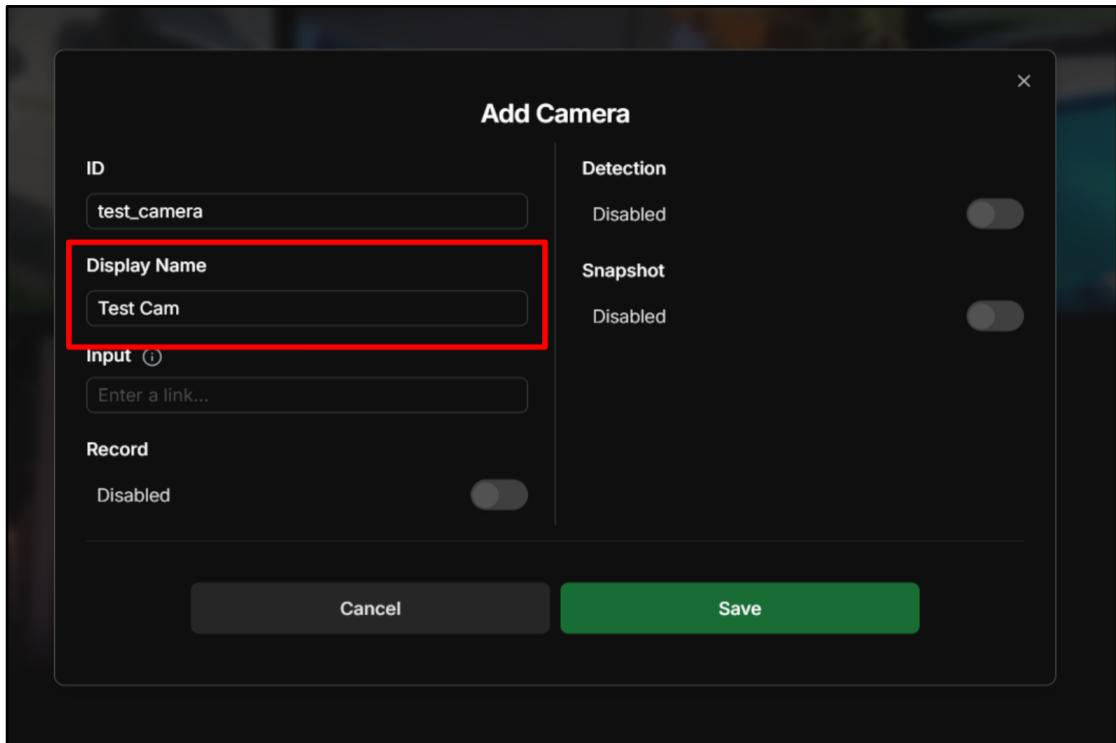


Figure 346. Manage cameras - Create camera

- Step 4: Enter camera input. Input is the address of your IP camera in its local network

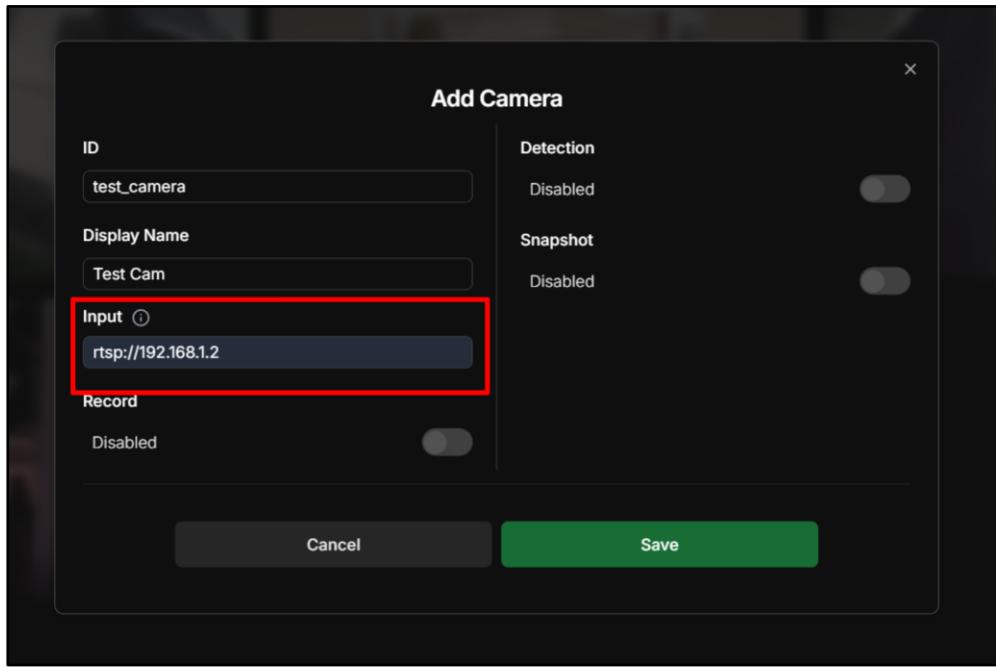


Figure 347. Manage cameras - Create camera

- Step 5: If you want the camera to take record of events, you can enable it by toggling the button on the right side of each line under **Record** section. Once recording is enabled, the recording options are displayed and can be customized. To turn it off, toggle off that button.

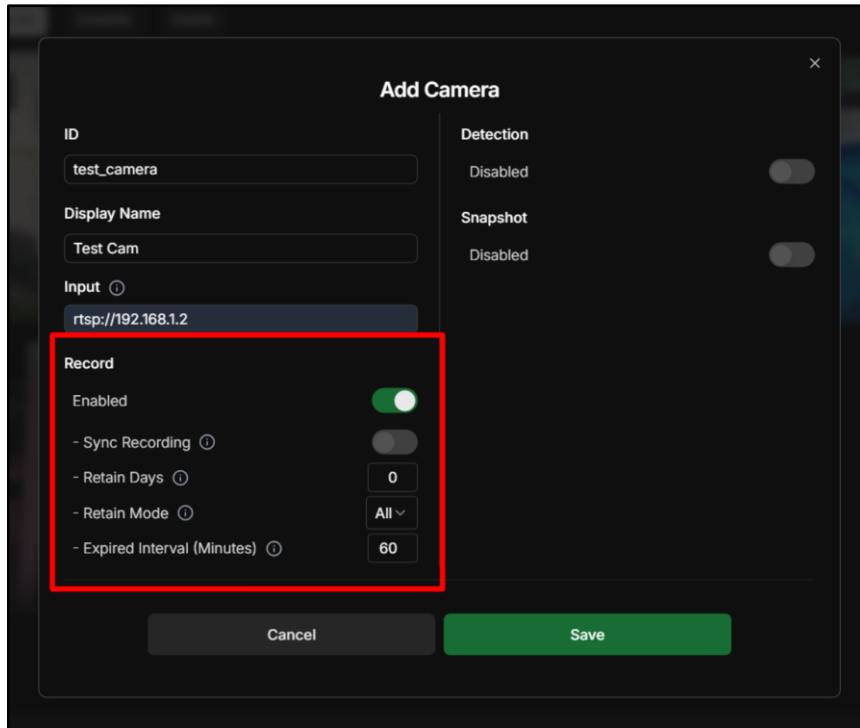


Figure 348. Manage cameras - Create camera

- Step 6: Similarly, if you want the camera to detect objects, you can enable it by toggling the button on the right side of each line under **Detection** section. Once detection is enabled, the detection options are displayed and can be customized. To turn it off, toggle off that button.

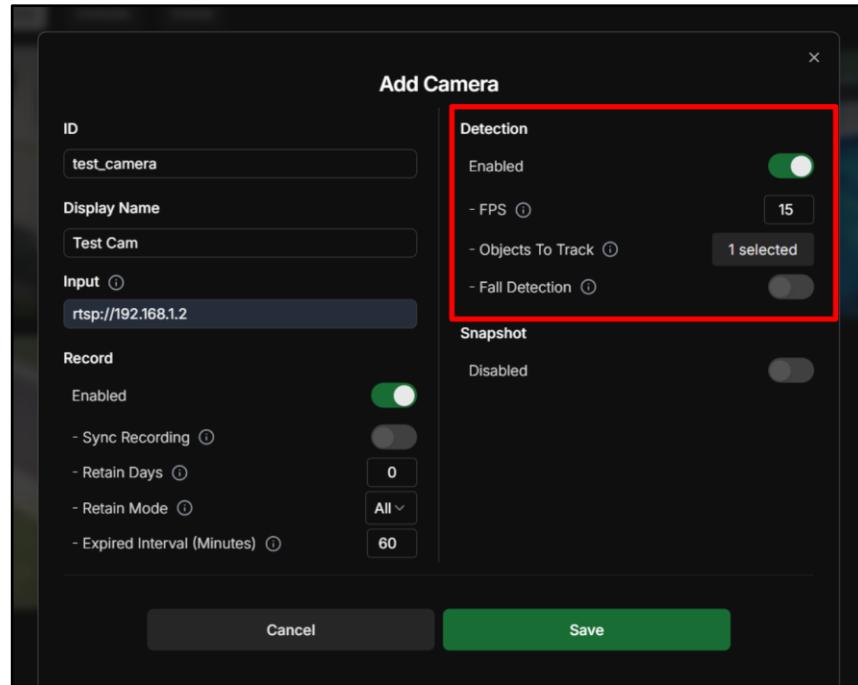


Figure 349. Manage cameras - Create camera

- Step 7: In this detection section, you can select the object for the camera to track and detect by clicking on the corresponding object box as shown in the image below.

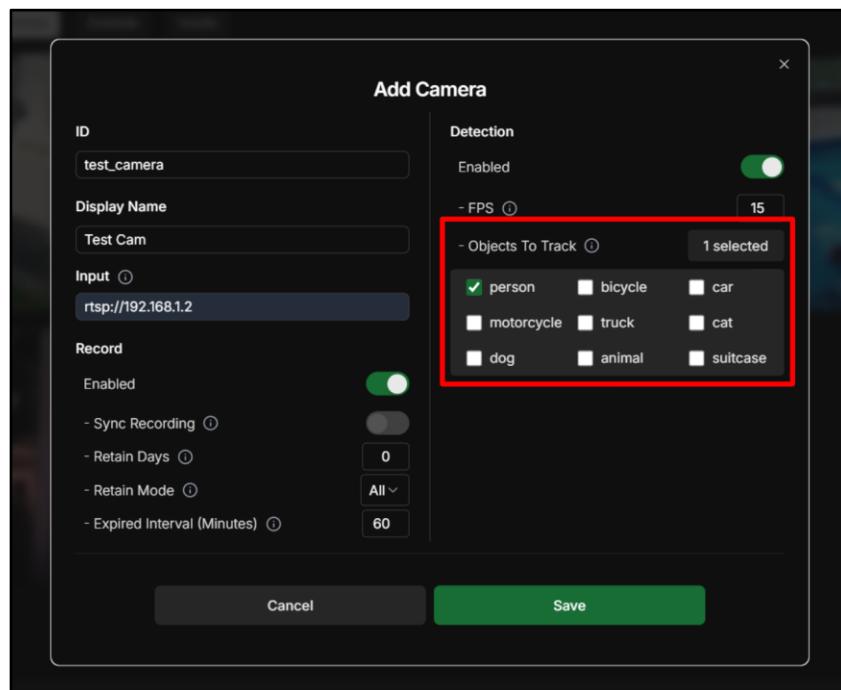


Figure 350. Manage cameras - Create camera

- Step 8: If you want the camera to detect falls, please check on **person** and enable **Fall Detection**.

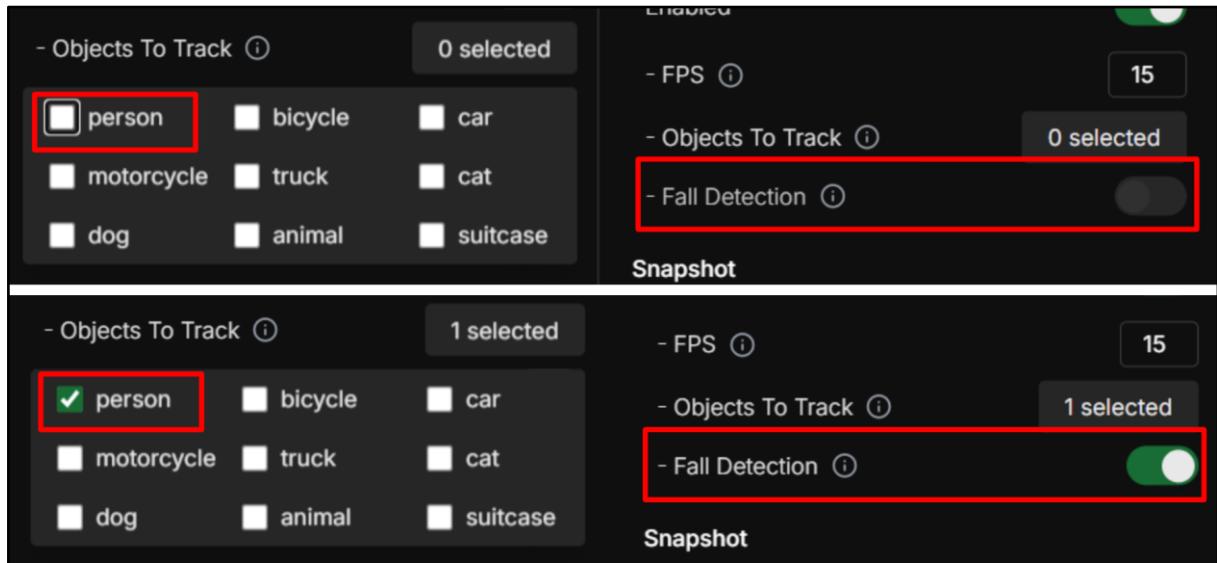


Figure 351. Manage cameras - Create camera

- Step 9: If you want the camera to take snapshot once an event happens, you can enable it by toggling the button on the right side of that line under **Snapshot** section. Once snapshot is enabled, the snapshot options are displayed and can be customized. To turn it off, just toggle off that button.

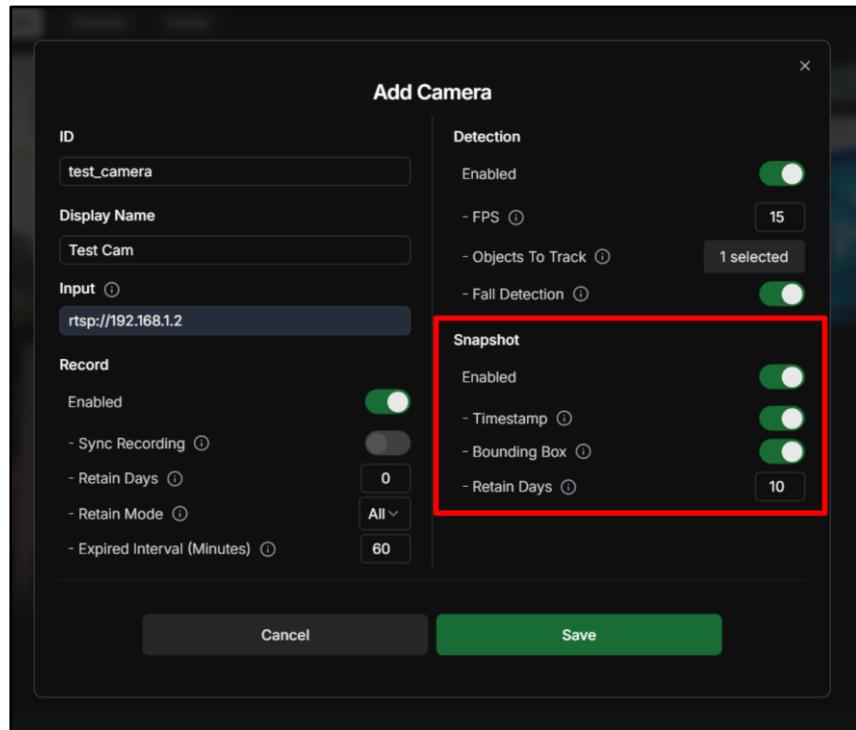


Figure 352. Manage cameras - Create camera

- Step 9: Click on “Save” button to add this new camera. If the camera is add successfully you can see a message saying that “**Camera <name> has been saved. Restart is required for changes to take effect**”. If you want to restart Vigision, click on button “Restart Now”

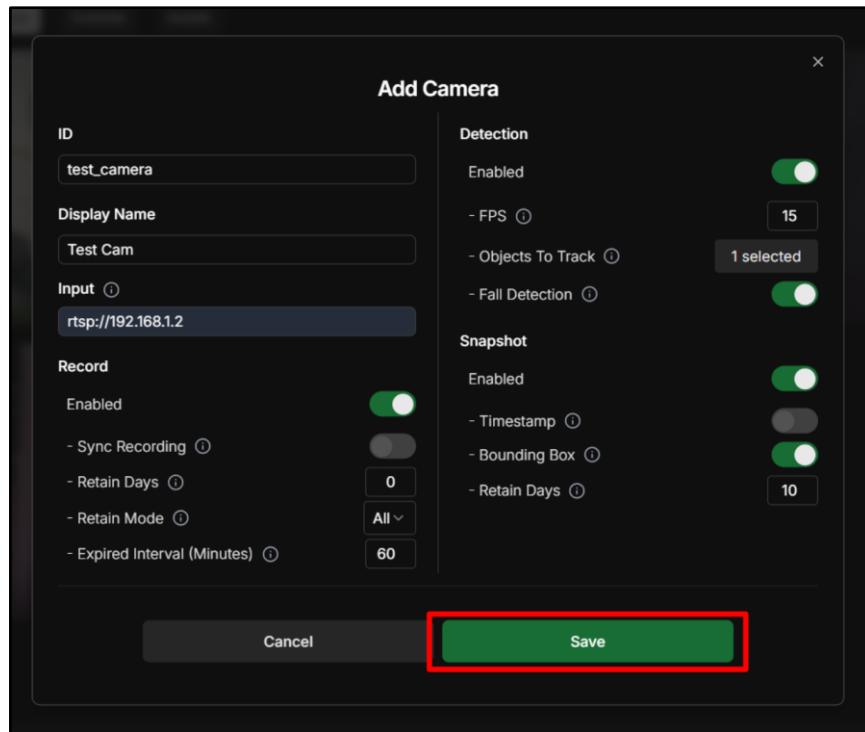


Figure 353. Manage cameras - Create camera

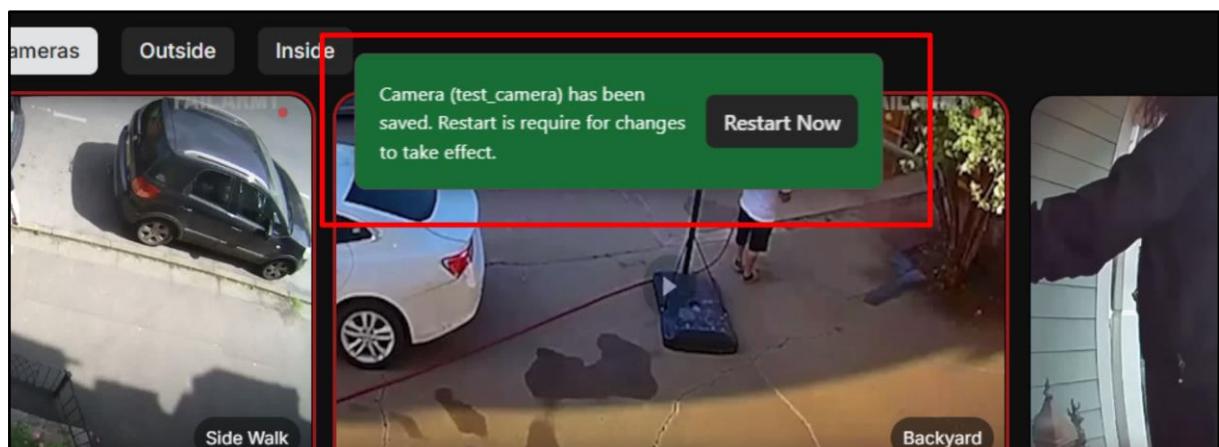


Figure 354. Manage cameras - Create camera

- Step 10: After clicking on button “**Restart Now**”, Vigision is restarting. You can wait for it for around 1 minutes or you can click on button “**Force Reload Now**”.

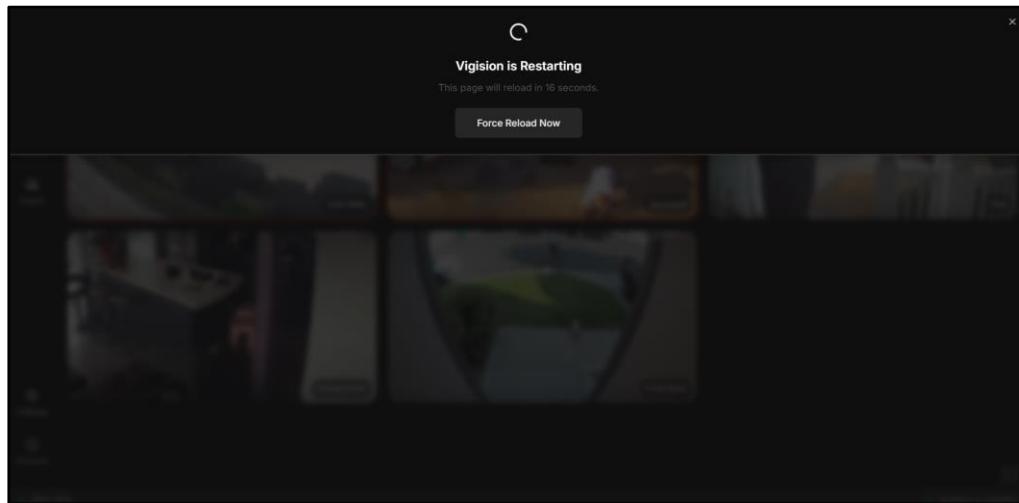


Figure 355. Manage cameras - Create camera

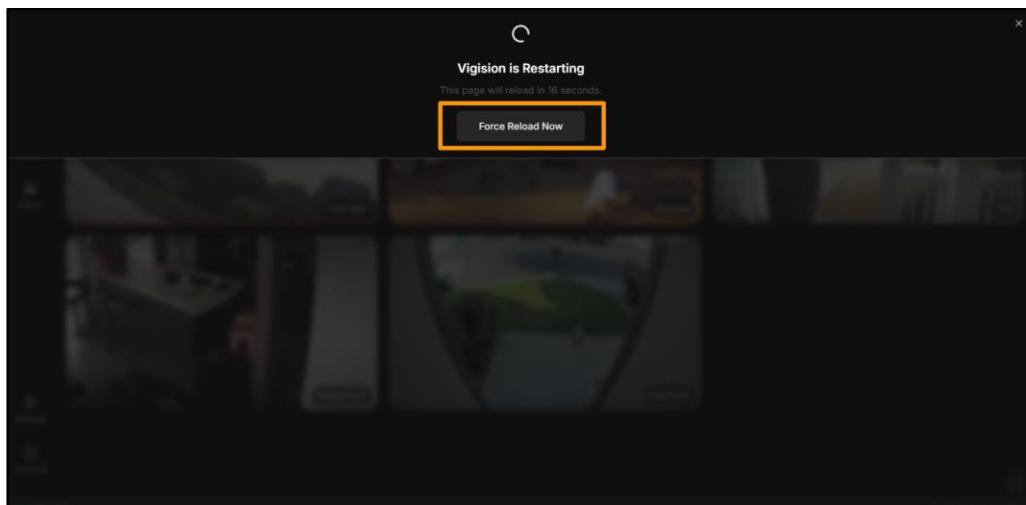


Figure 356. Manage cameras - Create camera

After restarting, you can check your new camera on the Live Dashboard as shown below:

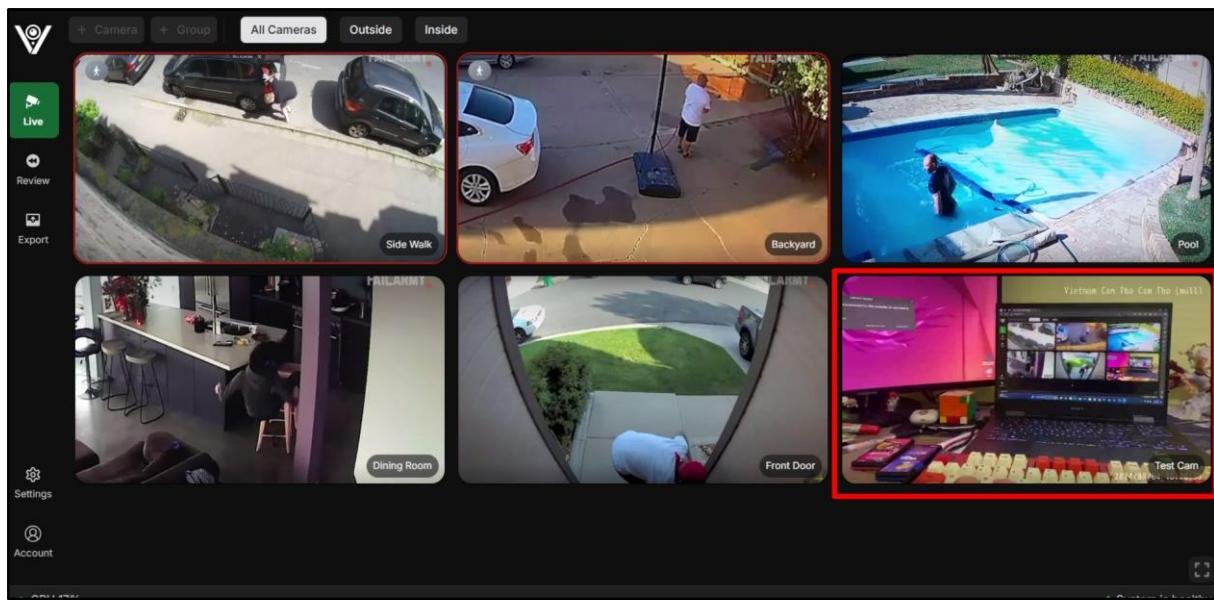


Figure 357. Manage cameras - Create camera

3.3.2.6.3 Update camera

- Step 1: In Live Dashboard page, click on button “+ Camera”

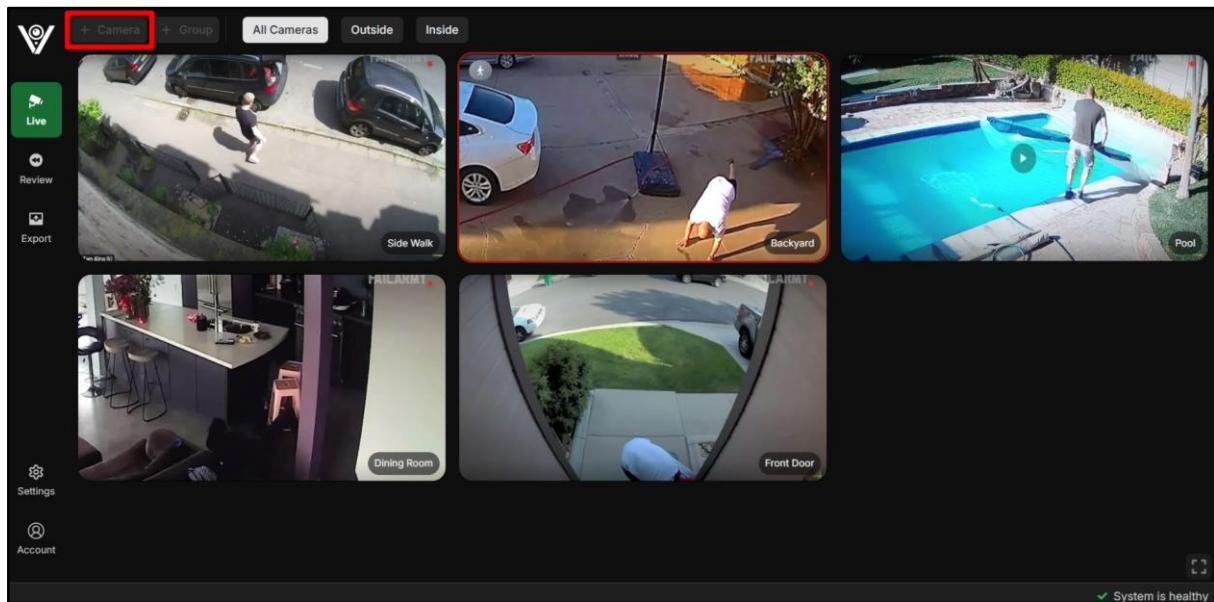


Figure 358. Manage cameras - Update camera

- Step 2: Click on button “Edit” corresponding to the camera needs to be updated.

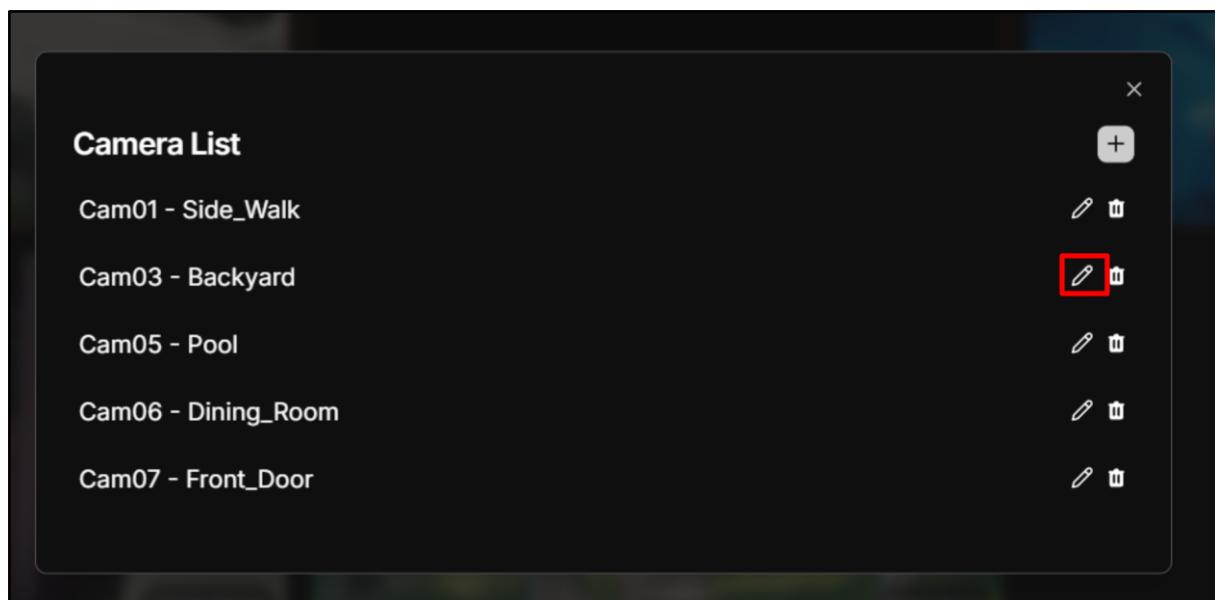


Figure 359. Manage cameras - Update camera

- Step 3: Edit camera configuration. Note that camera ID can not be changed.

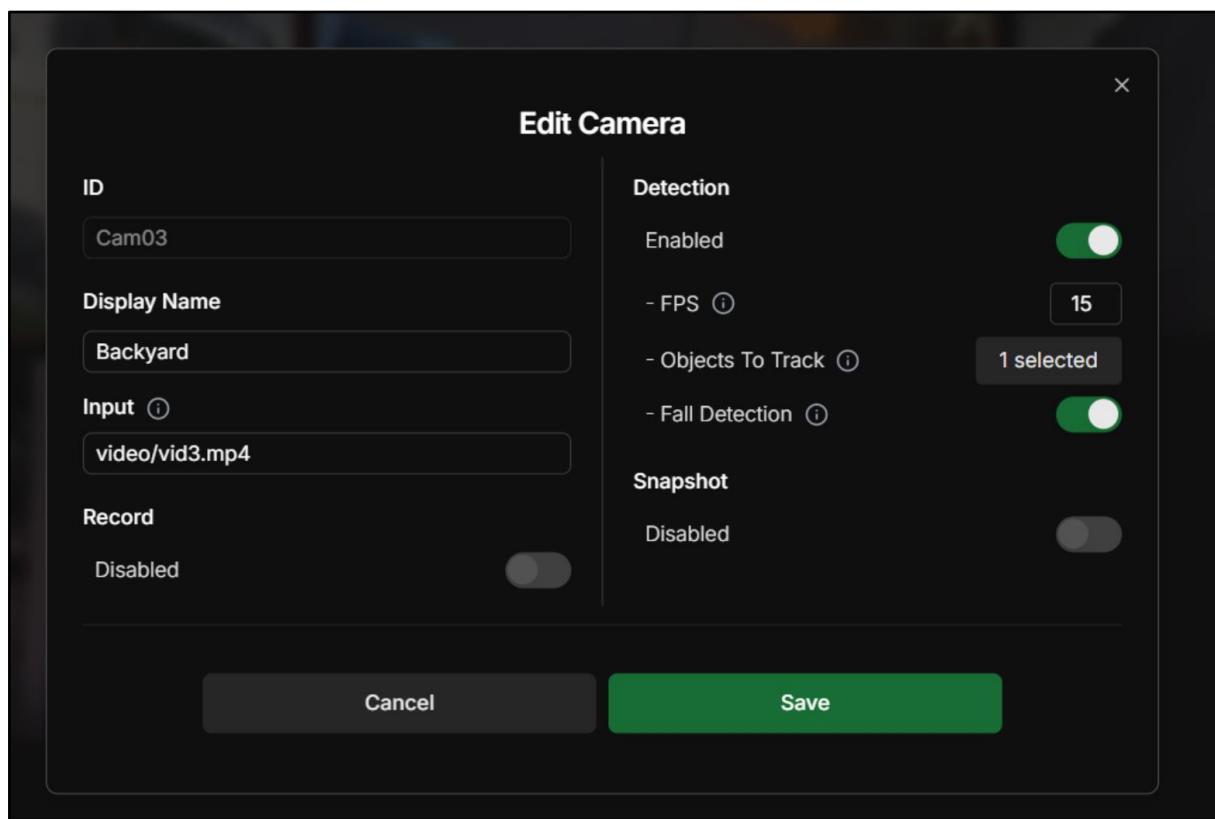


Figure 360. Manage cameras - Update camera

- Step 4: Click on “Save” button to update the camera. If the camera is add successfully you can see a message saying that “**Camera <name> has been saved. Restart is required for changes to take effect**”. If you want to restart Vigision, click on button “Restart Now”

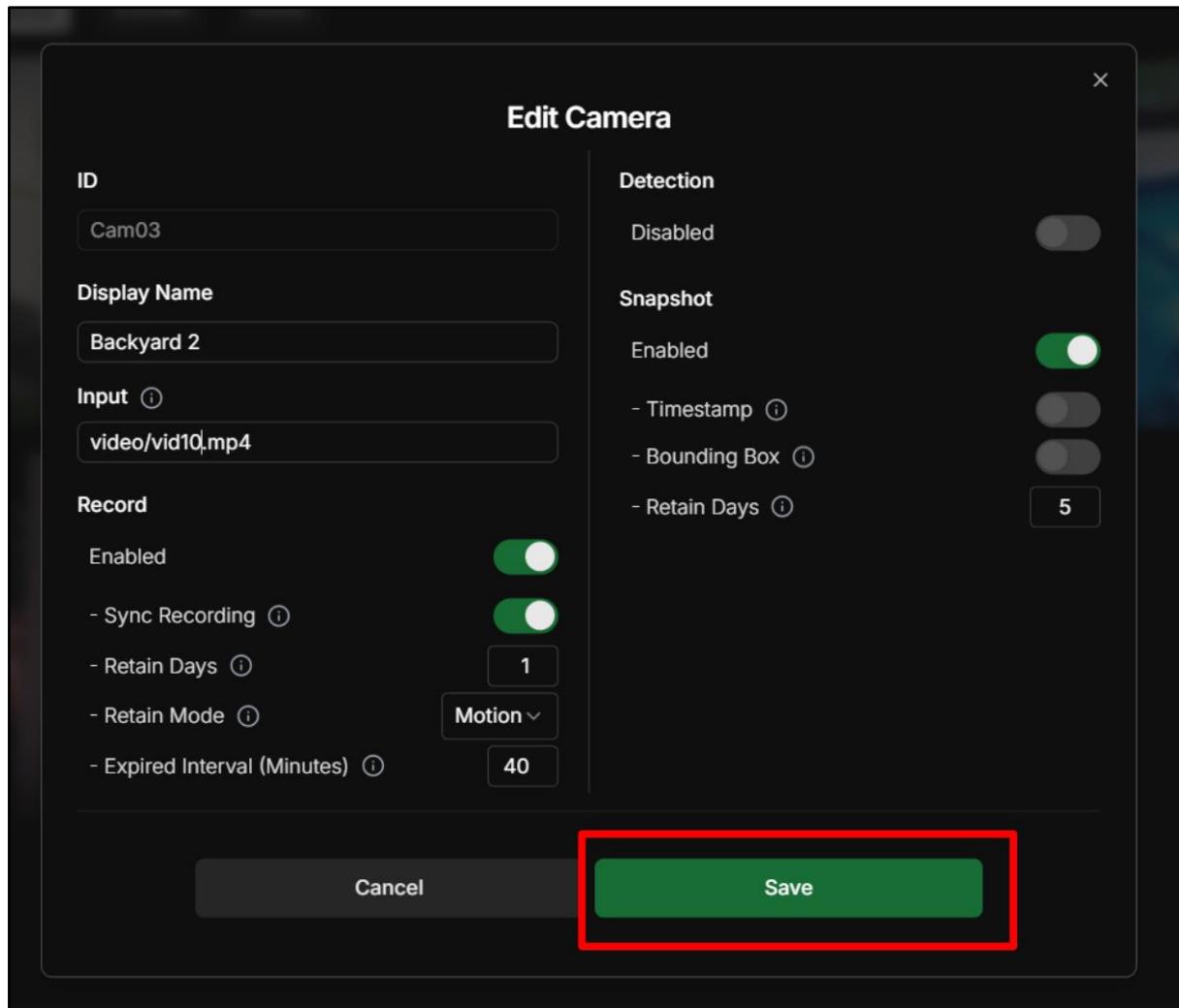


Figure 361. Manage cameras - Update camera

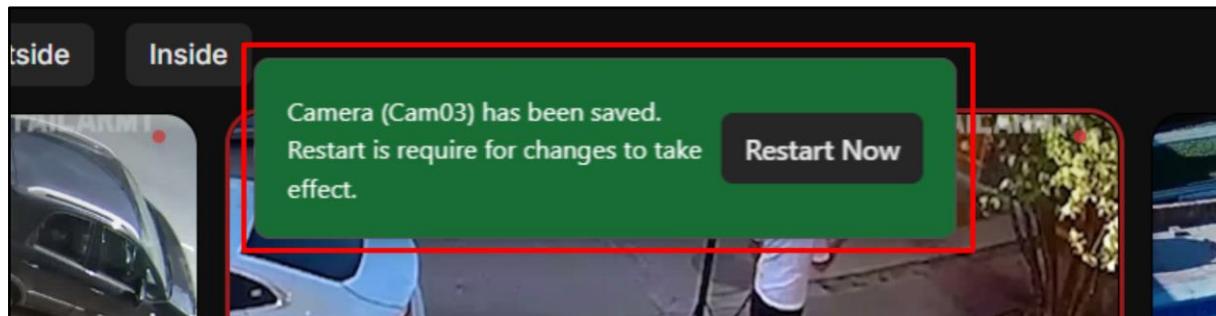


Figure 362. Manage cameras - Update camera

- Step 5: After clicking on button “**Restart Now**”, Vigision is restarting. You can wait for it for around 1 minutes or you can click on button “**Force Reload Now**”.

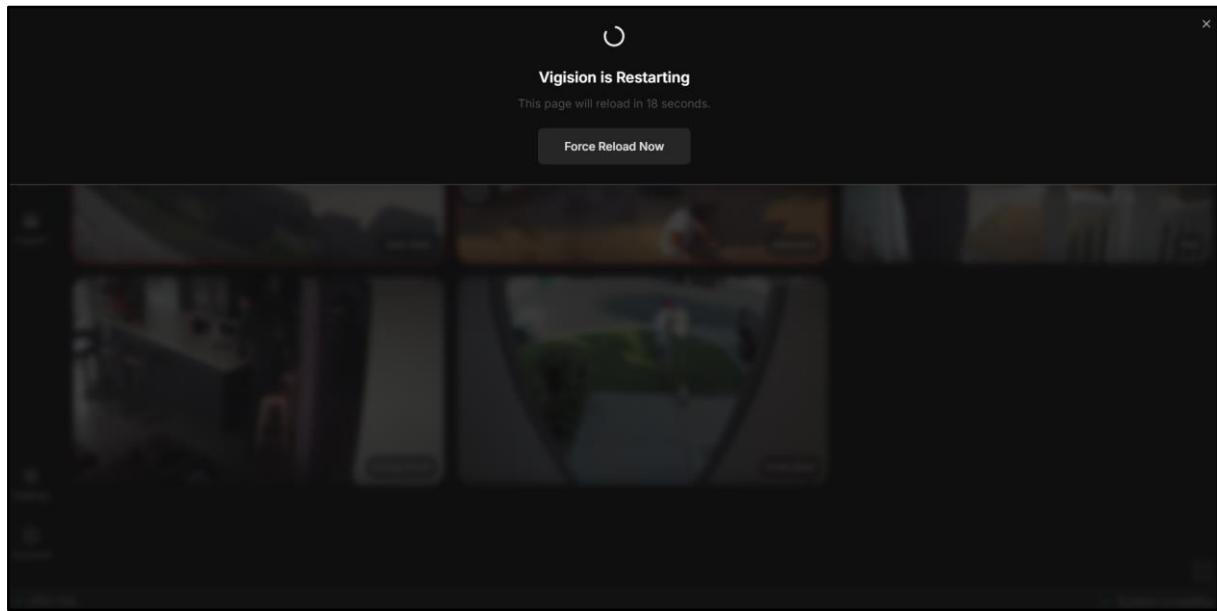


Figure 363. Manage cameras - Update camera

After restarting, you can check if the camera is updated on the Live Dashboard as shown below:

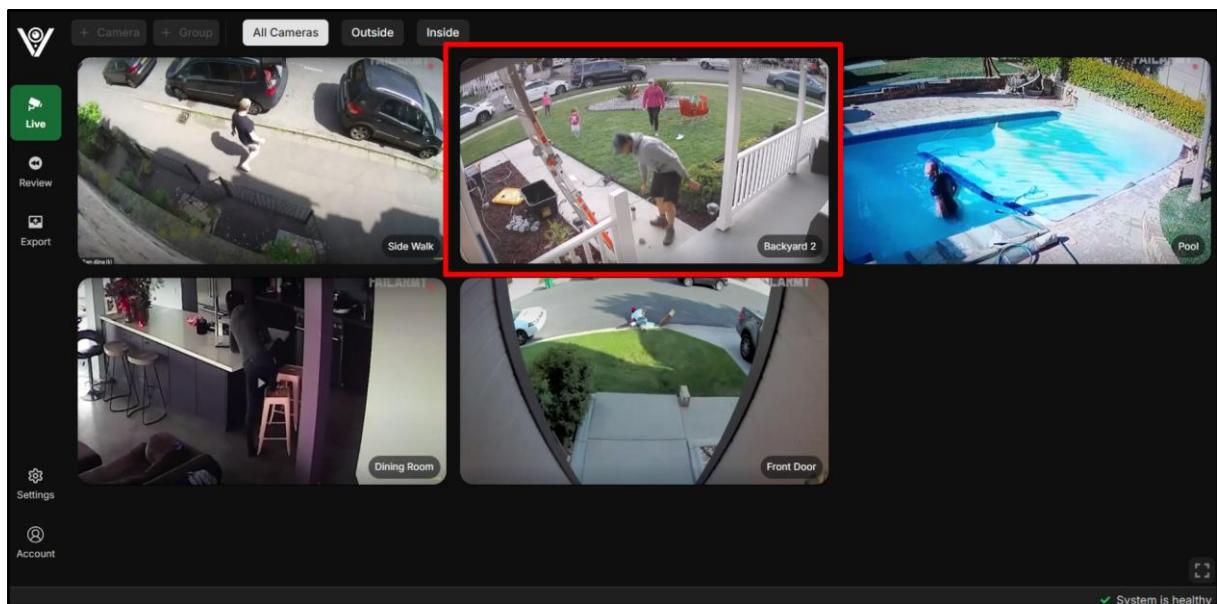


Figure 364. Manage cameras - Update camera

3.3.2.6.4 Delete camera

- Step 1: In Live Dashboard page, click on button “+ Camera”

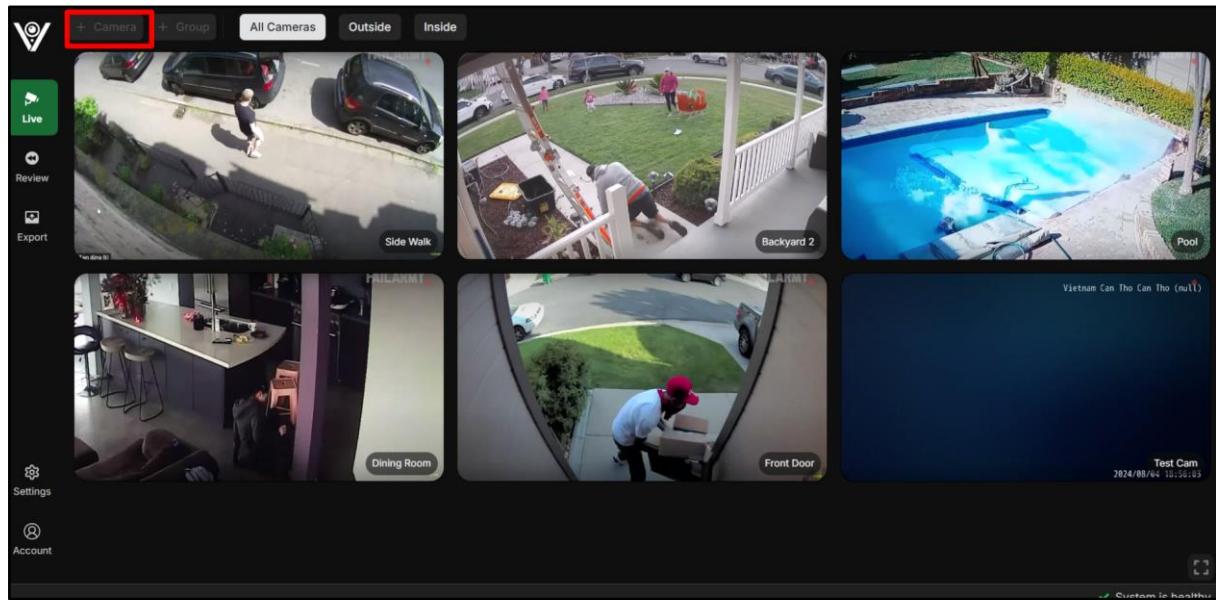


Figure 365. Manage cameras - Delete camera

- Step 2: Click on button “Delete” corresponding to the camera needs to be deleted.

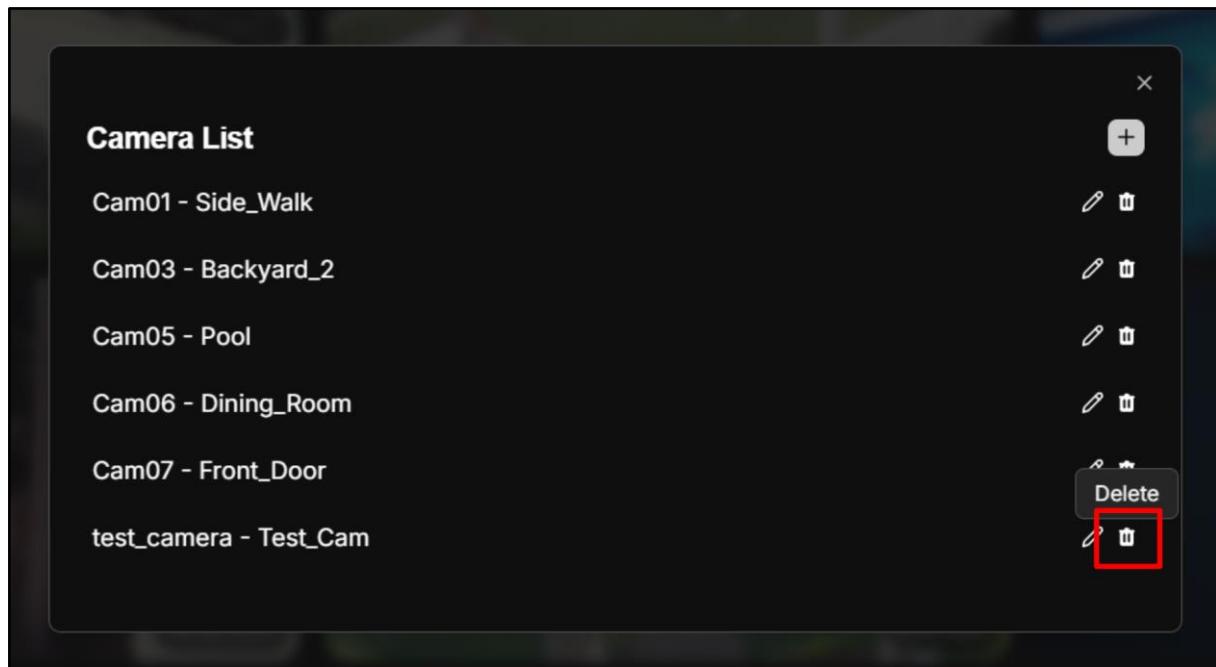


Figure 366. Manage cameras - Delete camera

- Step 3: Confirm delete.

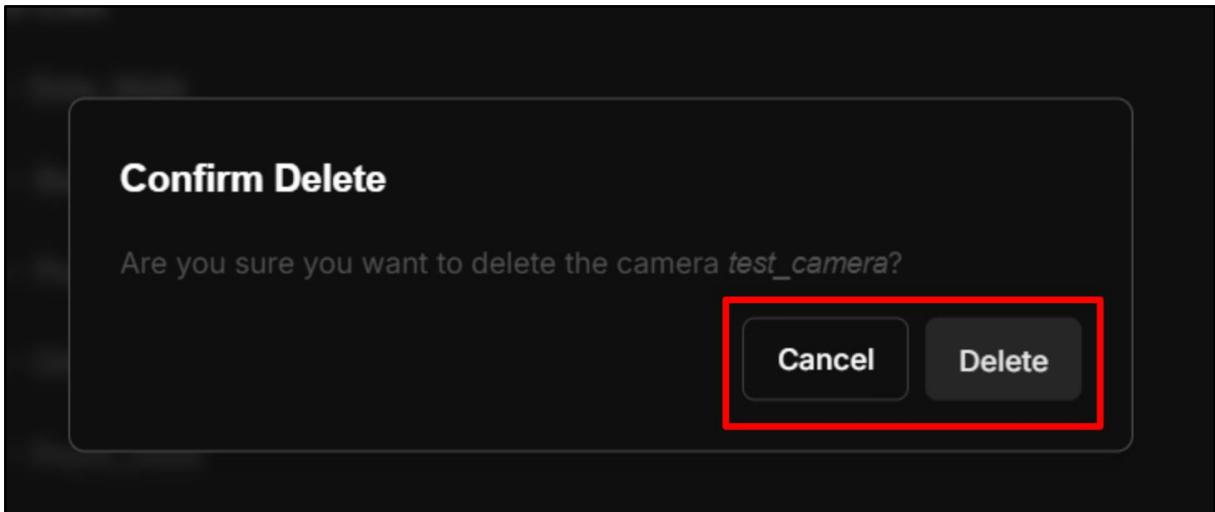


Figure 367. Manage cameras - Delete camera

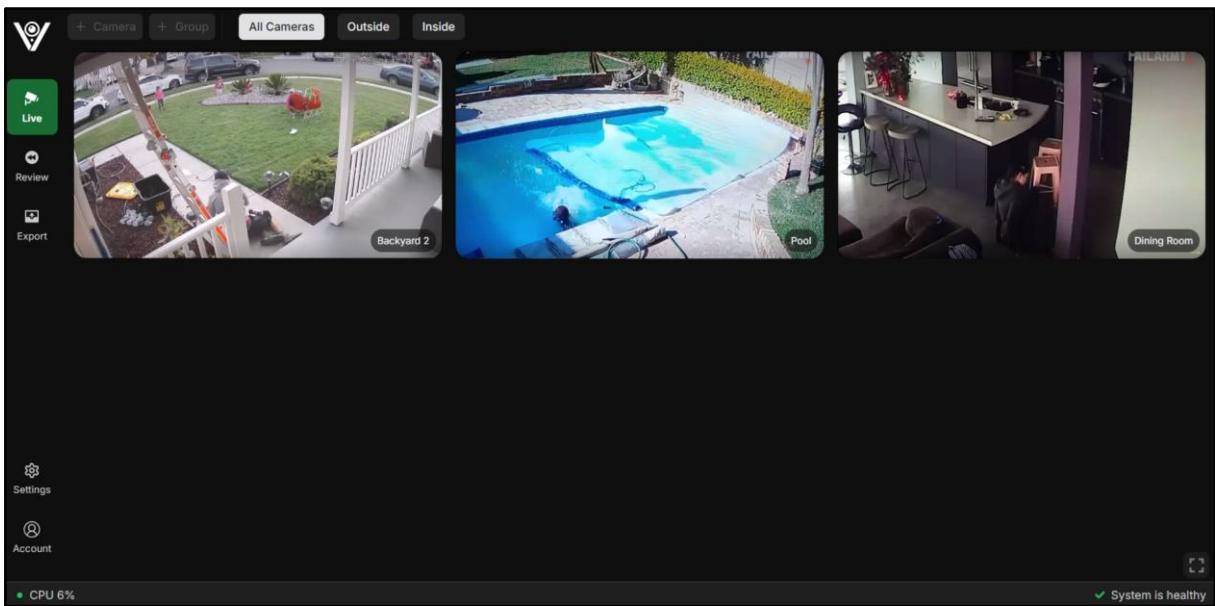


Figure 368. Manage cameras - Delete camera

3.3.2.7 Manage camera groups

3.3.2.7.1 View list of camera groups

There are 2 ways to view list of camera groups:

- View it in the top bar of Live Dashboard page. Note that default camera group is **All Cameras**.

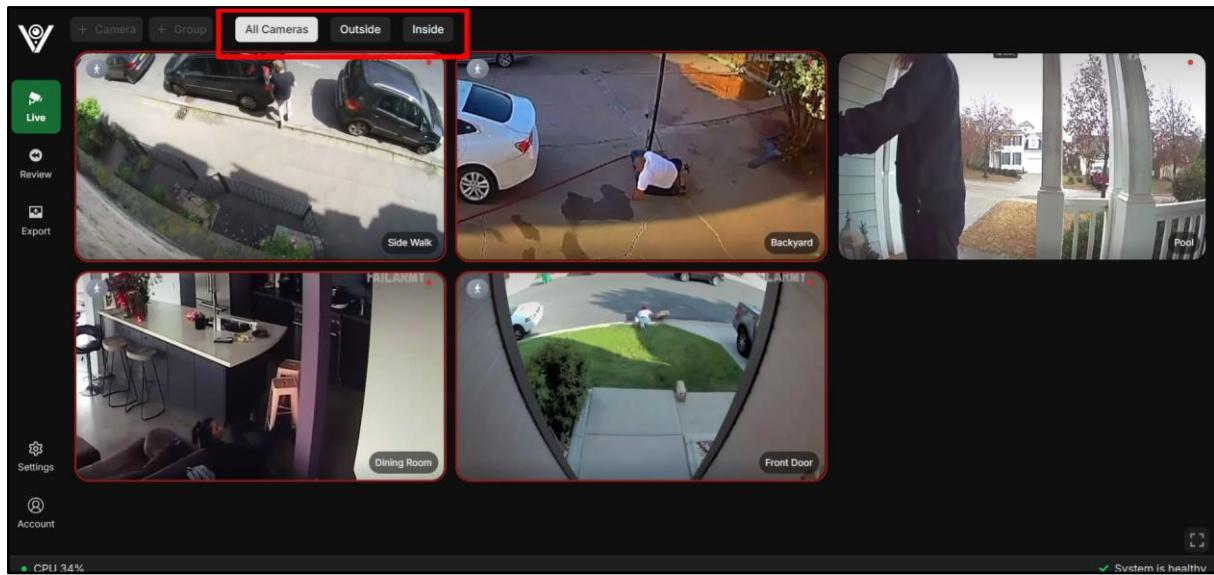


Figure 369. Manage camera groups - View list of camera groups

- View it by clicking on button “+ Group”. After that, the camera group list is shown.

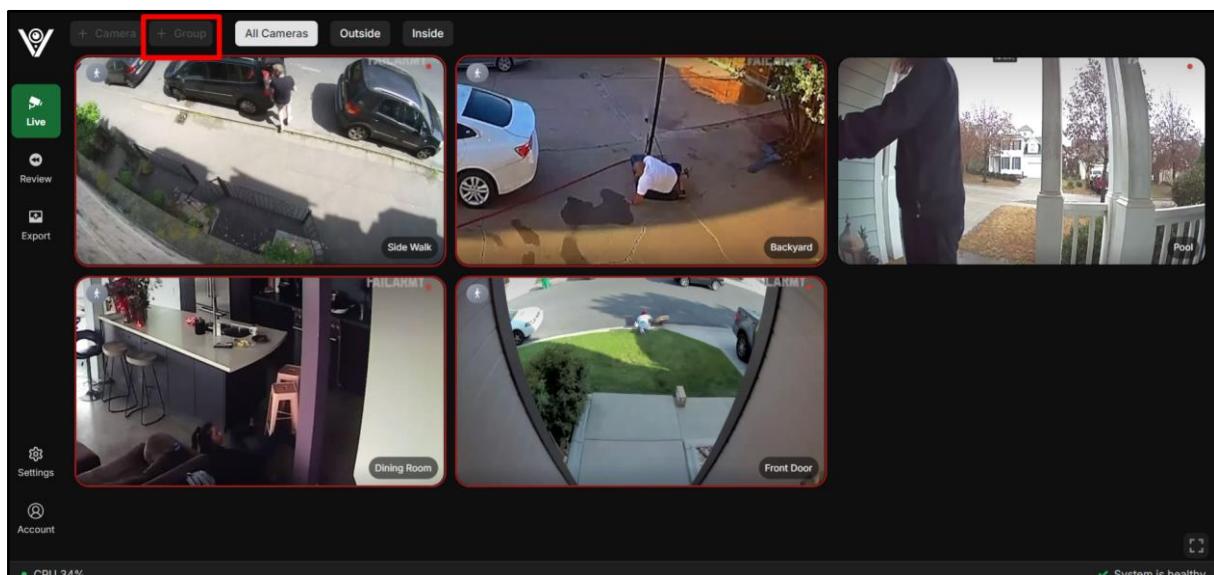


Figure 370. Manage camera groups - View list of camera groups

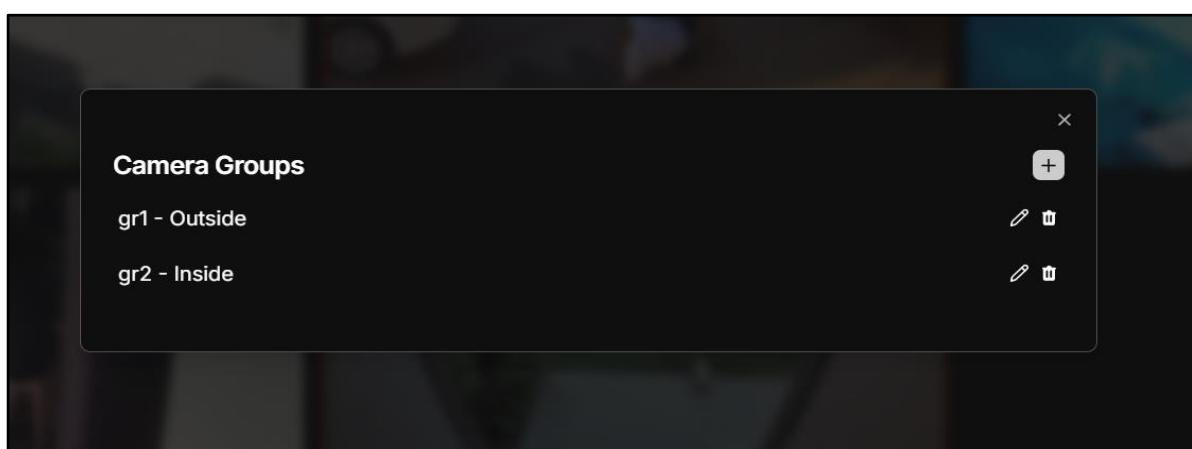


Figure 371. Manage camera groups - View list of camera groups

3.3.2.7.2 Create camera group

- Step 1: On Live Dashboard page, click on button “+ Group”.

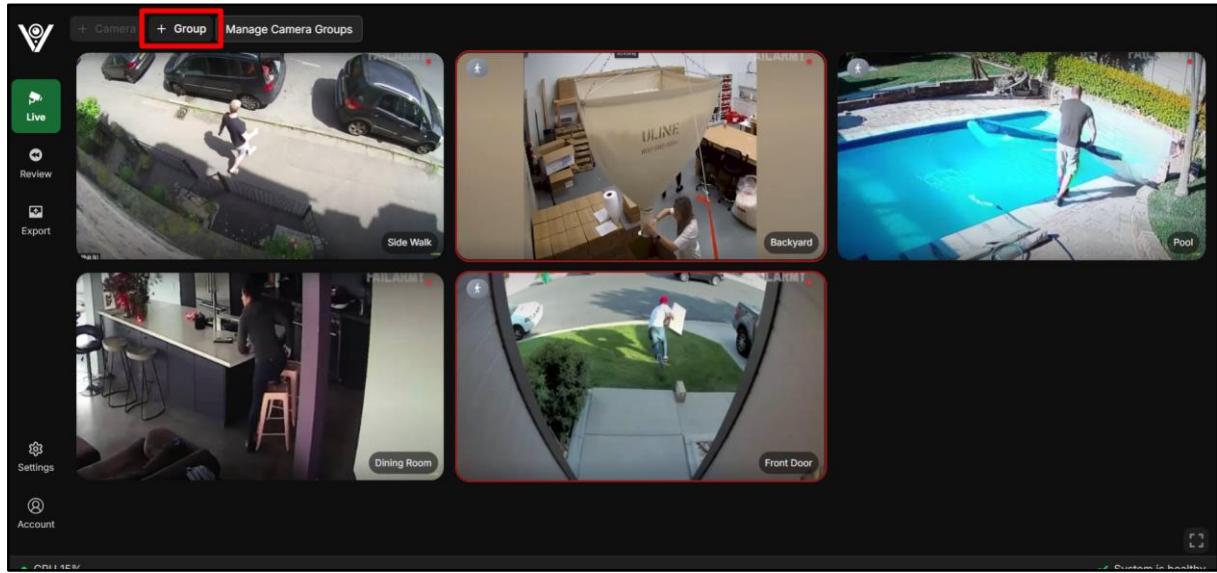


Figure 372. Manage camera groups - Create camera group

- Step 2: Click on button “Add Camera Group”.

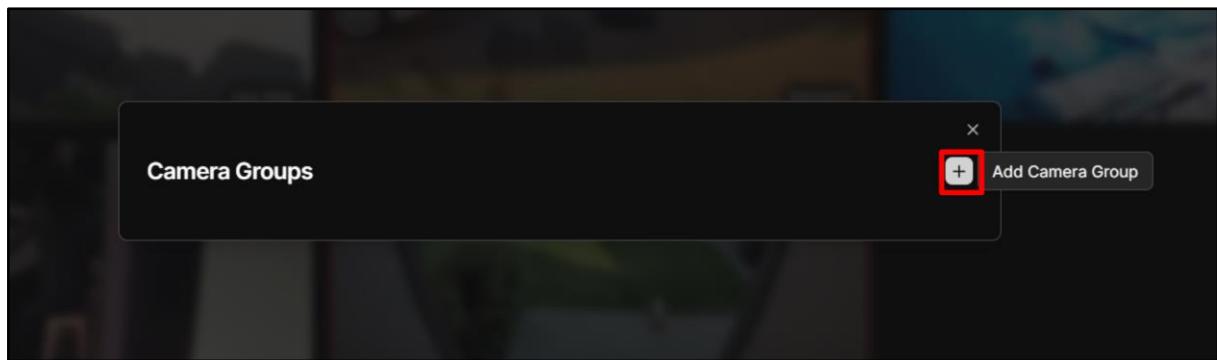


Figure 373. Manage camera groups - Create camera group

- Step 3: Once the form “**Add Camera Group**” appears on the screen, enter group information including group ID, group name, choose cameras to be grouped together.

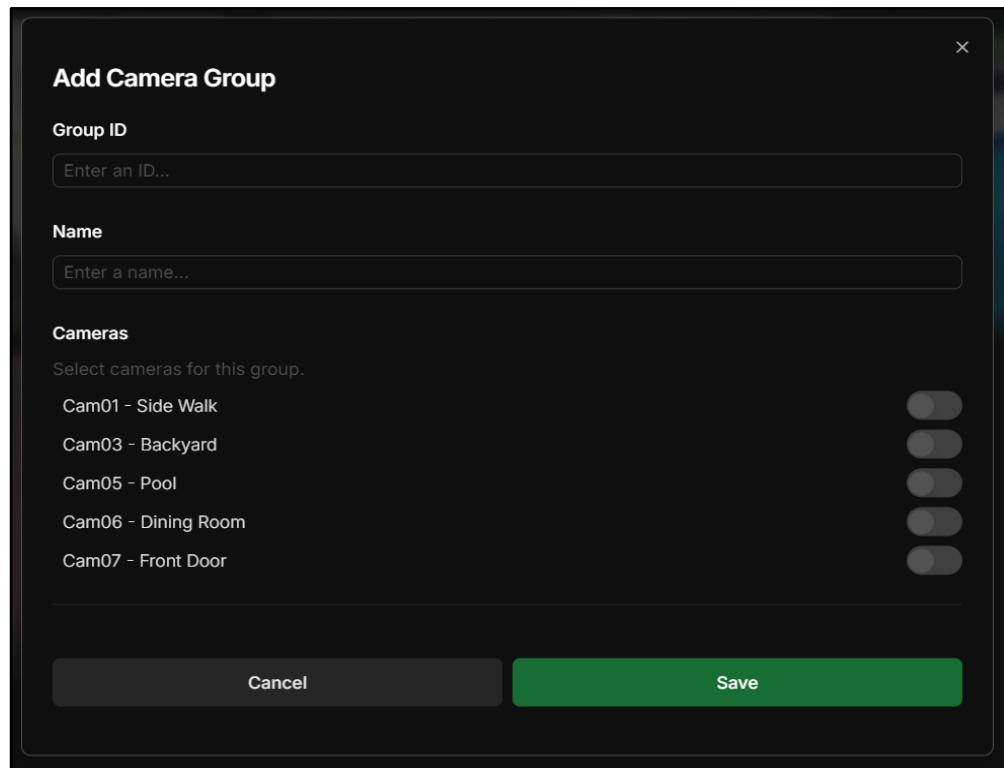


Figure 374. Manage camera groups - Create camera group

- Step 4: Click on button “Save” to save new group. Once the group is saved, the message “Camera group <group name> has been saved” will be displayed.

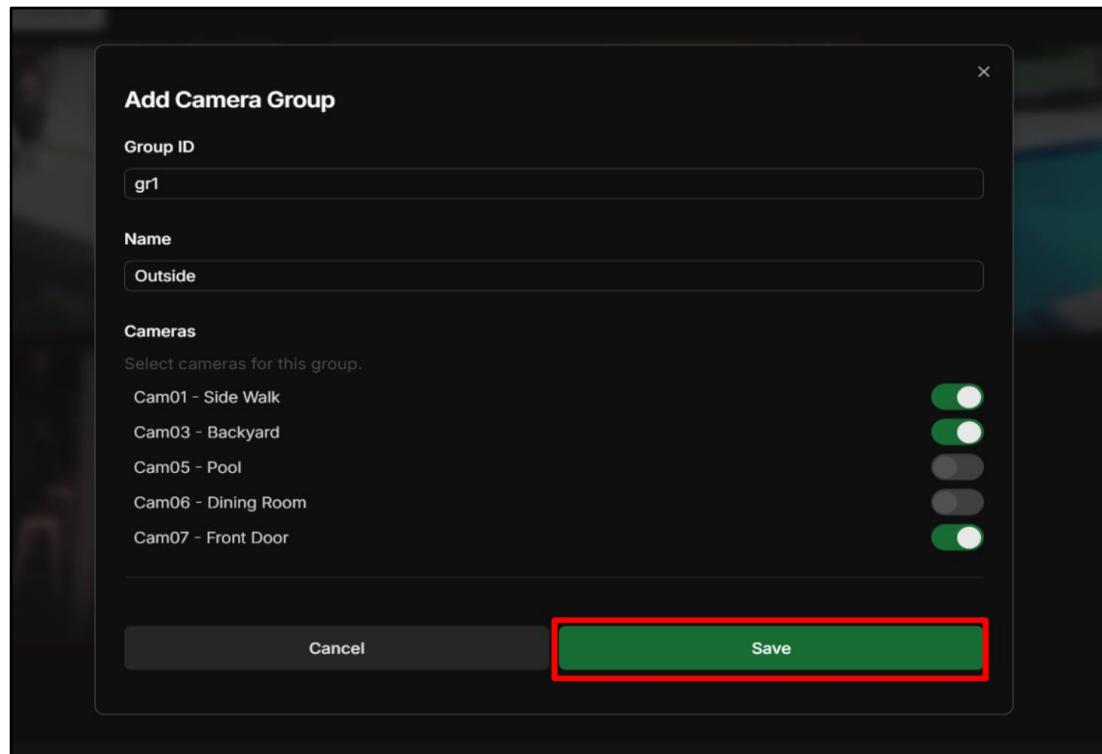


Figure 375. Manage camera groups - Create camera group

You can check this newly created group again by finding its group name on the Live Dashboard page's top bar.

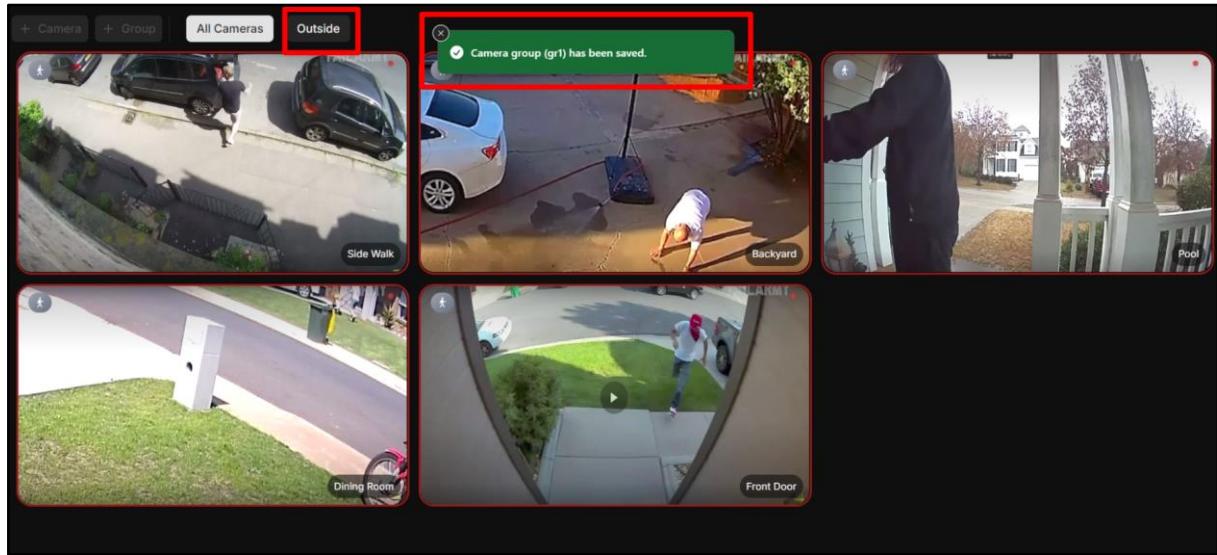


Figure 376. Manage camera groups - Create camera group

3.3.2.7.3 Update camera group

- Step 1: On Live Dashboard page, click on button.

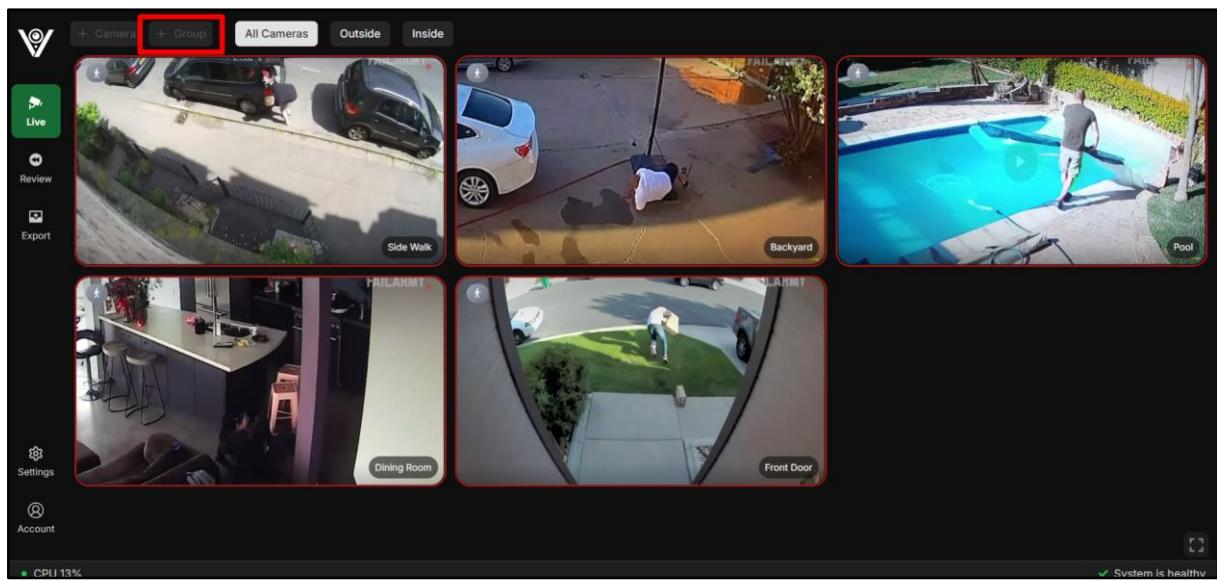


Figure 377. Manage camera groups - Update camera group

- Step 2: Click on button **Edit** corresponding to the camera group needs to be updated.

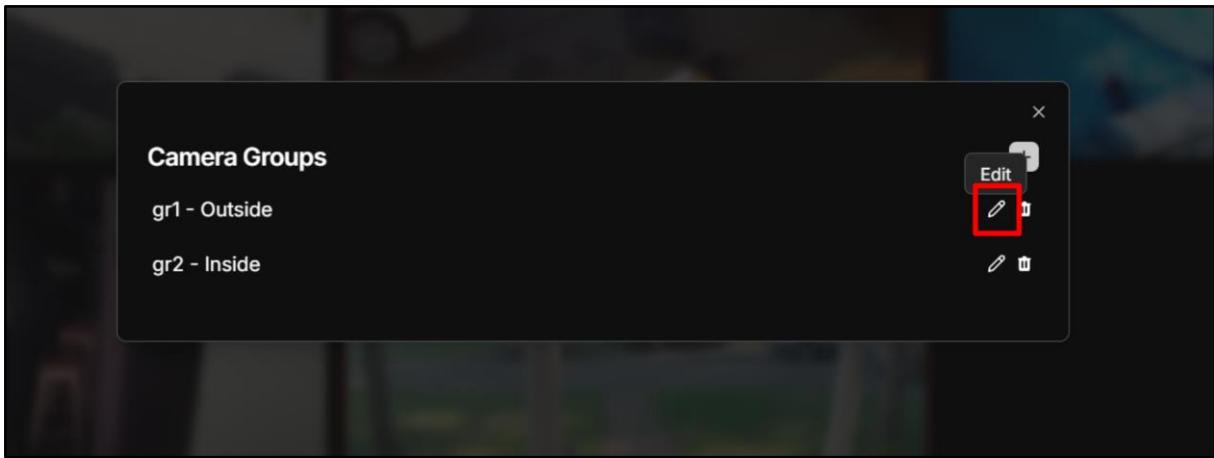


Figure 378. Manage camera groups - Update camera group

Or if you have been viewing that group already, you can simply click on button **Edit** on the bottom-right corner.

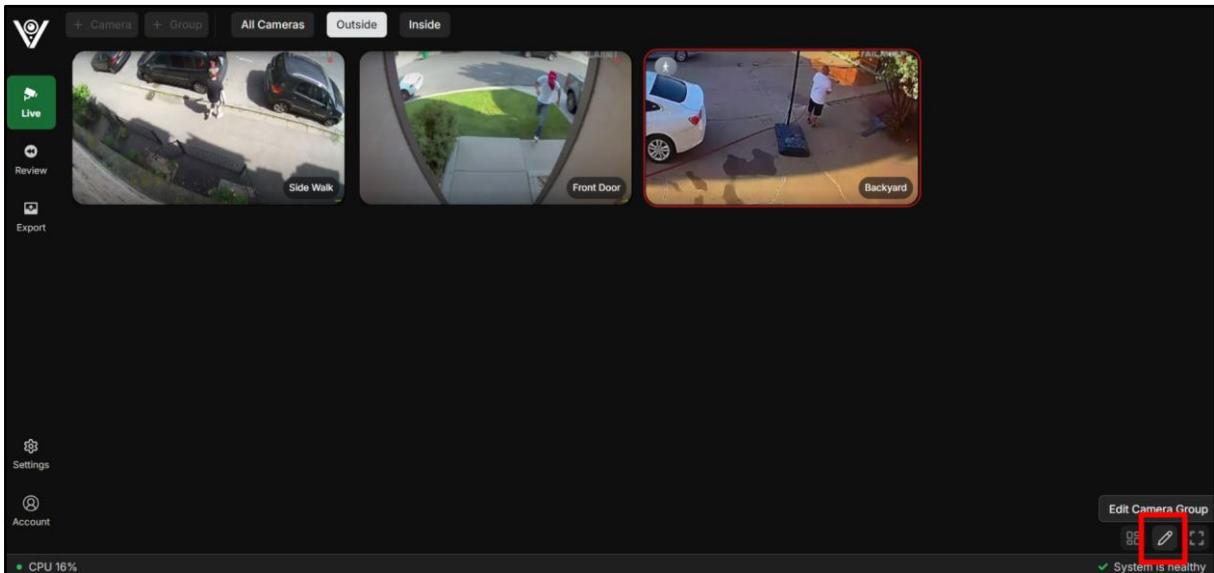


Figure 379. Manage camera groups - Update camera group

- Step 3: Once the form “**Edit Camera Group**” appears on the screen, you can edit group name, add into and/or delete some cameras from the group.
- Step 4: Click on button “Save” to save new group. Once the group is saved, the message “Camera group <group name> has been saved” will be displayed.

You can check this group again by clicking its group name on the Live Dashboard page's top bar and check the cameras inside it.

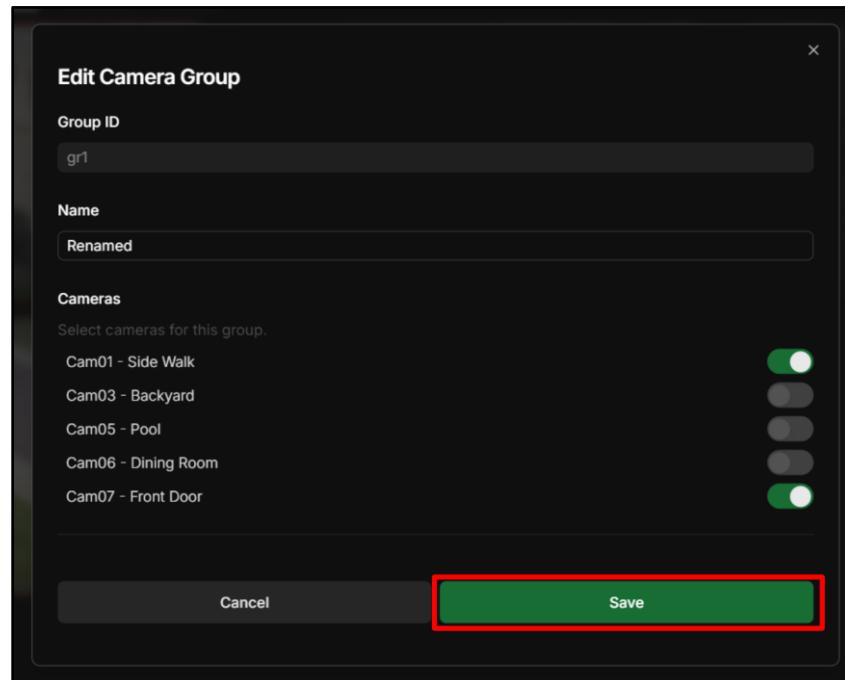


Figure 380. Manage camera groups - Update camera group

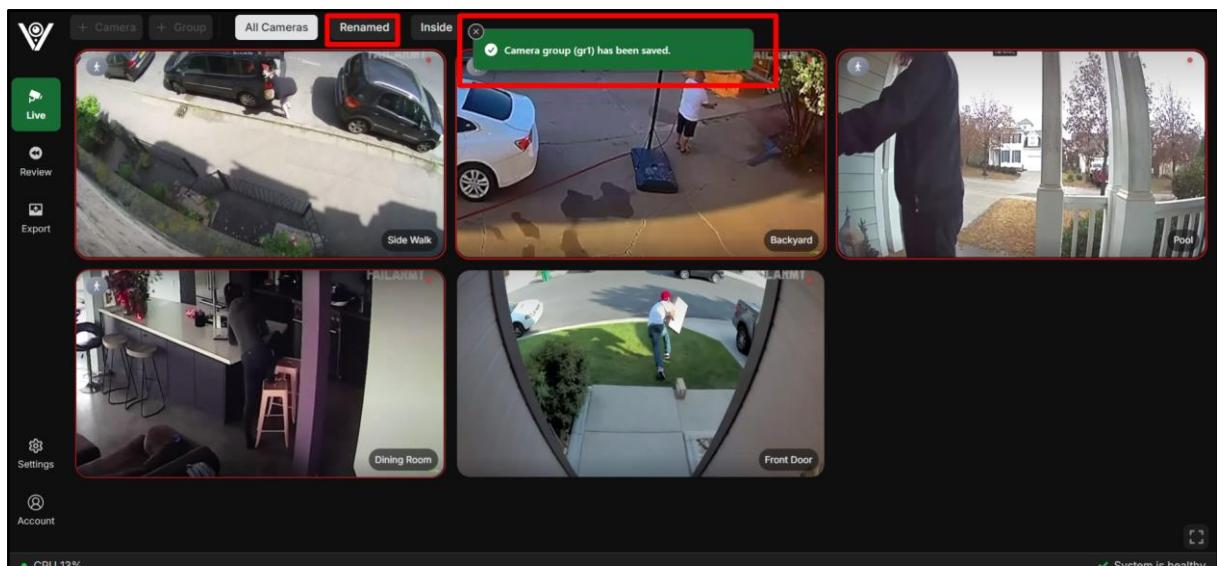


Figure 381. Manage camera groups - Update camera group

3.3.2.7.4 Delete camera group

- Step 1: On Live Dashboard page, click on button.

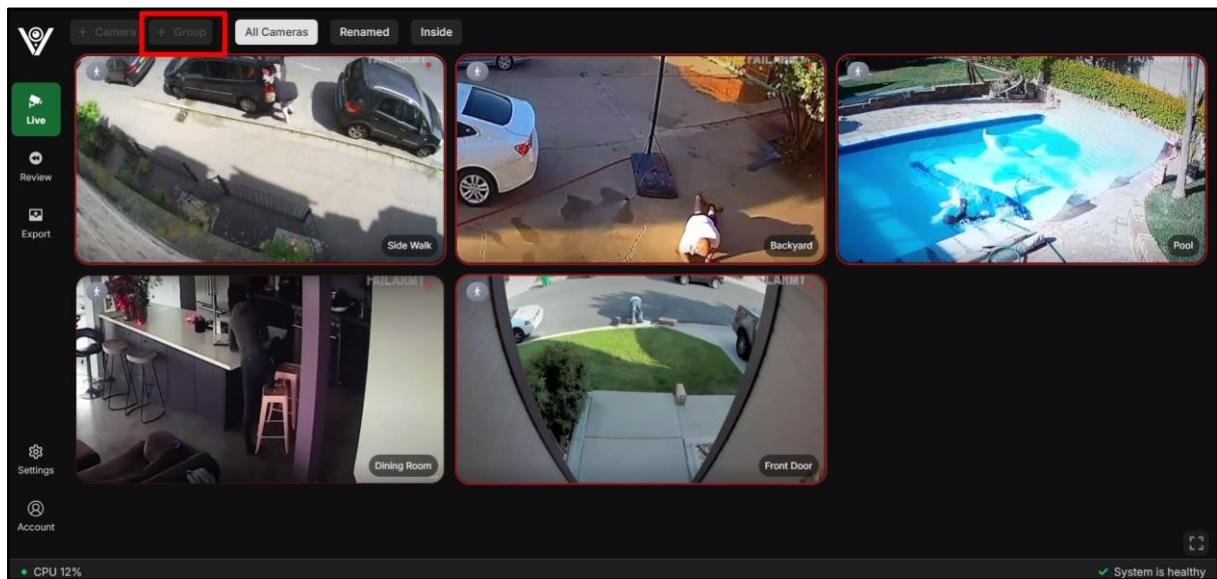


Figure 382. Manage camera groups - Delete camera group

- Step 2: Click on button corresponding to the camera group needs to be deleted.

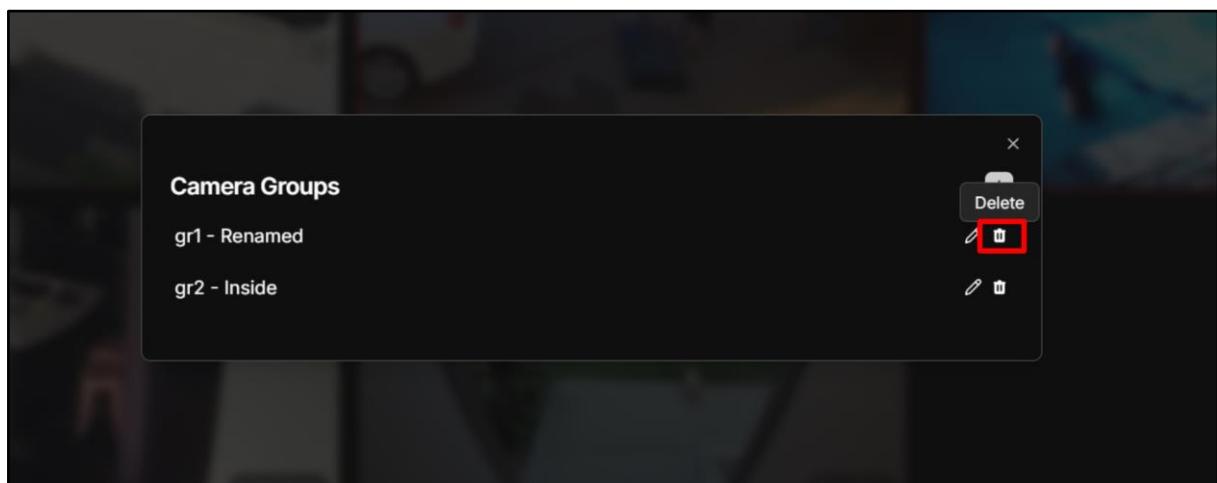


Figure 383. Manage camera groups - Delete camera group

- Step 3: Confirm delete.

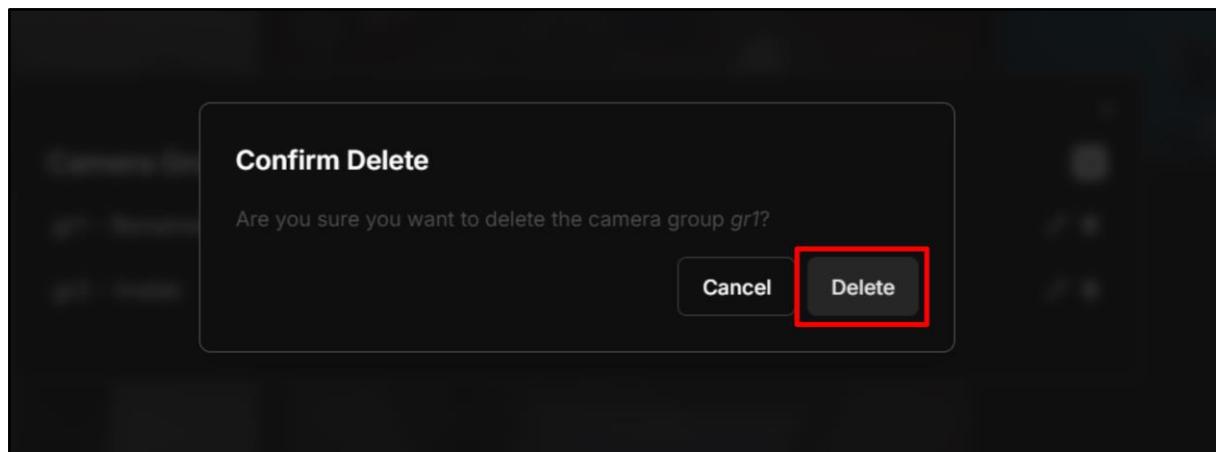


Figure 384. Manage camera groups - Delete camera group

As the result, the camera group is deleted.

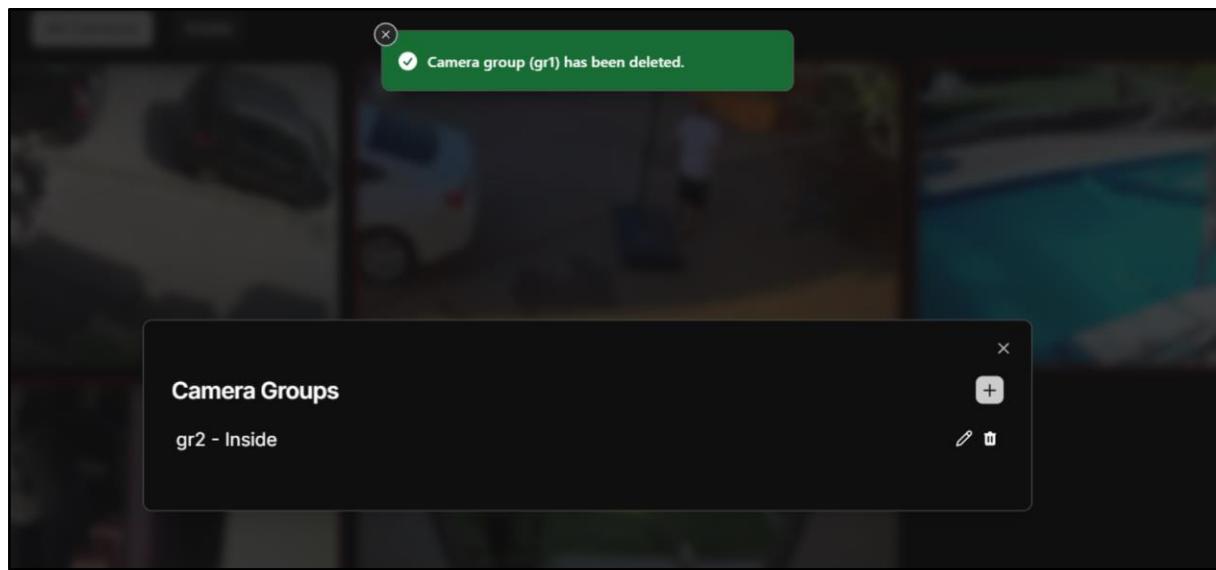


Figure 385. Manage camera groups - Delete camera group

3.3.2.8 Edit camera group layout

- Step 1: If you are viewing All Cameras, switch to the group you want to edit layout. Then, you can simply click on button “**Edit Layout**” on the bottom-right corner.

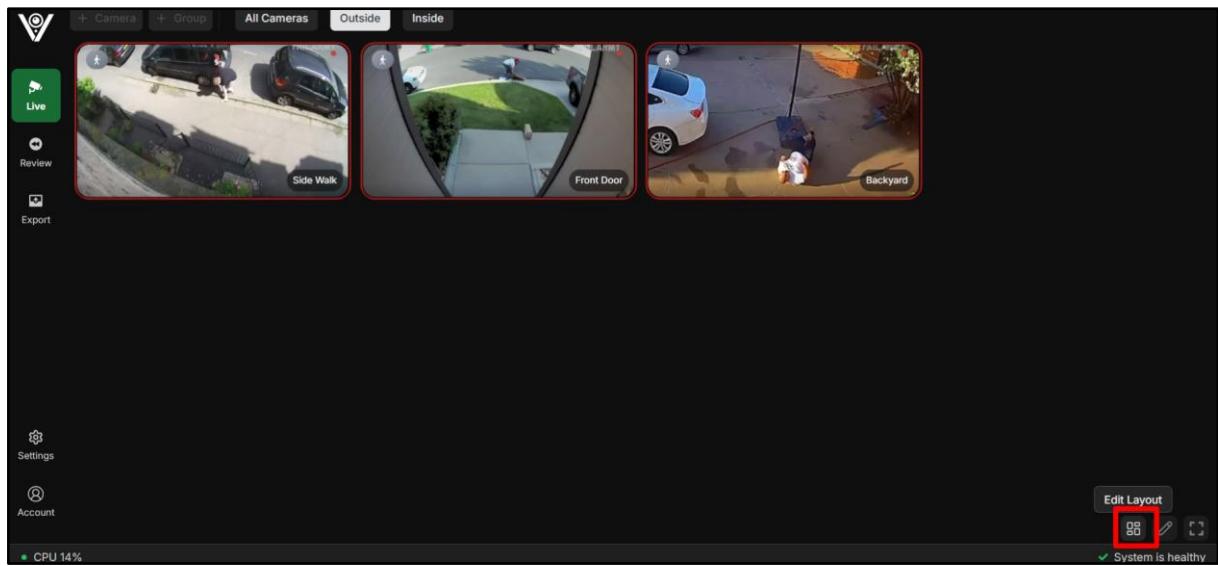


Figure 386. Edit camera group layout

- Step 2: Drag and drop each camera in the group to create the desired new layout. Finally, click button at the bottom-right corner to apply new layout.

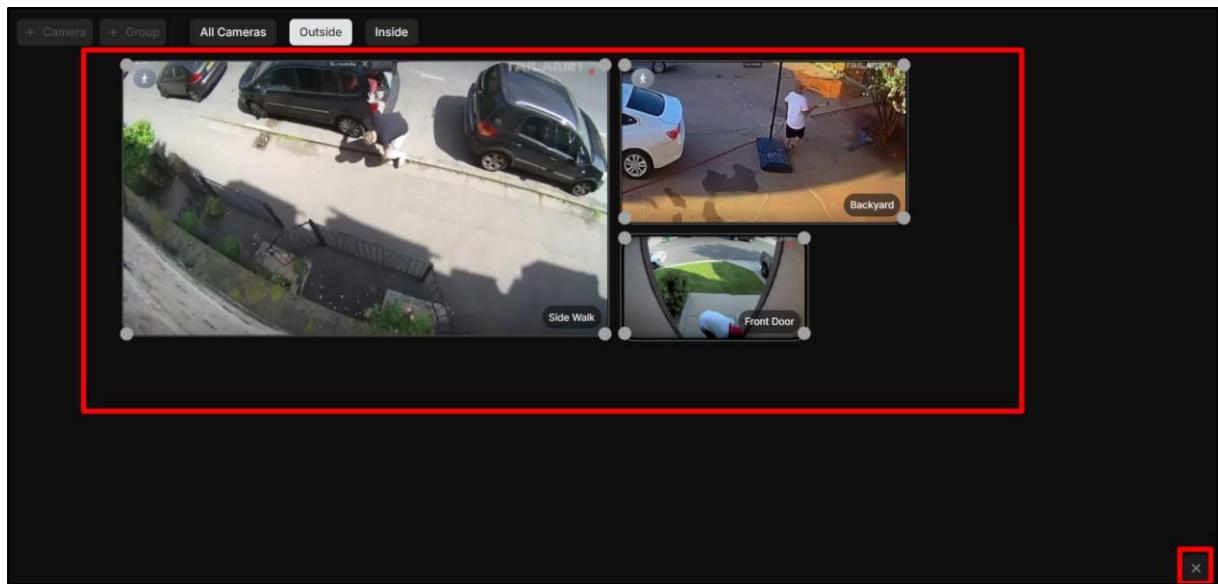


Figure 387. Edit camera group layout

3.3.2.9 View in full screen

Case 1: On the Live Dashboard page, click on button at the bottom-right corner.

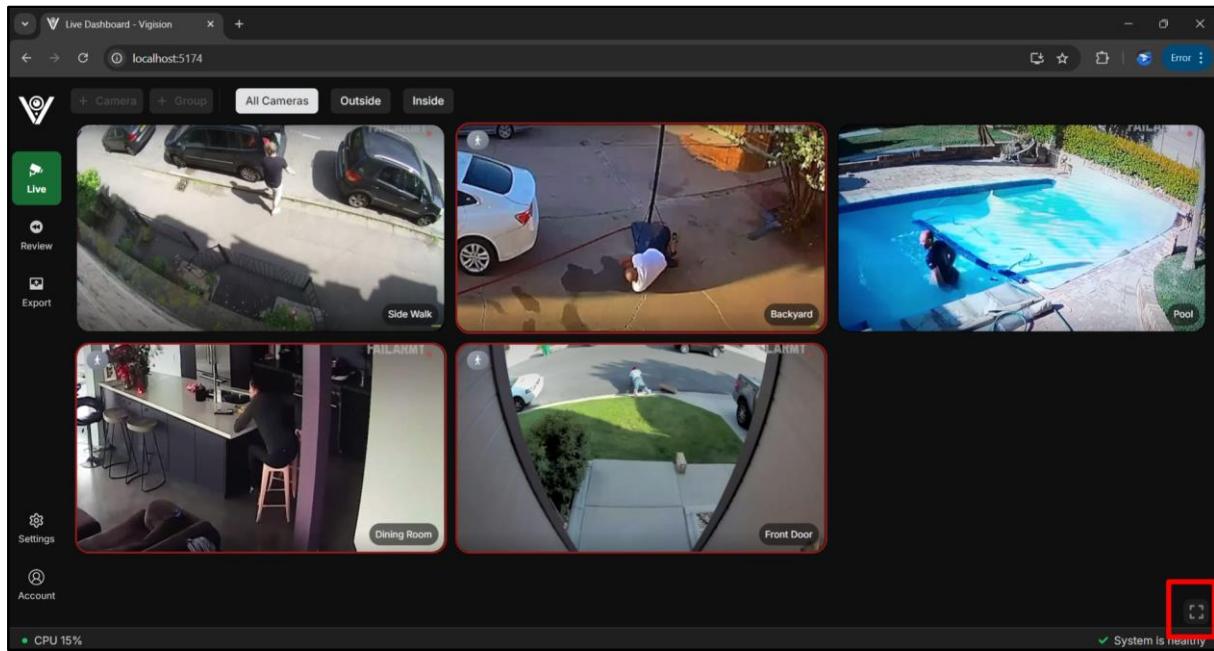


Figure 388. View in full screen

The full screen mode is on. Now you can view in full screen.

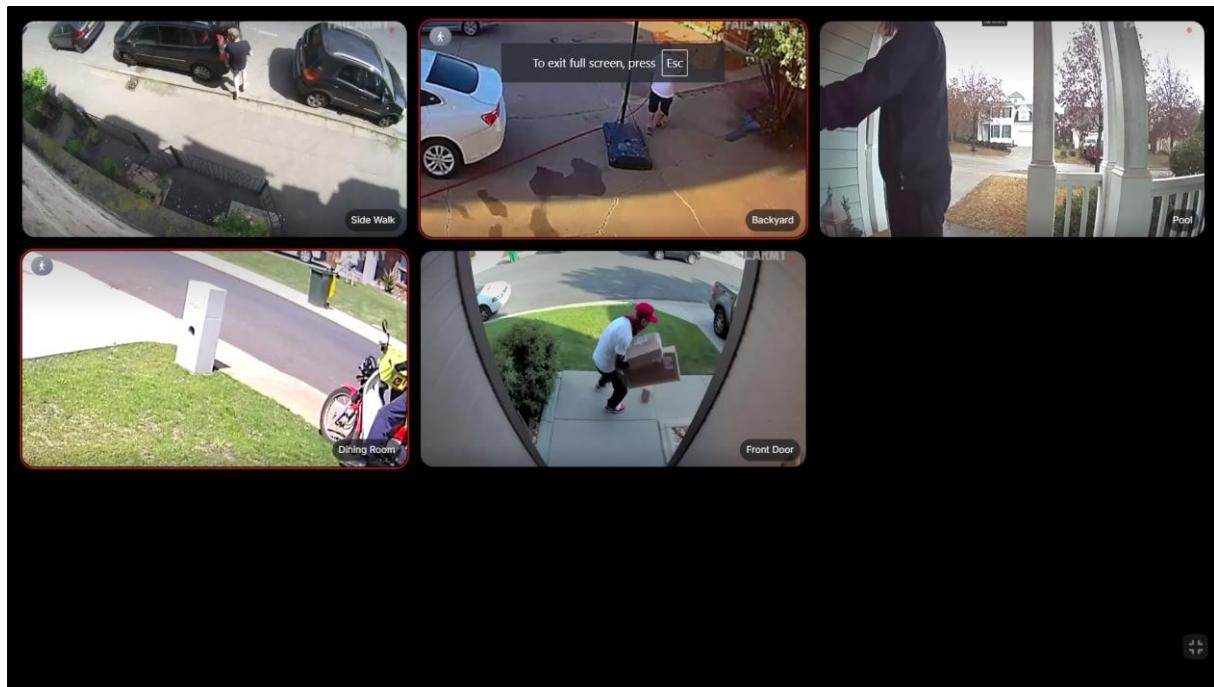


Figure 389. View in full screen

Case 2: Or if you are currently viewing a camera group, it also has button at the bottom-right corner.

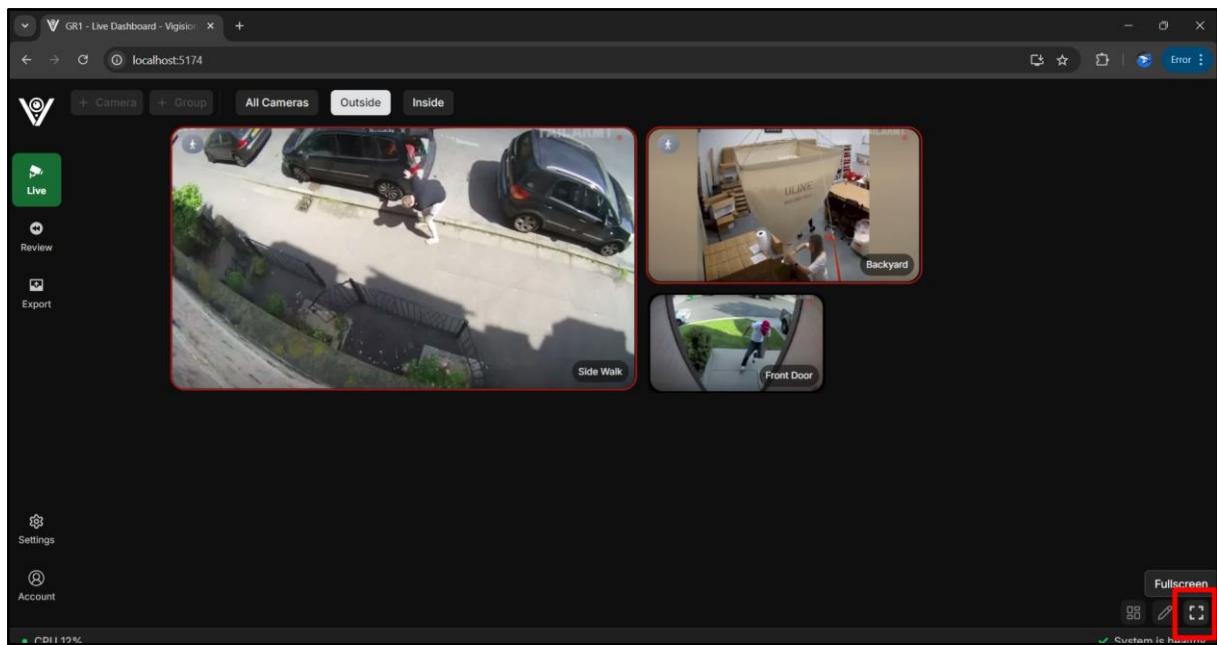


Figure 390. View in full screen

The full screen mode is on. Now you can view in full screen.

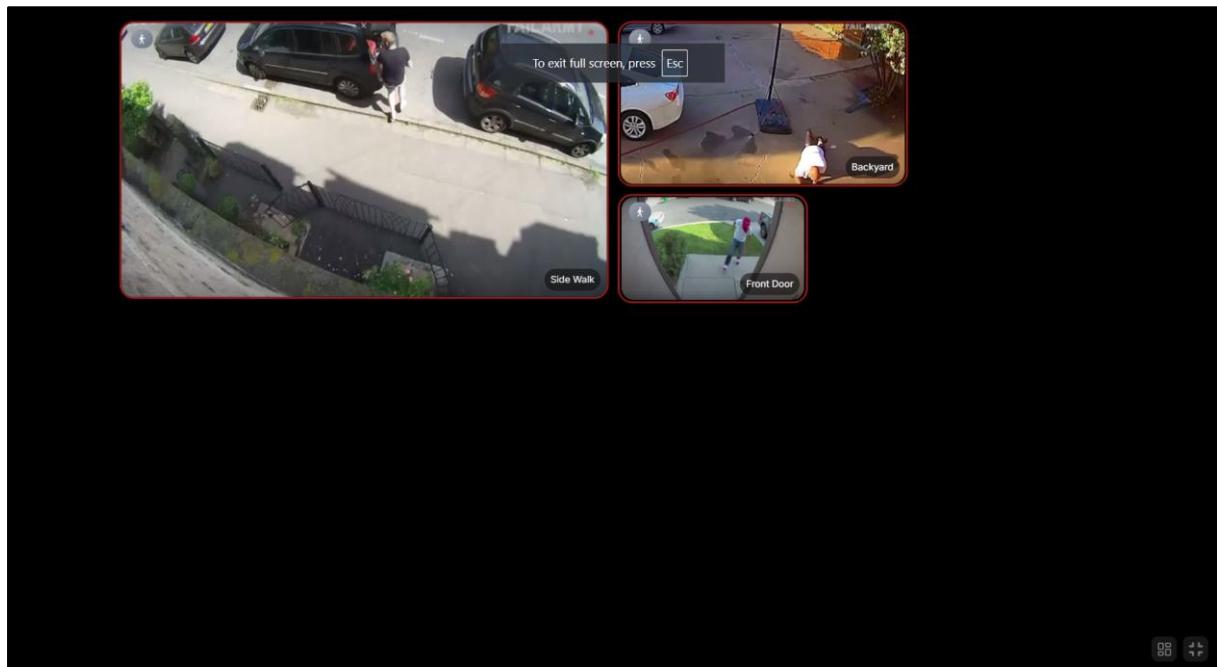


Figure 391. View in full screen

Case 3: You are viewing a certain camera in details. You can find the button at the top-right corner of the screen. Click on it and view full screen.

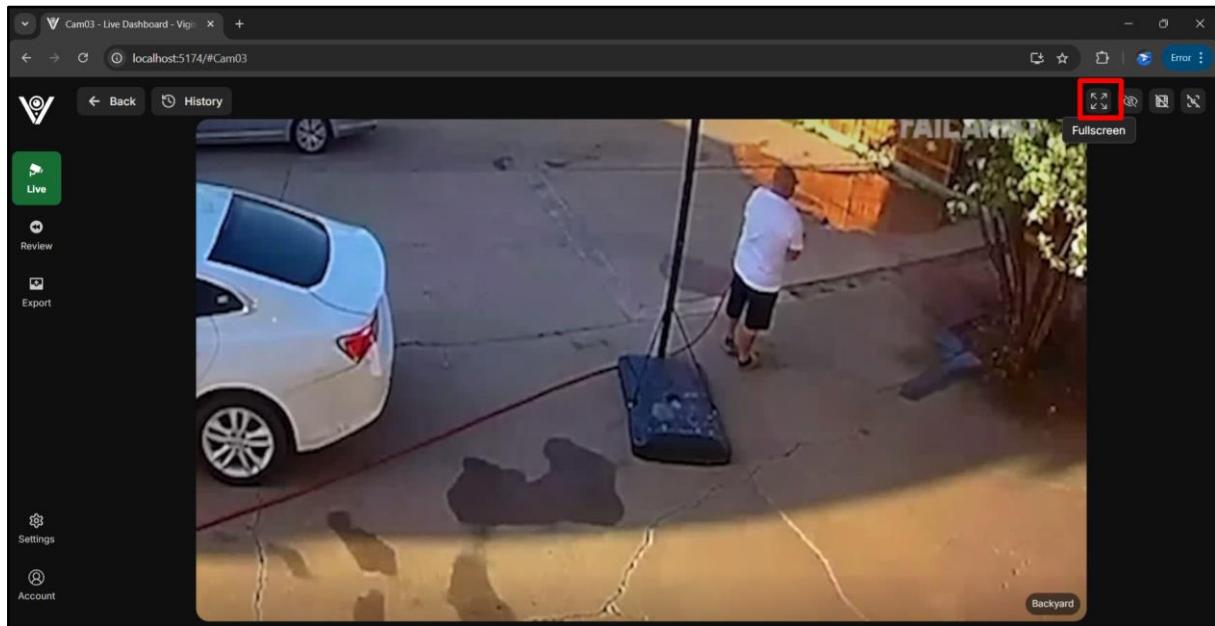


Figure 392. View in full screen

To exit full screen, you can click on button at the top-right corner of the screen.



Figure 393. View in full screen

3.3.2.10 View detail of live camera

On the Live Dashboard page, click on the thumbnail of the live camera you want to view detail.

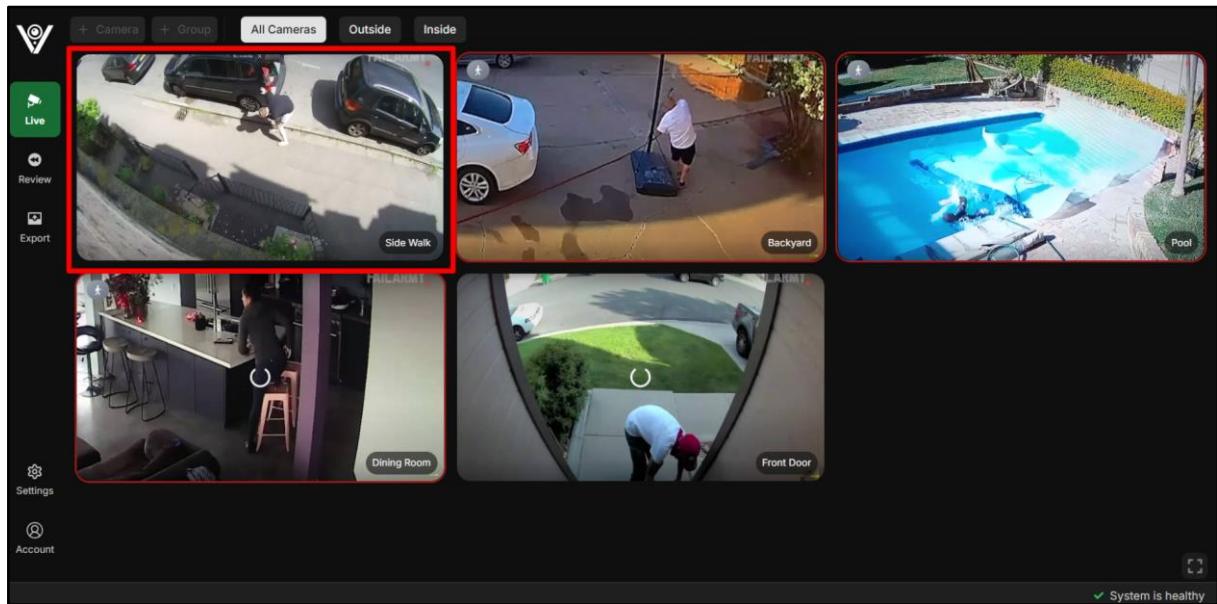


Figure 394. View detail of live camera

Detail of a live camera is shown below.



Figure 395. View detail of live camera

3.3.2.11 Enable/disable detection

Case 1: Enable detect:

On the page of camera detail, click on button “**Enable Detect**” on the top-right corner.



Figure 396. Enable/disable detection

The system then sends a message stating that detection has been successfully enabled.

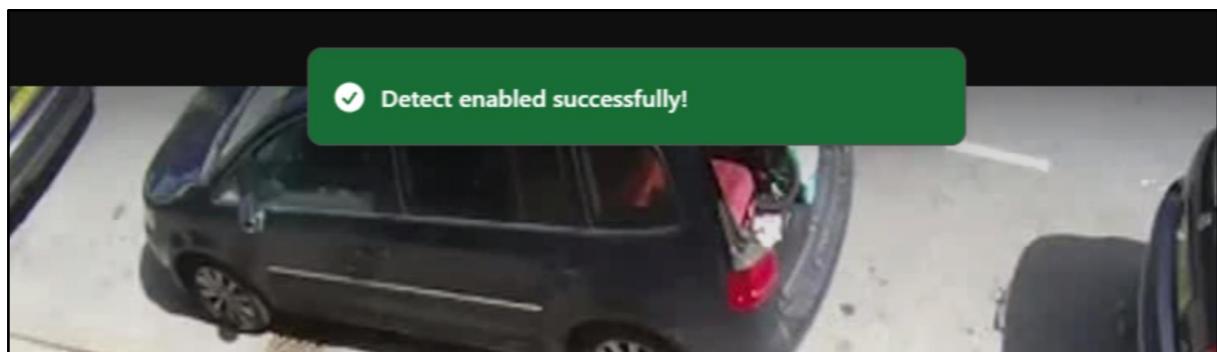


Figure 397. Enable/disable detection

Case 2: Disable detect:

On the page of camera detail, click on button “**Disable Detect**” on the top-right corner.



Figure 398. Enable/disable detection

The system then sends a message stating that detection has been successfully disabled.

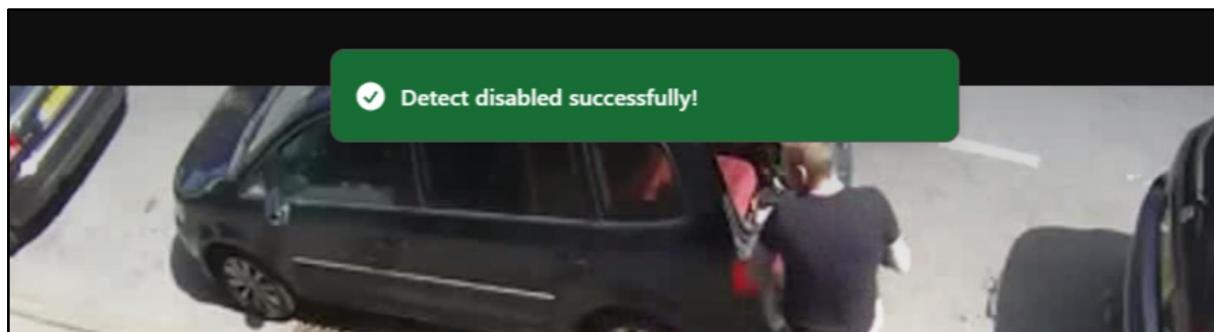


Figure 399. Enable/disable detection

3.3.2.12 Enable/disable recording

Case 1: Enable recording:

On the page of camera detail, click on button “**Enable Recording**” on the top-right corner.



Figure 400. Enable/disable recording

If you have not enable record in camera configuration, the system then sends a message stating action failed.

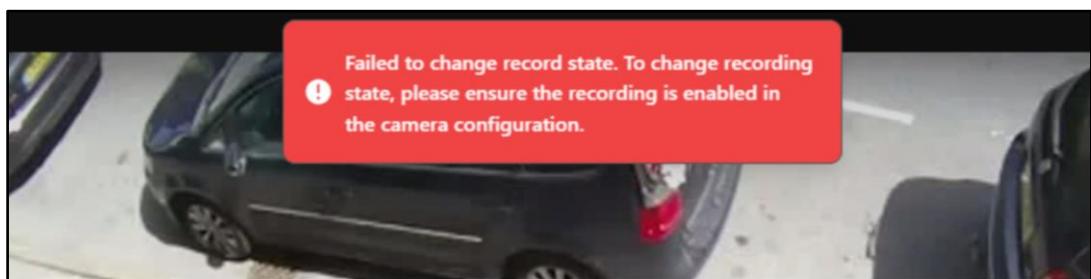


Figure 401. Enable/disable recording

So, you need to enable record in the camera configuration first.

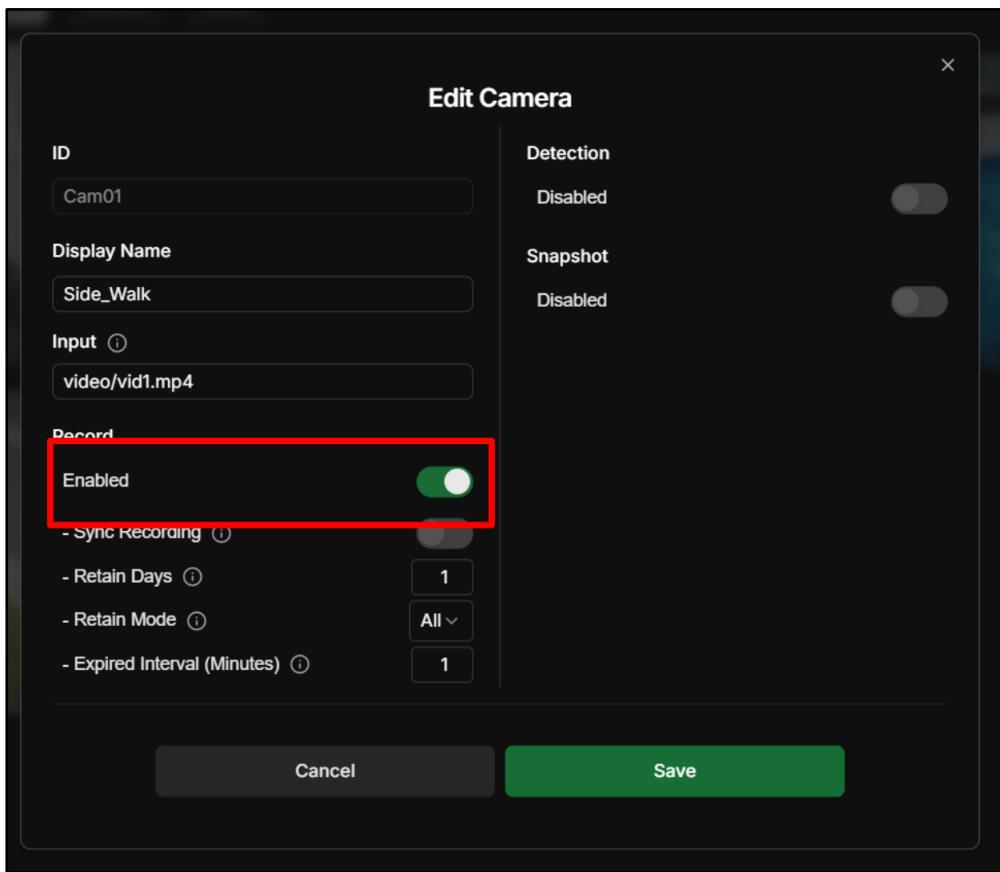


Figure 402. Enable/disable recording

Once, camera configuration has been updated. Click on button “**Enable Recording**” again. Then, system send a message state that recording enabled successfully.

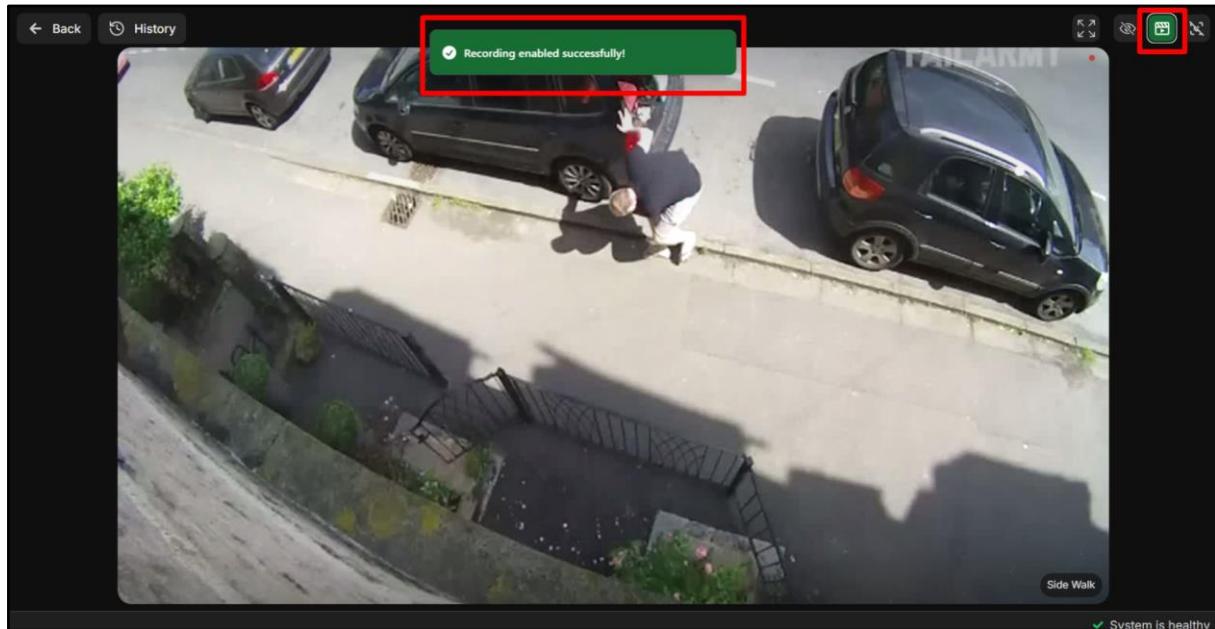


Figure 403. Enable/disable recording

Case 2: Disable recording

On the page of camera detail, click on button “**Disable Record**” on the top-right corner.



Figure 404. Enable/disable recording

The system then sends a message stating that recording has been successfully disabled.

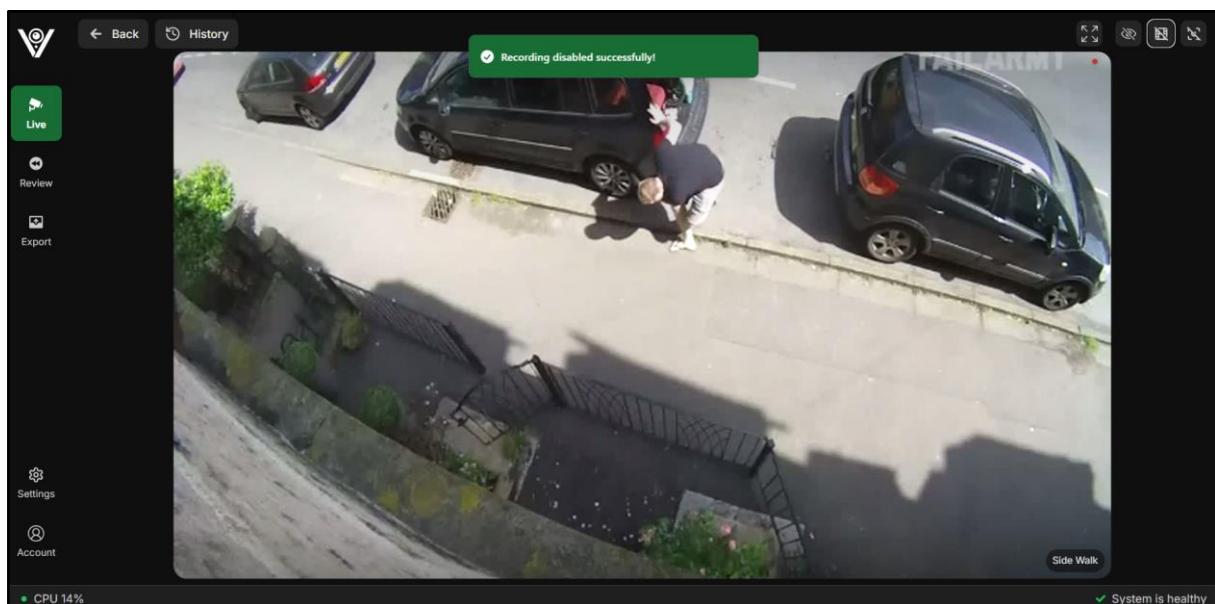


Figure 405. Enable/disable recording

3.3.2.13 Enable/disable snapshots

Case 1: Enable snapshots:

On the page of camera detail, click on button “**Enable Snapshots**” on the top-right corner.



Figure 406. Enable/disable snapshots

The system then sends a message stating that snapshot has been successfully enabled.

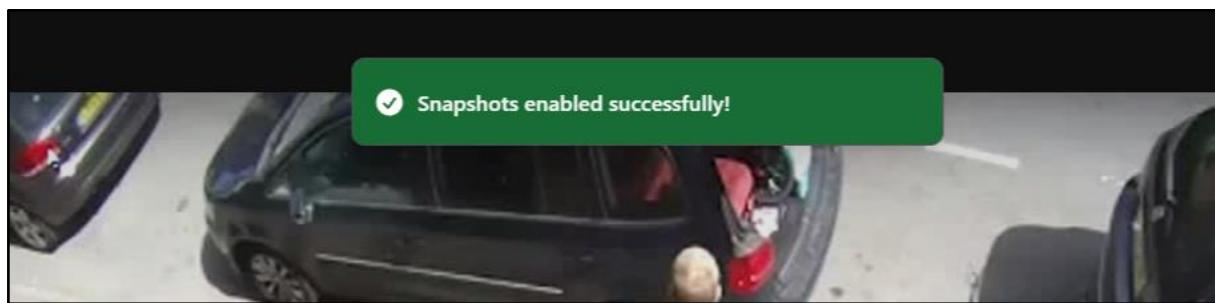


Figure 407. Enable/disable snapshots

Case 2: Disable snapshots:

On the page of camera detail, click on button “**Disable Snapshots**” on the top-right corner.

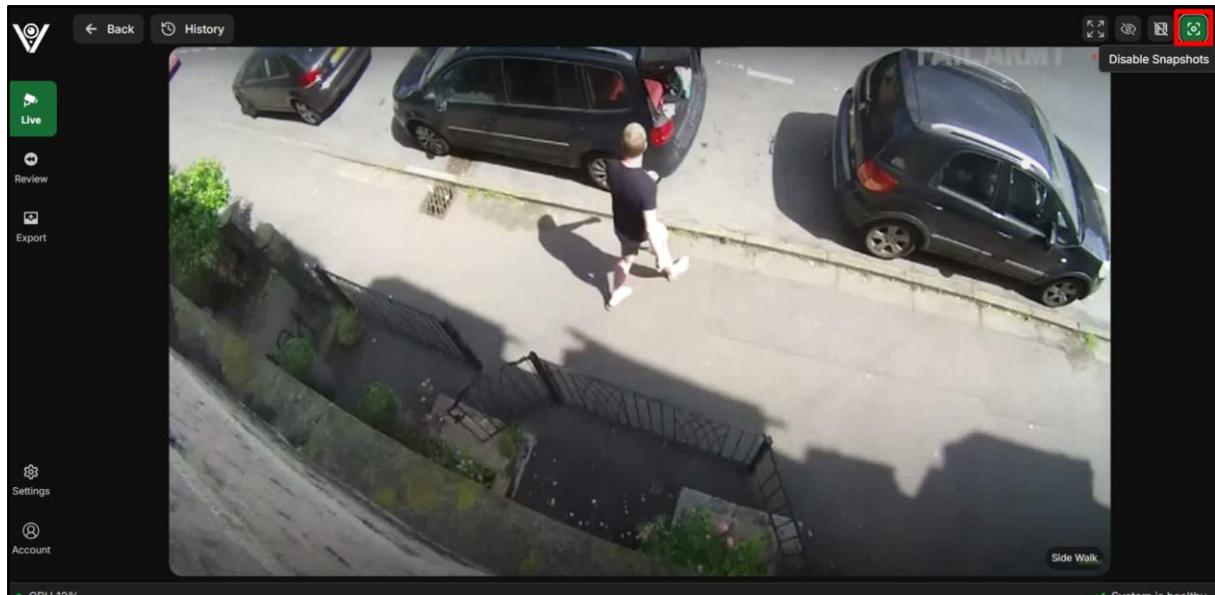


Figure 408. Enable/disable snapshots

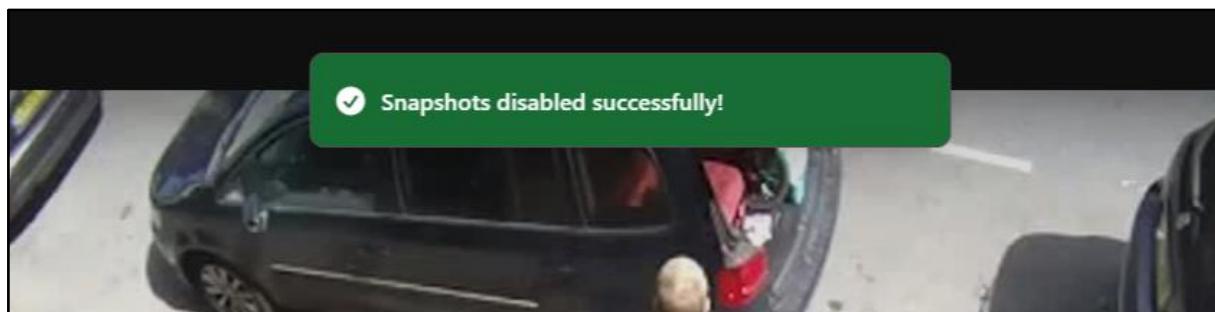


Figure 409. Enable/disable snapshots

The system then sends a message stating that detection has been successfully disabled.

3.3.2.14 View defined masks and zones

- Step 1: Click on button “Settings”.

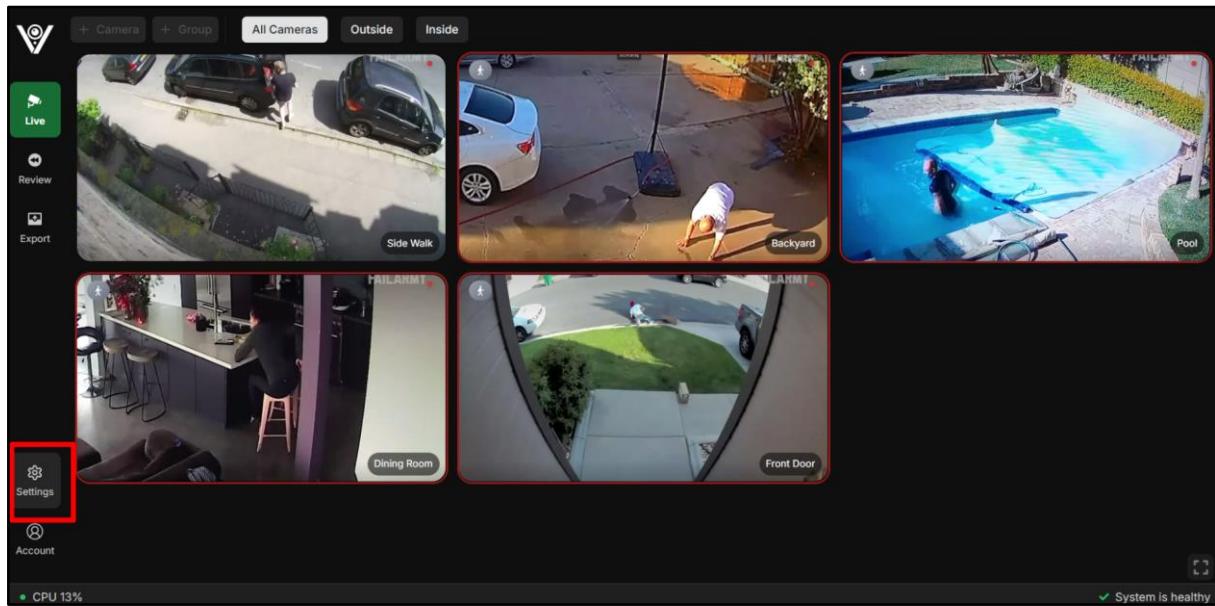


Figure 410. View defined masks and zones

- Step 2: Click on “Settings”.

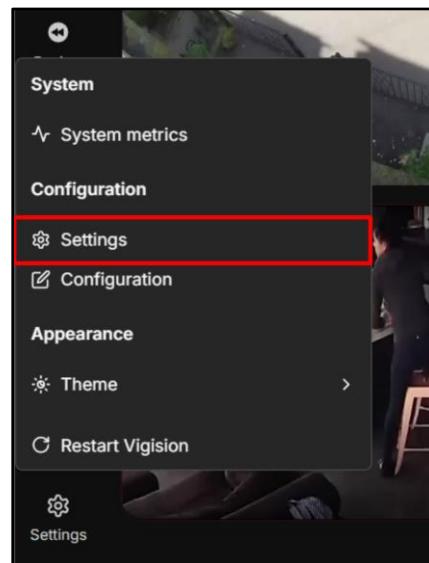


Figure 411. View defined masks and zones

- Step 3: View defined masks and zones in the right pannel “Masks/ Zones”

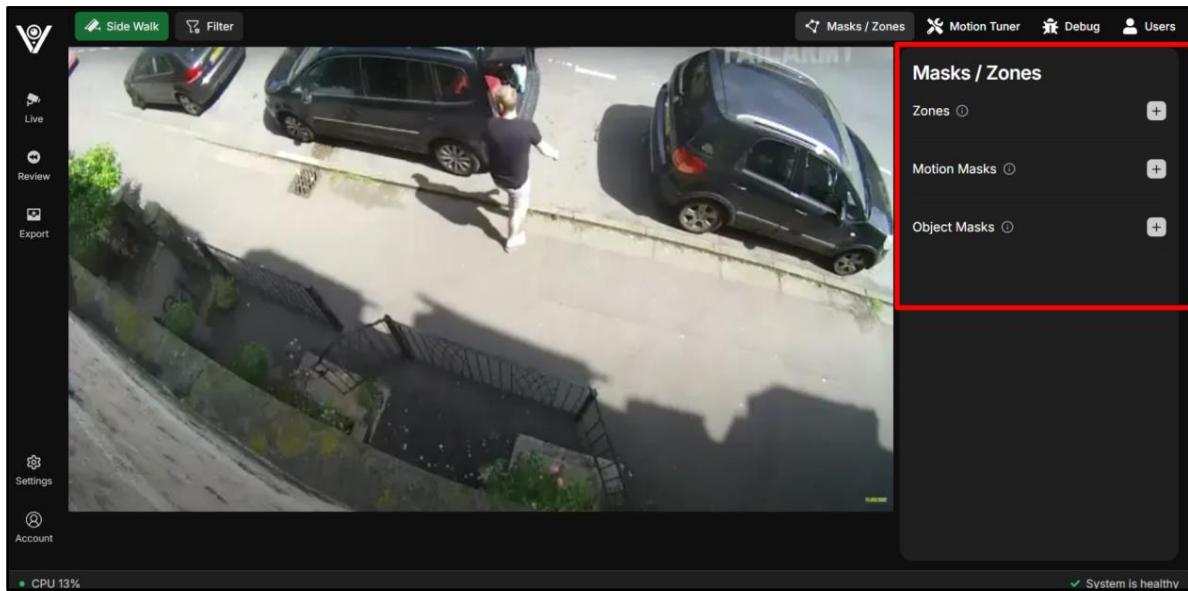


Figure 412. View defined masks and zones

In the image above, there are no defined masks/zones so have a look at another camera.

To view masks/zones of other camera you can click on the green button at the top-left corner, a list of available camera is shown. Then, just simply select another camera. This is shown as the image below:

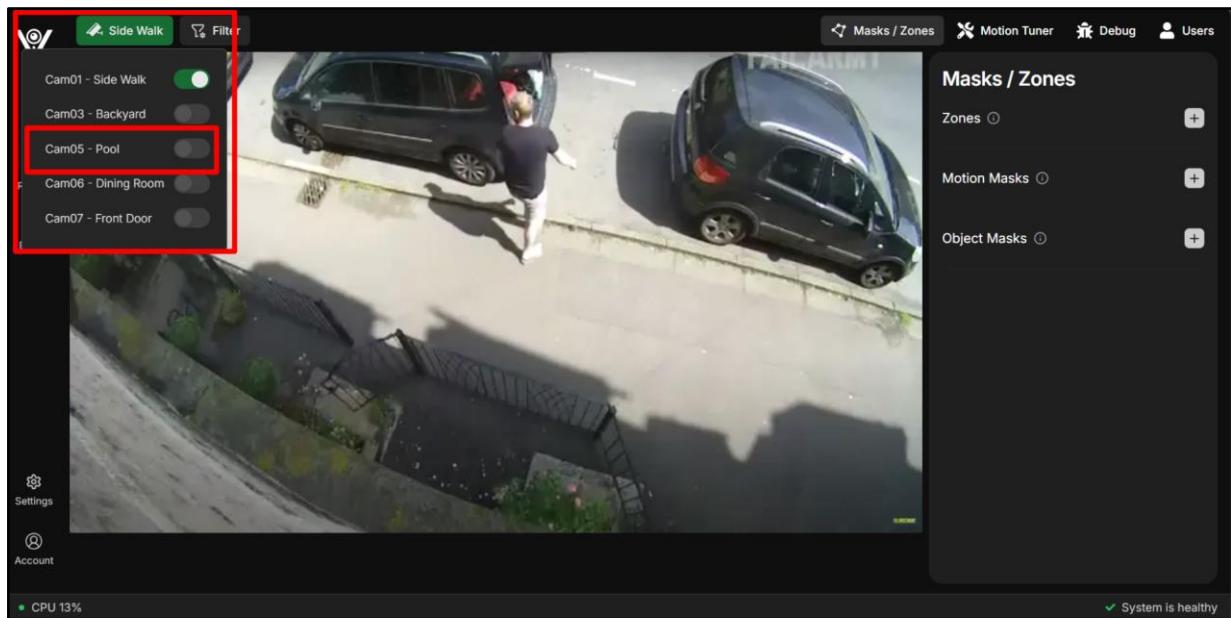


Figure 413. View defined masks and zones

Now you can see defined masks/zones in the left panel.

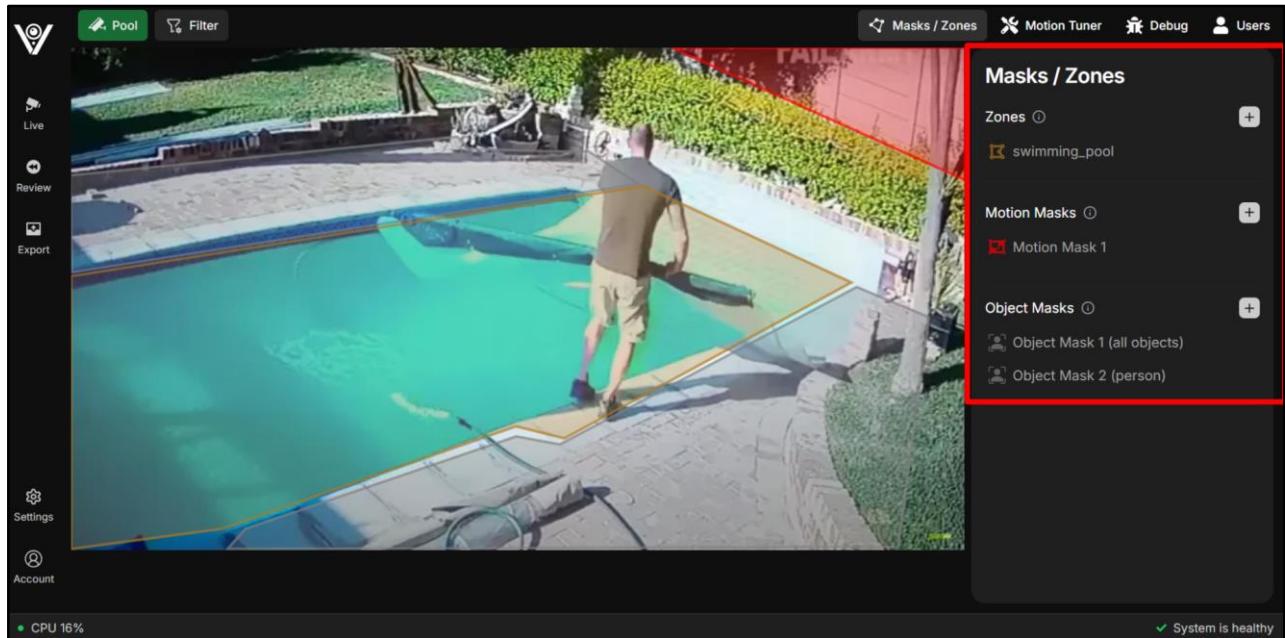


Figure 414. View defined masks and zones

3.3.2.15 Manage motion masks

3.3.2.15.1 Create motion mask

- Step 1: Click green button to change to the camera you want to create motion mask for.

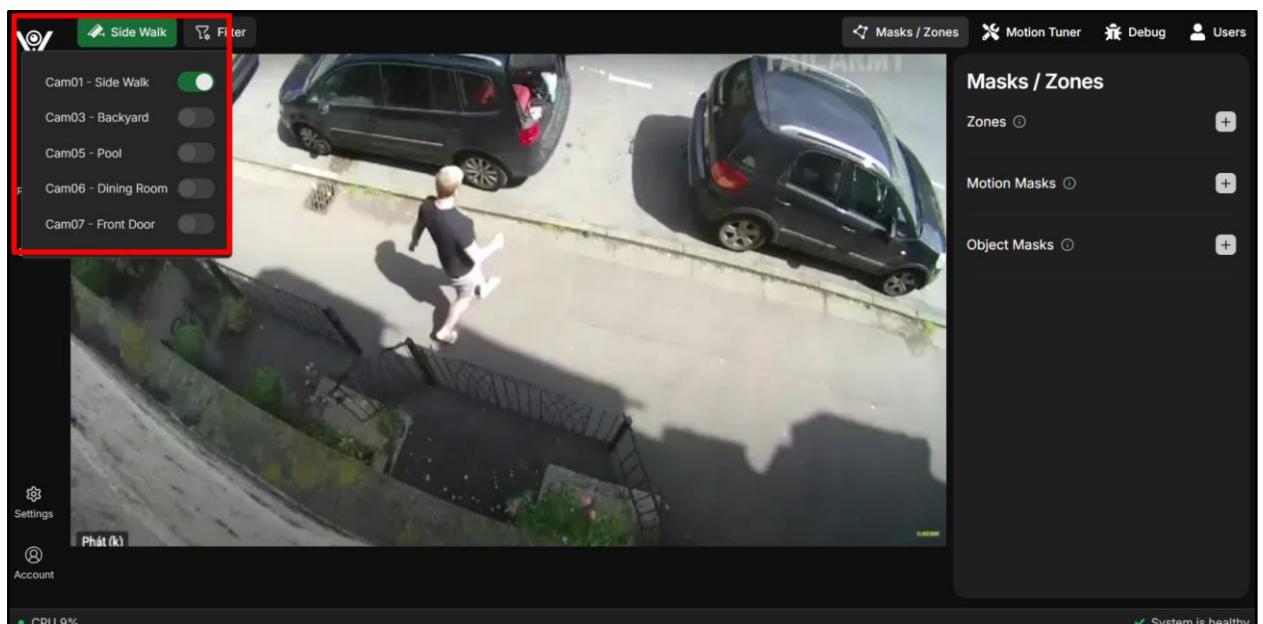


Figure 415. Manage motion masks - Create motion mask

- Step 2: Click on button “Add Motion Mask”

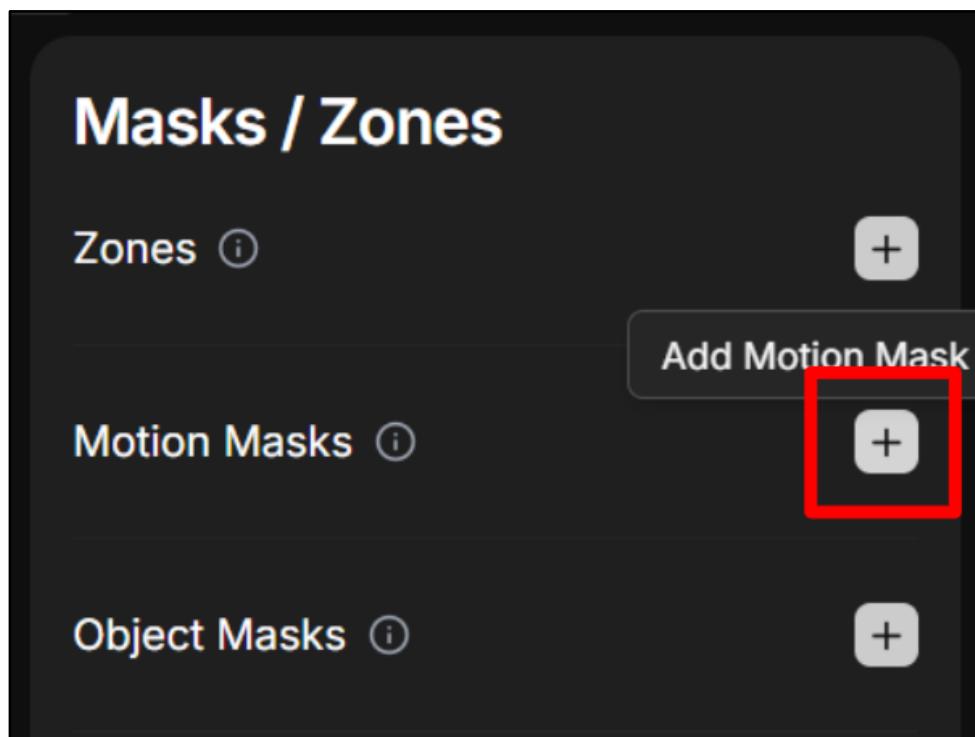


Figure 416. Manage motion masks - Create motion mask

...and “New Motion Mask” panel appears.

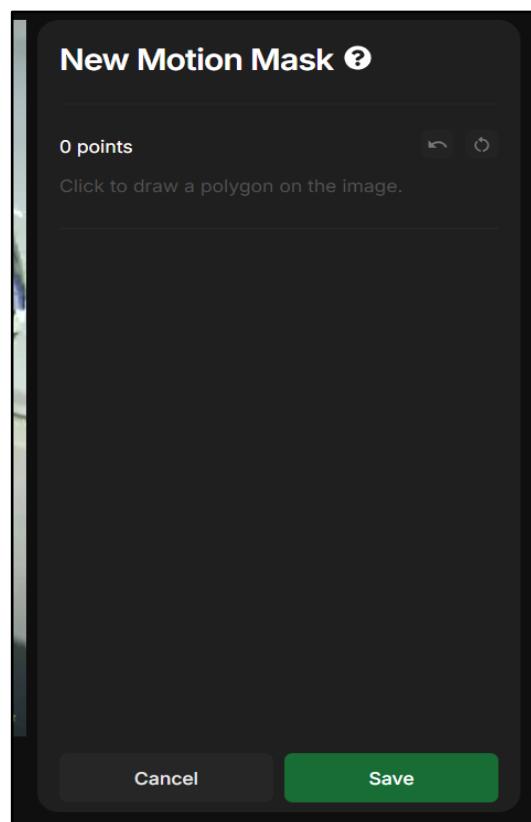


Figure 417. Manage motion masks - Create motion mask

- Step 3: Draw motion mask by draw a polygon that covers the motion mask area .

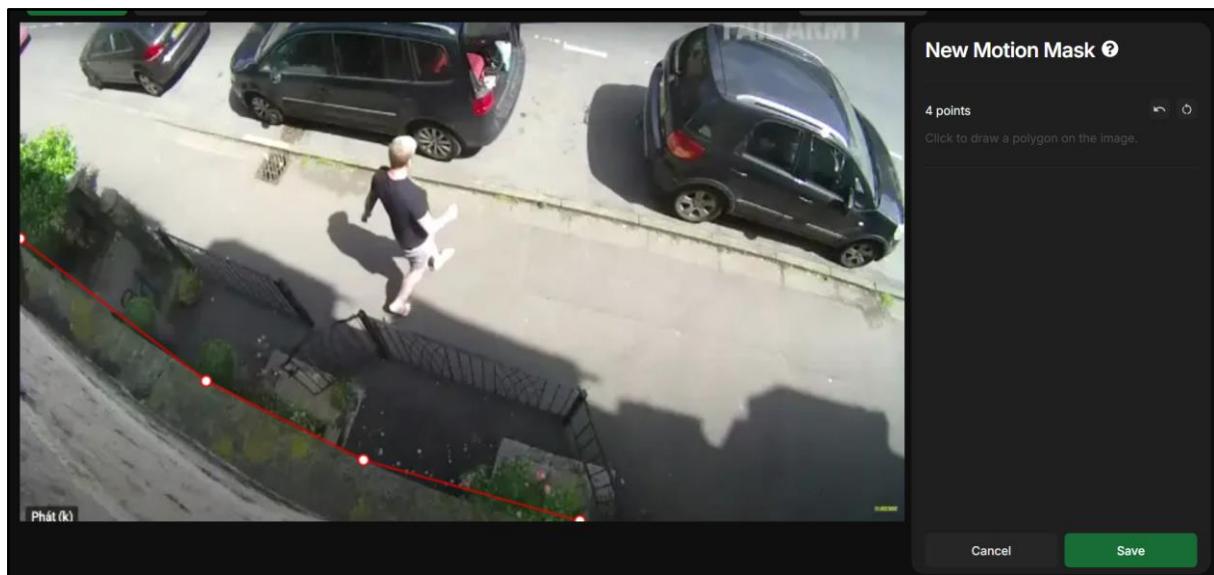


Figure 418. Manage motion masks - Create motion mask

... You can also Undo or Reset this motion mask by clicking button “Undo” and “Reset”



Figure 419. Manage motion masks - Create motion mask

- Step 4: Click on button “Save” to save new motion mask.

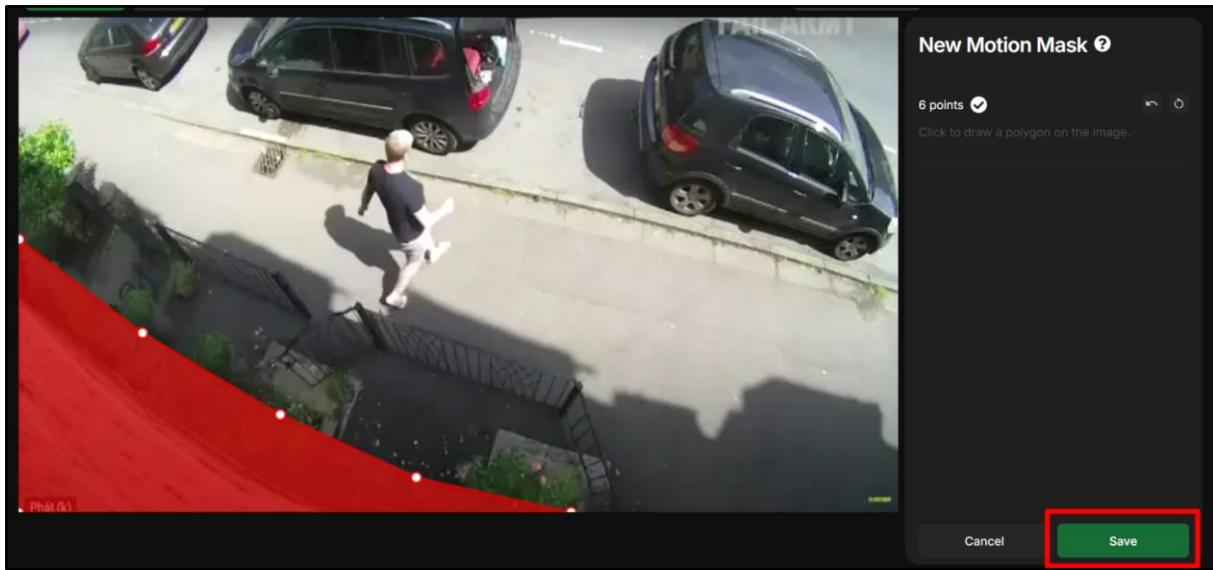


Figure 420. Manage motion masks - Create motion mask

... Then, Vigilant send you a message indicates motion mask has been added. By the way, restart is required.

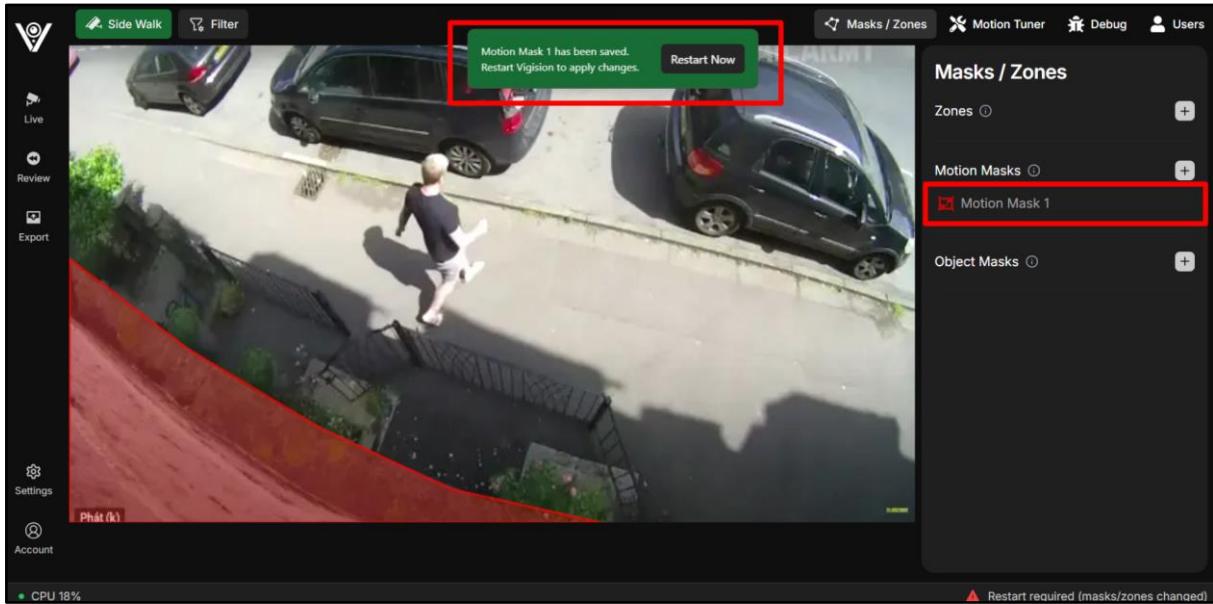


Figure 421. Manage motion masks - Create motion mask

...Once restarted, the motion mask is applied to the camera.

3.3.2.15.2 Update motion mask

- Step 1: To update motion mask, please click on button “Edit” on the row of the motion mask to be updated.

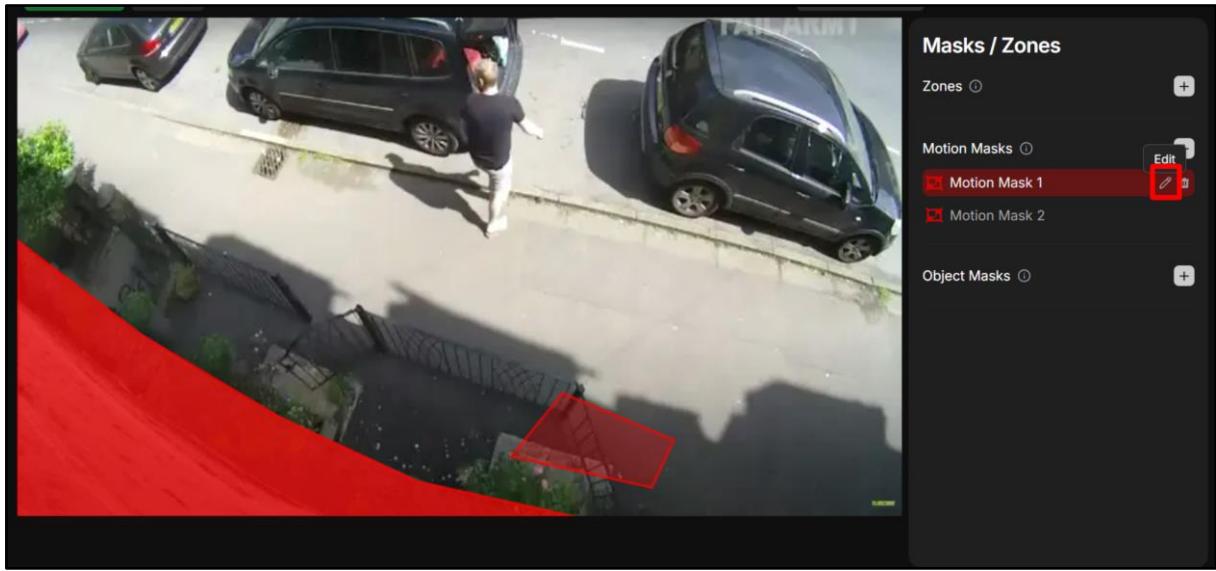


Figure 422. Manage motion masks - Update motion mask

- Step 2: Update motion mask by drag, drop, add new points for the old mask and click button “Save”.

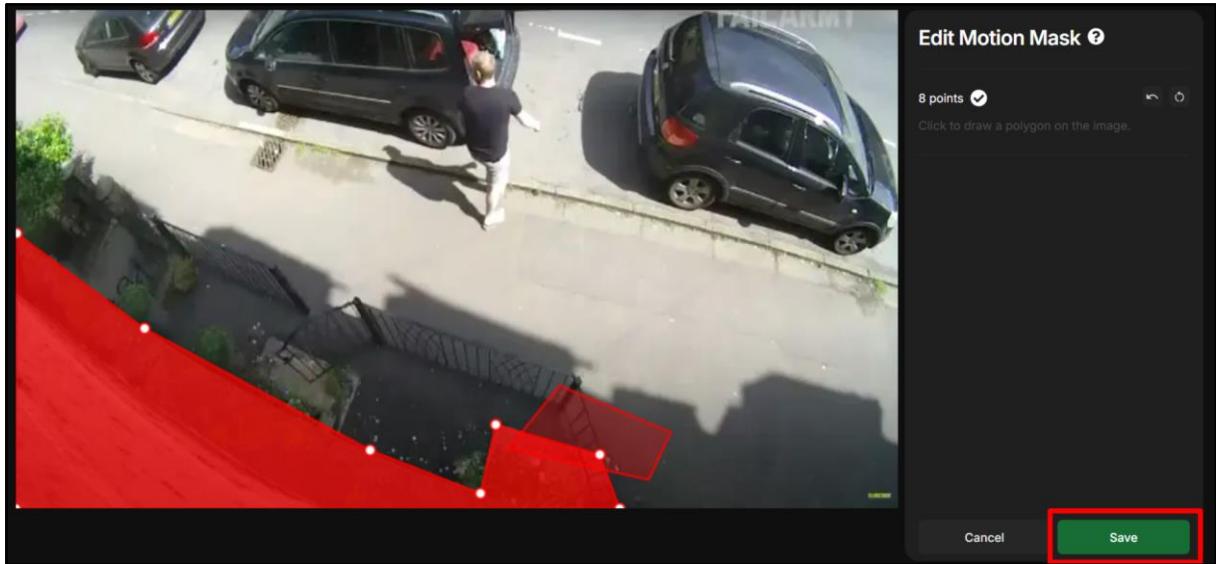


Figure 423. Manage motion masks - Update motion mask

- Step 3: After update motion mask, the system displays a message indicating that motion mask has been saved and restart is required.

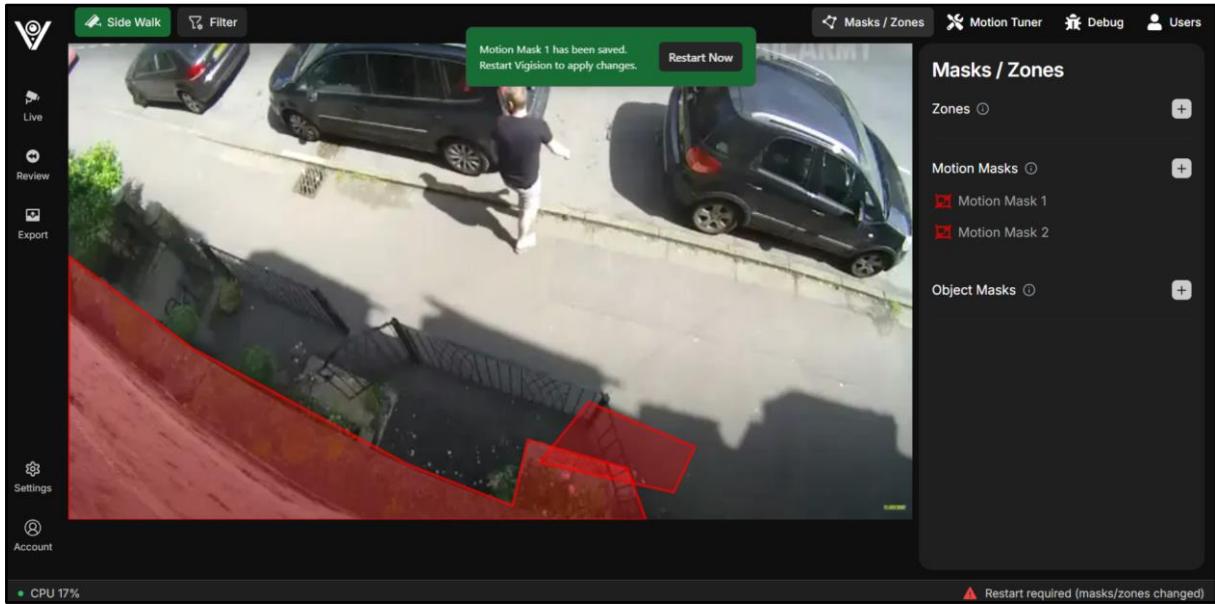


Figure 424. Manage motion masks - Update motion mask

- Step 4: Click on button “Restart Now”, wait and see the result.

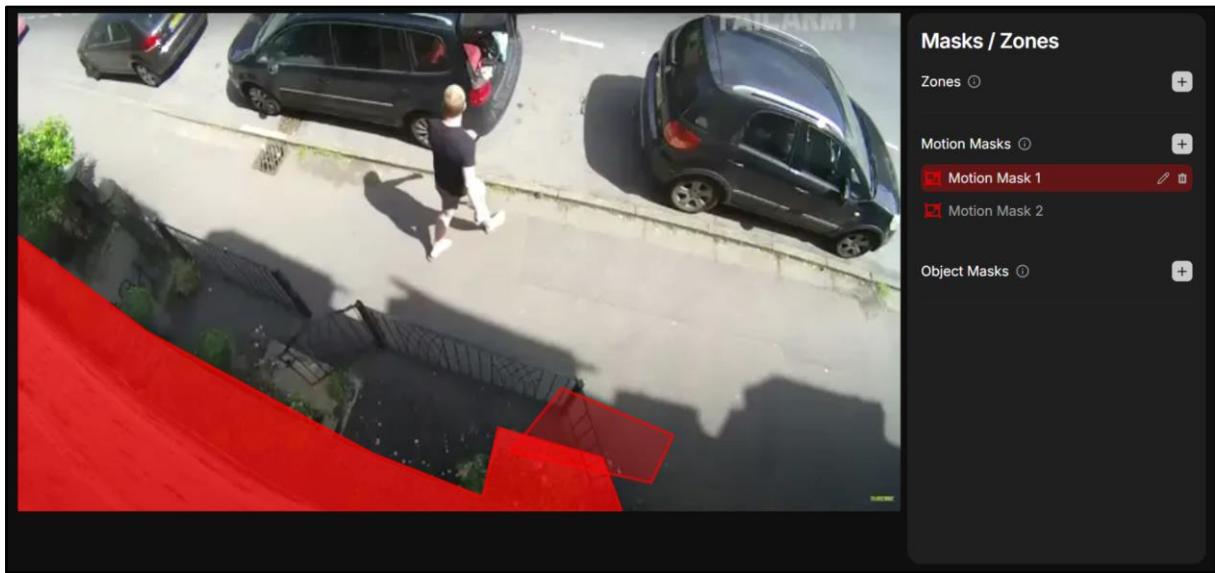


Figure 425. Manage motion masks - Update motion mask

3.3.2.15.3 Delete motion mask

- Step 1: To delete motion mask, please click on button “Delete” on the row of the motion mask to be deleted.

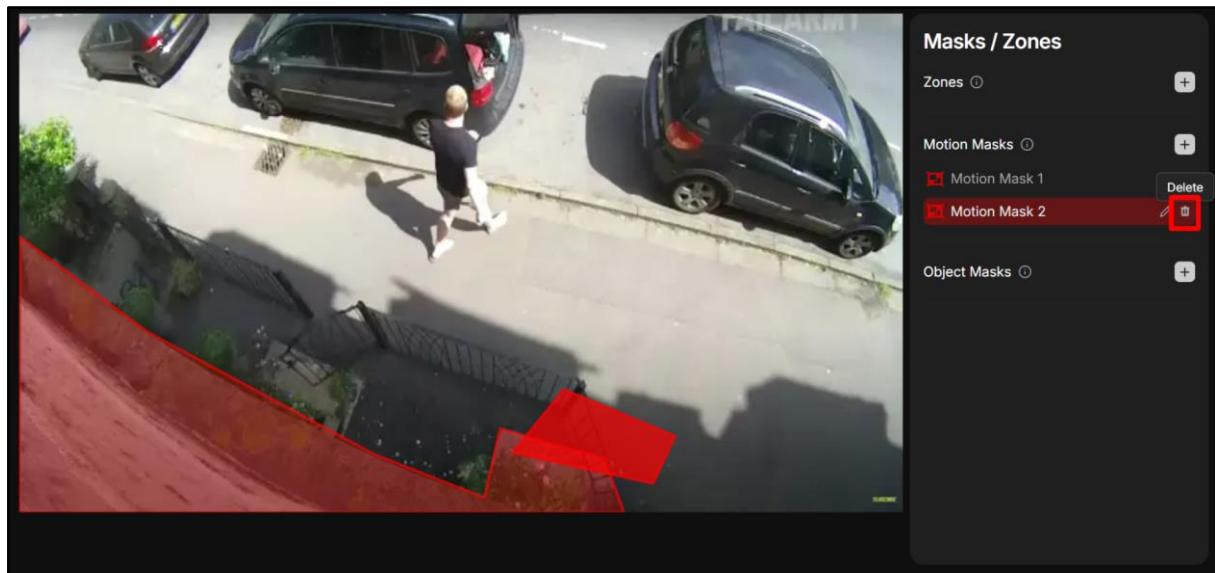


Figure 426. Manage motion masks - Delete motion mask

- Step 2: Confirm delete.

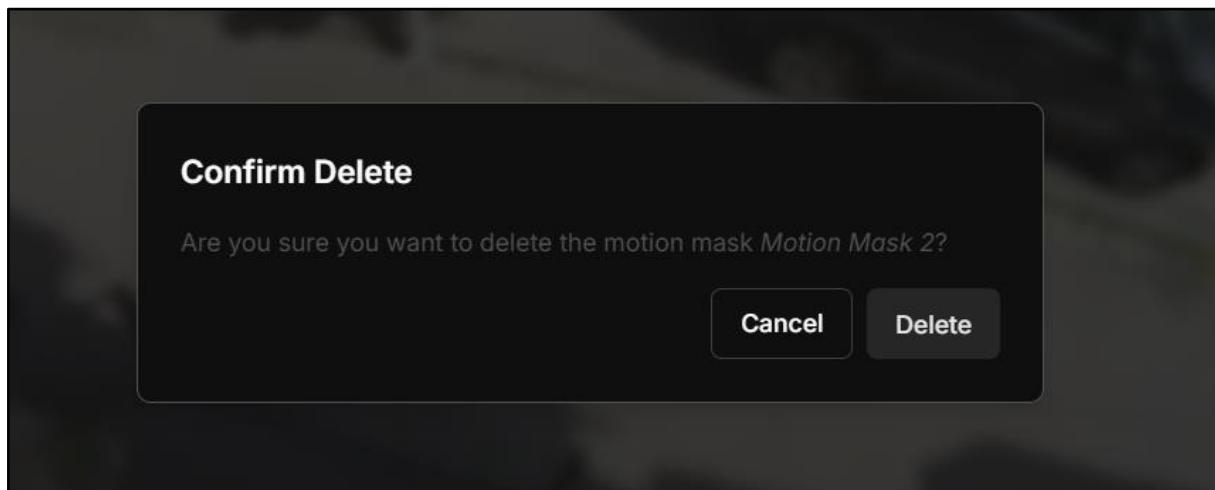


Figure 427. Manage motion masks - Delete motion mask

- Step 3: After delete motion mask, the system displays a message indicating that motion mask has been deleted and restart is required.

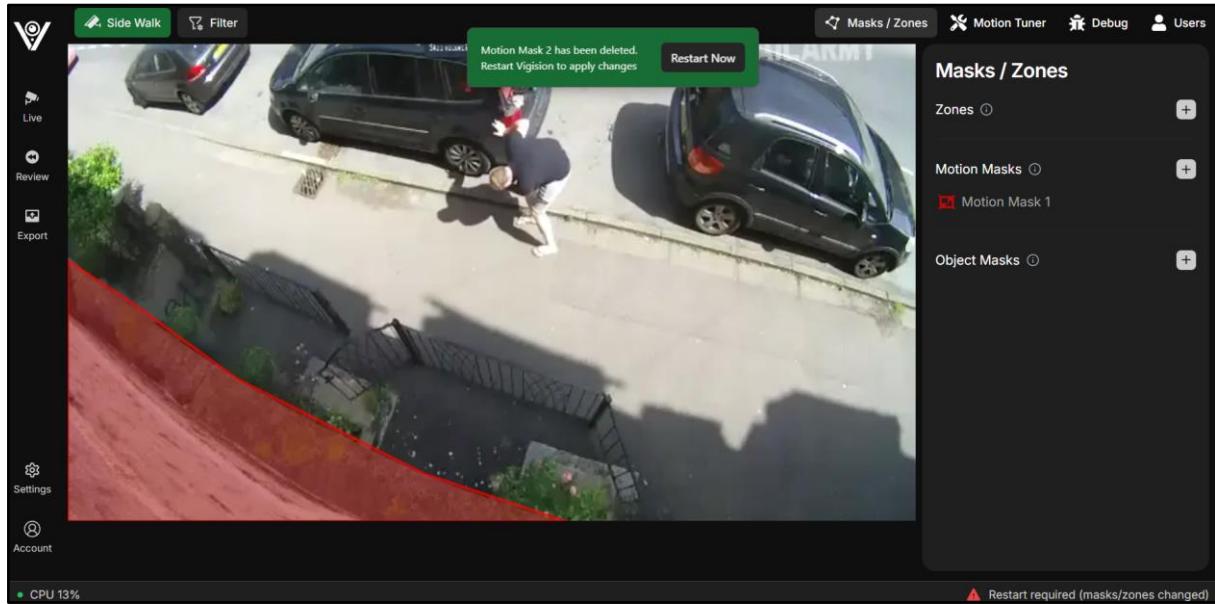


Figure 428. Manage motion masks - Delete motion mask

3.3.2.16 Manage zones

3.3.2.16.1 Create zone

- Step 1: Click green button to change to the camera you want to create zone for. Then click on button “Add Zone” to add zone for the selected camera.

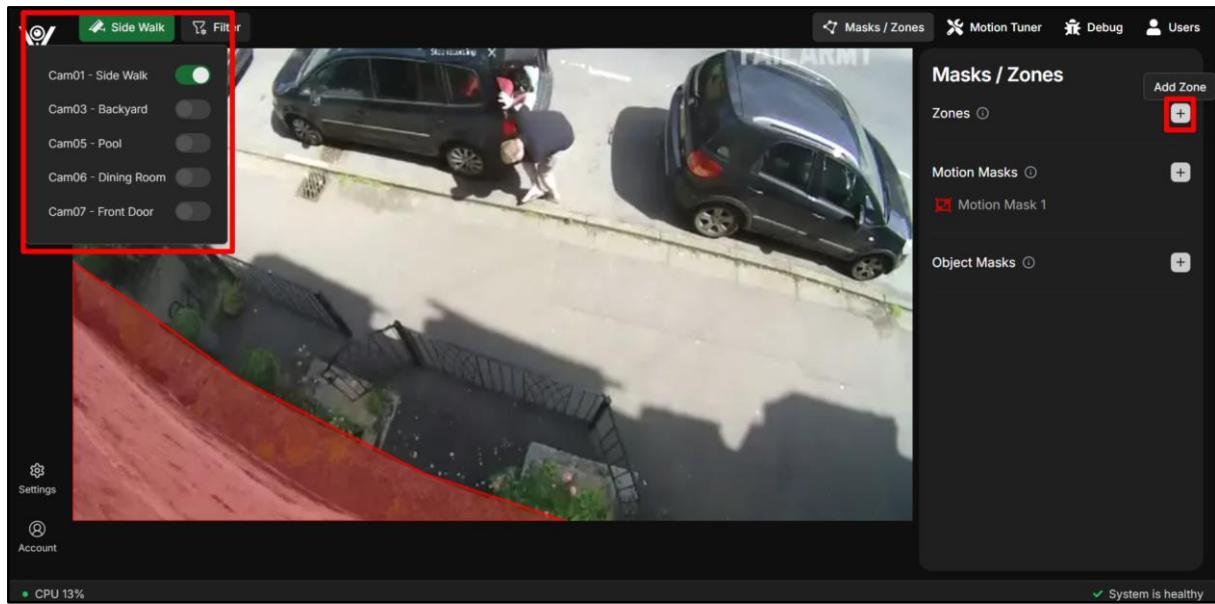


Figure 429. Create zone

...and “New Zone” panel appears.

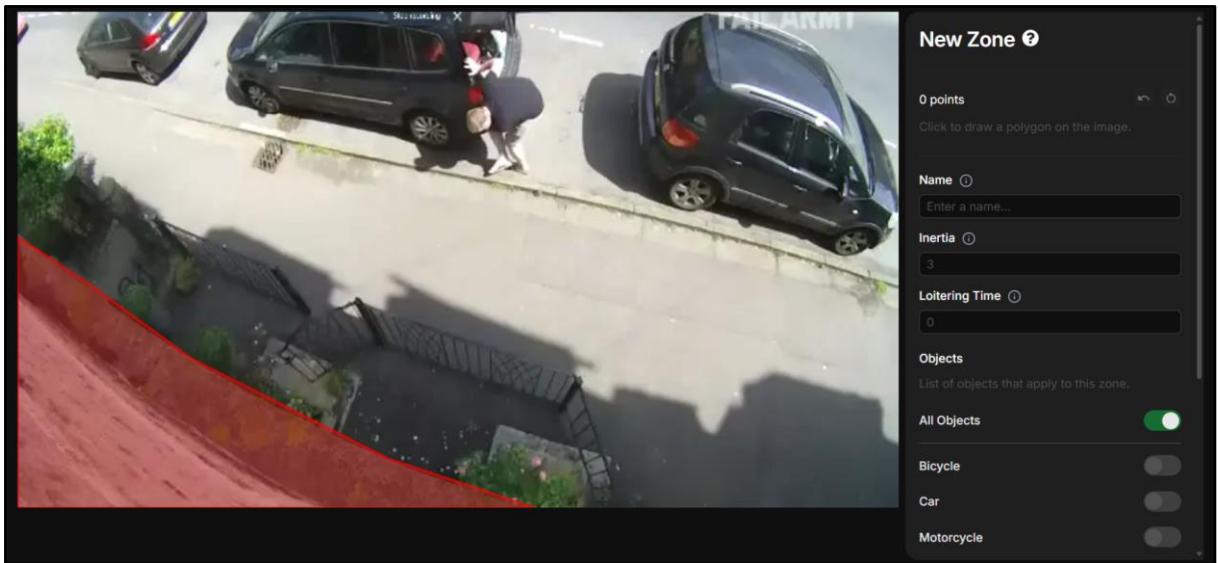


Figure 430. Manage zones - Create zone

- Step 2: Draw zone by draw a polygon that covers the zone. You can also Undo or Reset this zone by clicking button “Undo” and “Reset”.

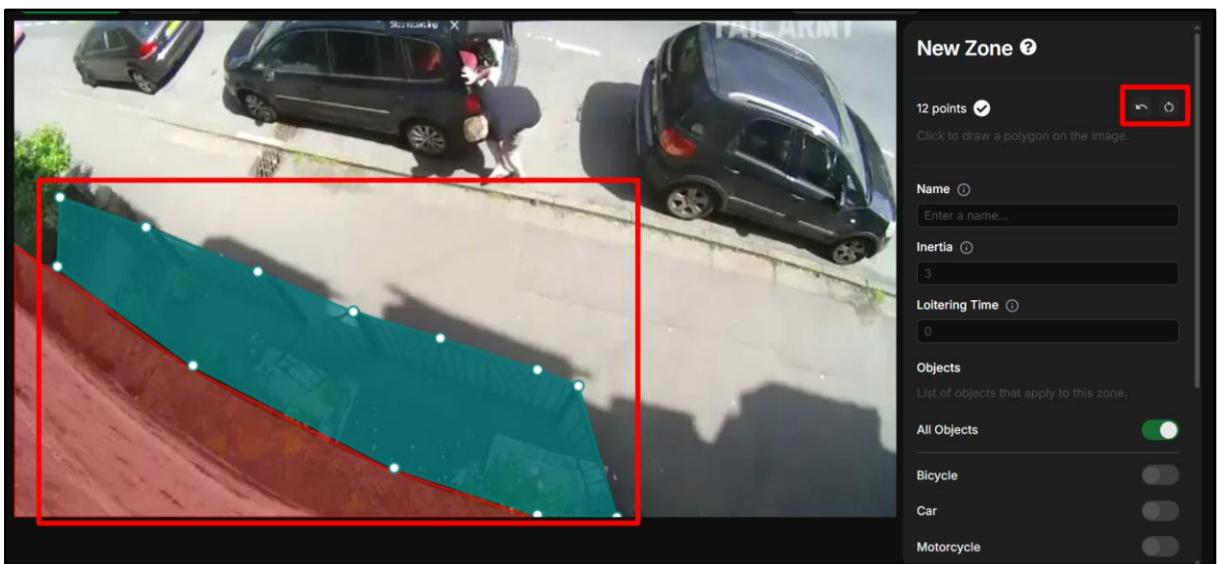


Figure 431. Manage zones - Create zone

- Step 3: Enter name for zone.



Figure 432. Manage zones - Create zone

- Step 4: Enter Inertia for zone

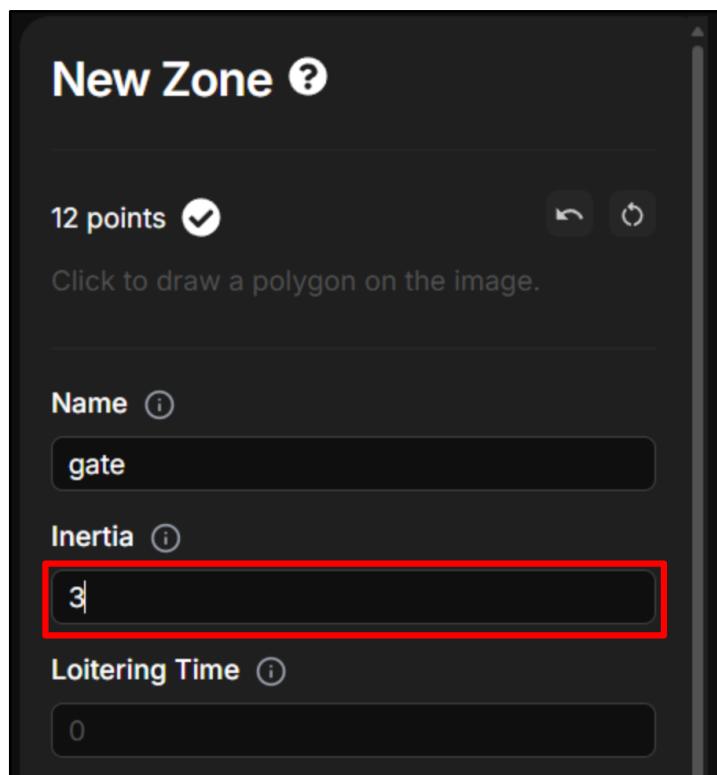


Figure 433. Manage zones - Create zone

- Step 5: Enter Loitering Time for zone

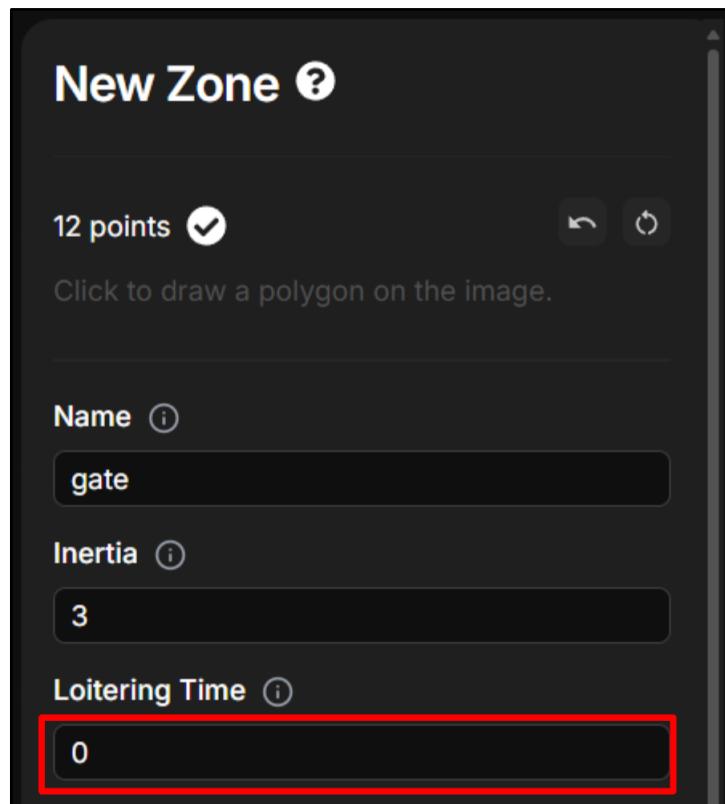


Figure 434. Manage zones - Create zone

- Step 6: Choose objects that apply to the zone.

Example 1: Apply all objects (left image) or apply object person to the zone (right image).

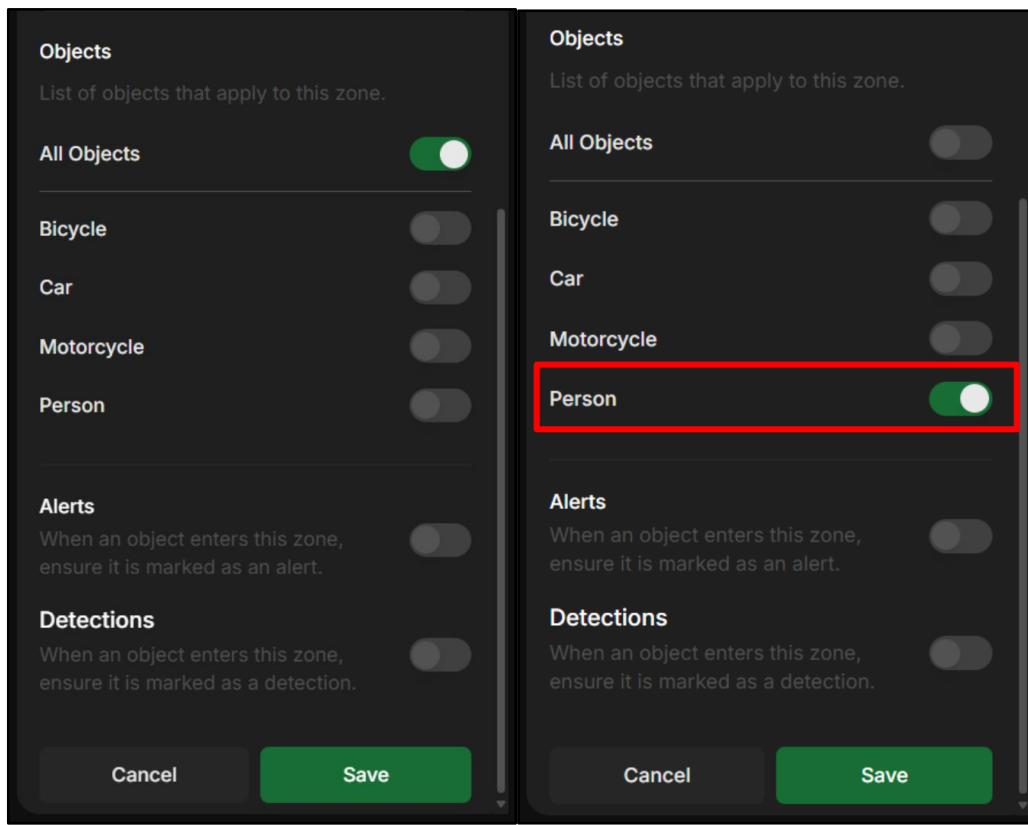


Figure 436. Manage zones - Create zone

- Step 7: (Optional) If you want that when an object enters this zone, it is marked as an alert, please enable **Alert**.

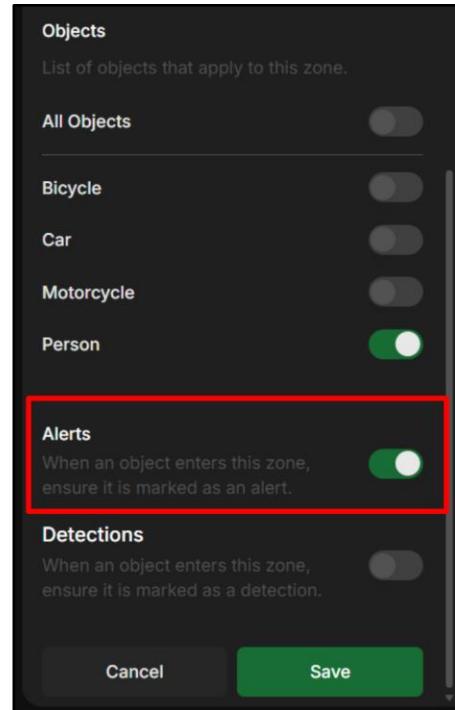


Figure 437. Manage zones - Create zone

- Step 8: (Optional) If you want that when an object enters this zone, it is marked as a detection, please enable **Detections**.
- Step 9: Click on button “**Save**” to save new zone.

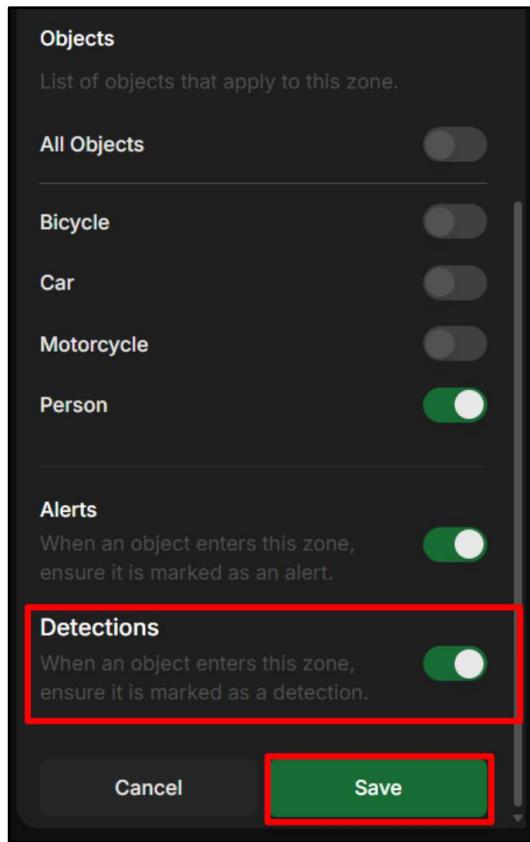


Figure 438. Manage zones - Create zone

... Then, Vigision sends you a message indicates zone has been added. You can see that new By the way, restart is required.

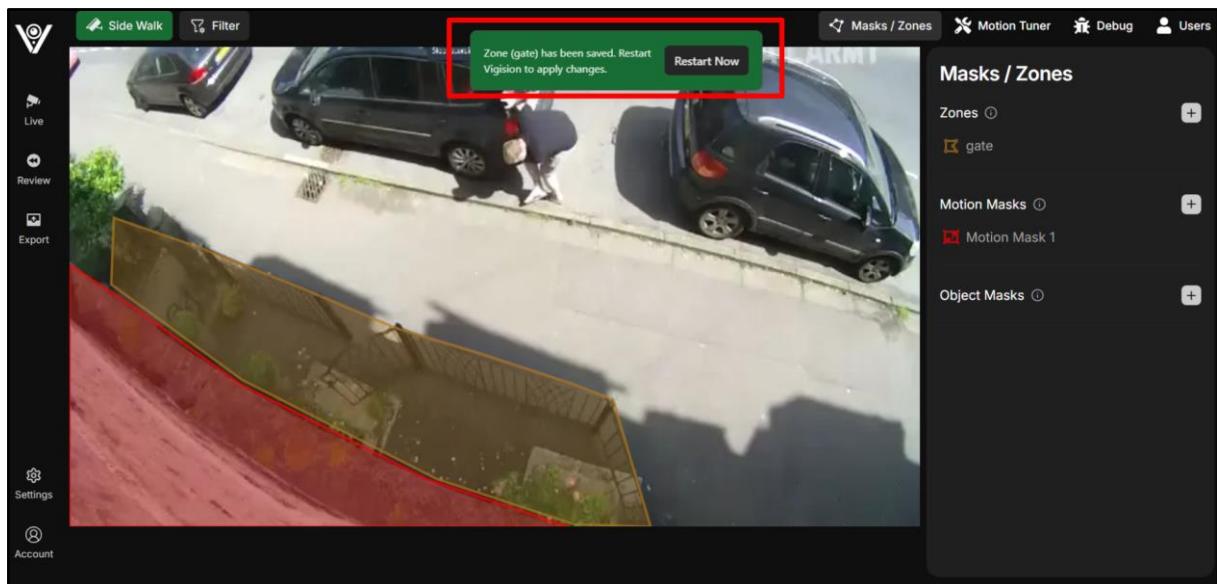


Figure 439. Manage zones - Create zone

You can double check by finding the newly created zone and its polygon (yellow) is displayed as below.

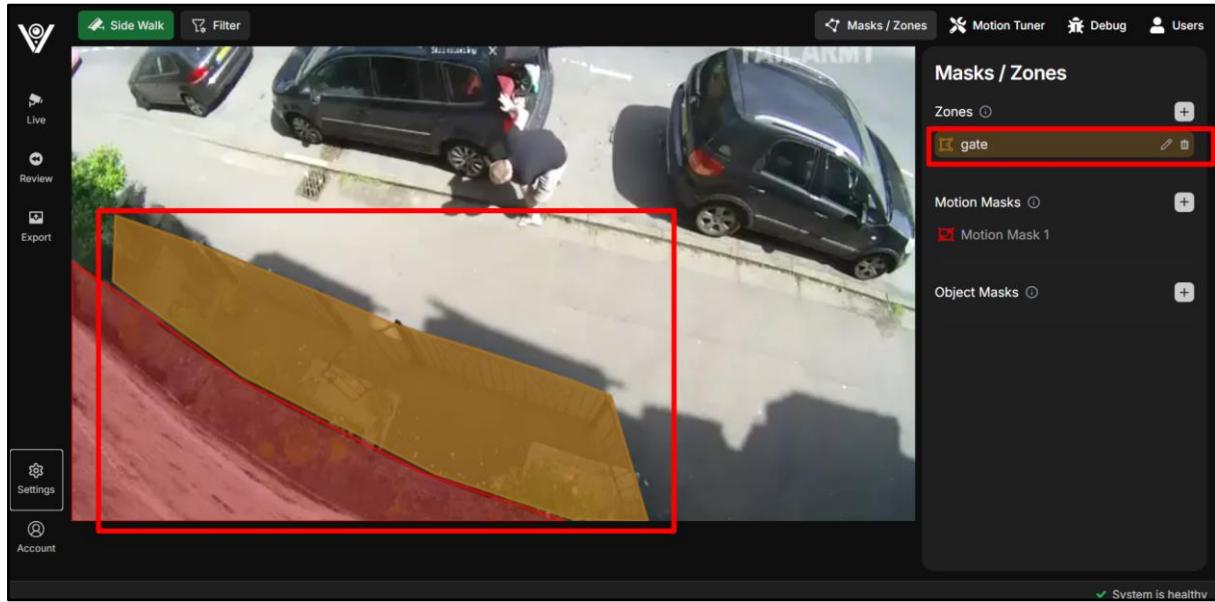


Figure 440. Manage zones - Create zone

3.3.2.16.2 Update zone

- Step 1: Click green button to change to the camera you want to create zone for. Then, click on button “Edit” on the row of the zone (ex: gate) to be updated.

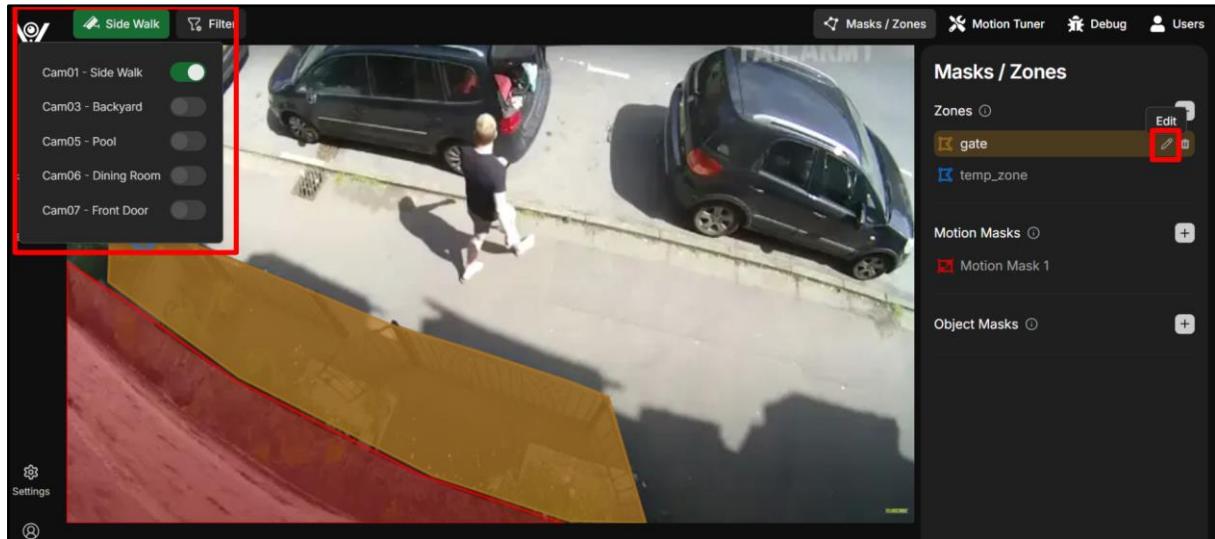


Figure 441. Manage zones - Update zone

- Step 2: (Optional) Update motion mask by drag, drop, add new points for the old mask

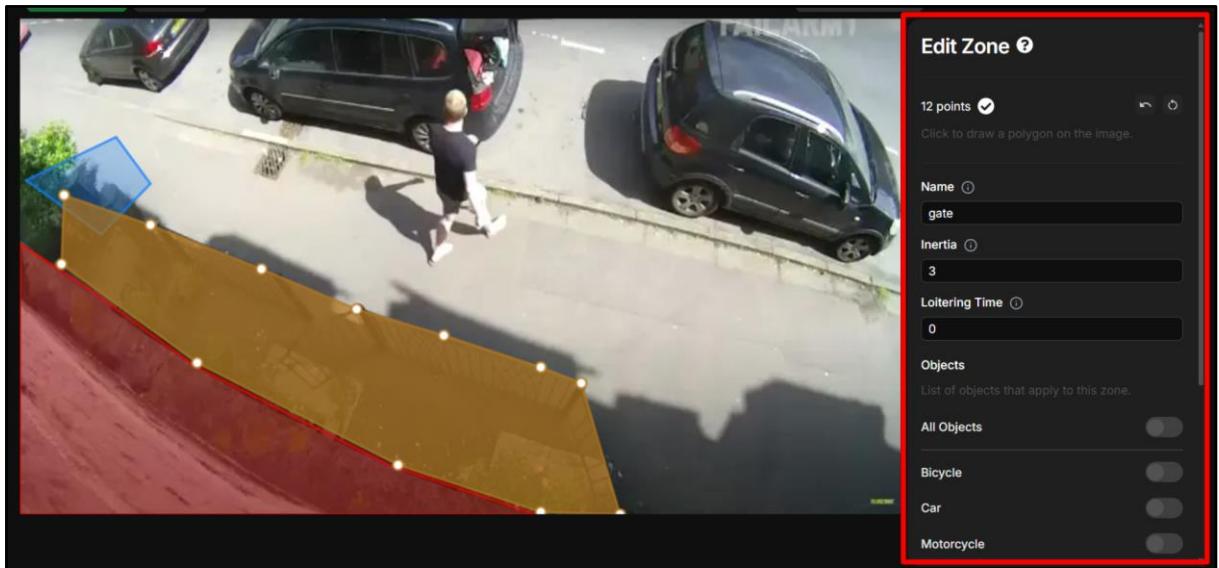


Figure 442. Manage zones - Update zone

The image below shows an example of edited zone polygon which is expand to the left.

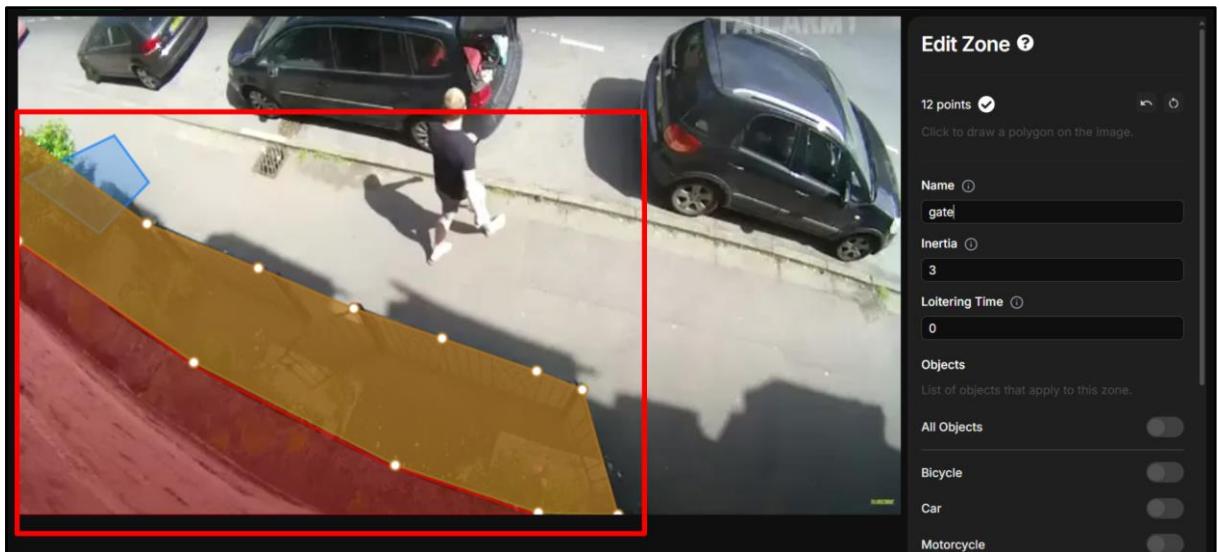


Figure 443. Manage zones - Update zone

- Step 3: (Optional) Update other zone configuration.

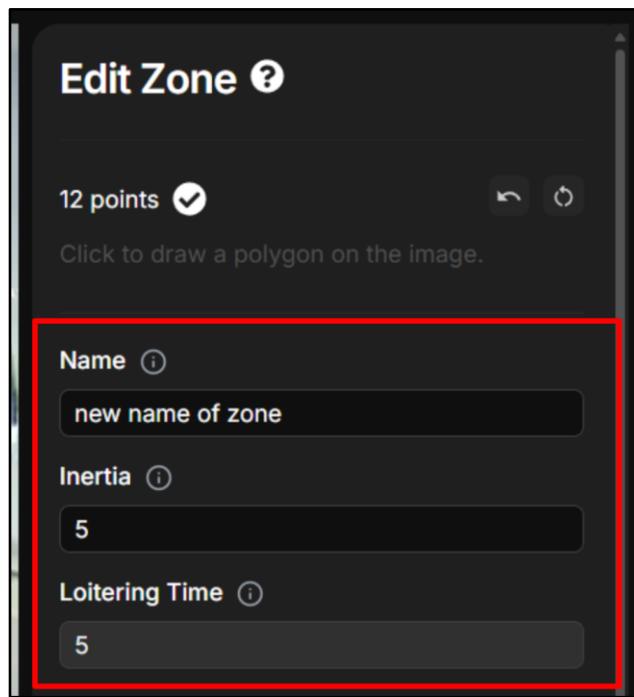


Figure 444. Manage zones - Update zone

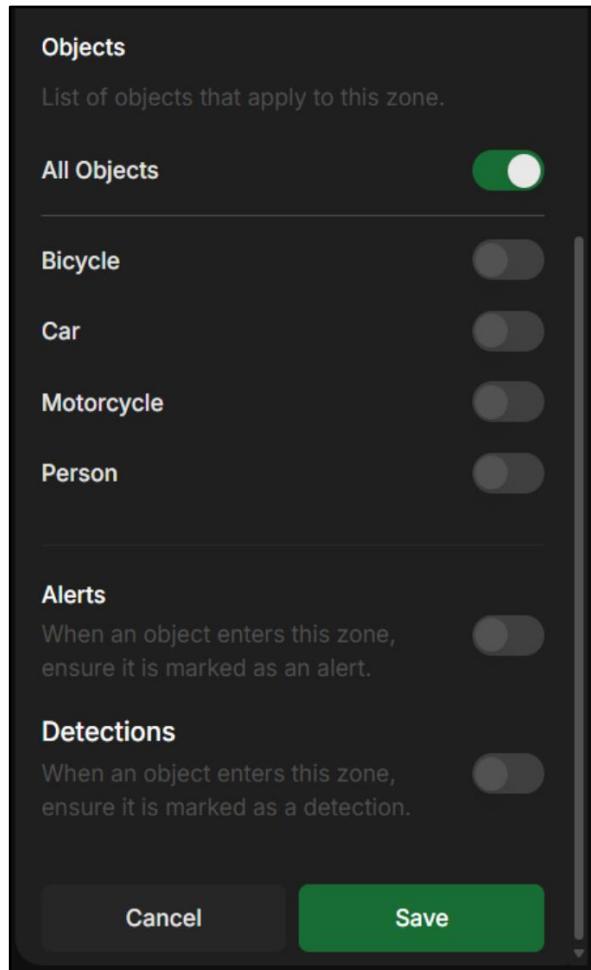


Figure 445. Manage zones - Update zone

- Step 4: (Optional) Click “Save” button to submit change.

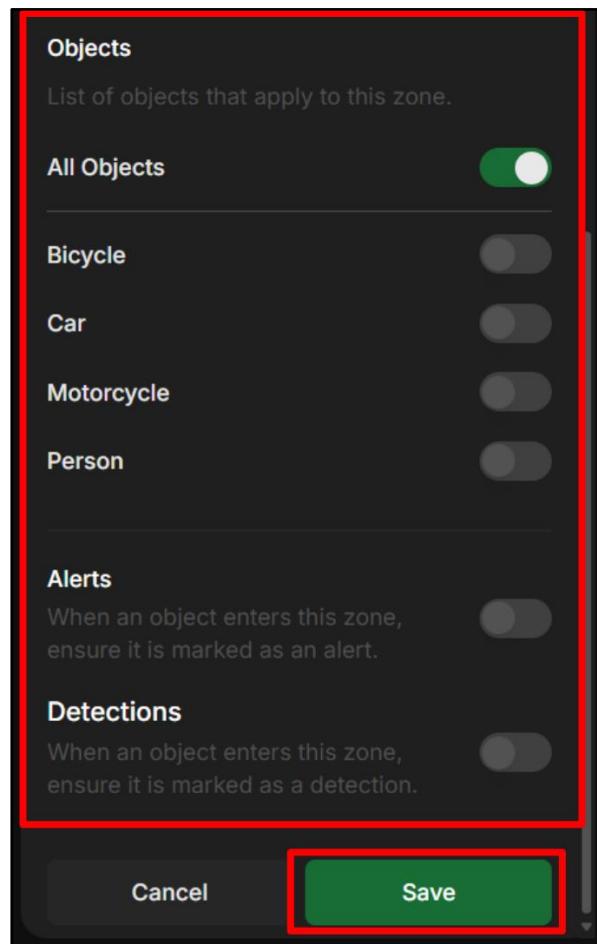


Figure 446. Manage zones - Update zone

- Step 5: After update zone, the system displays a message indicating that zone has been saved and restart is required.

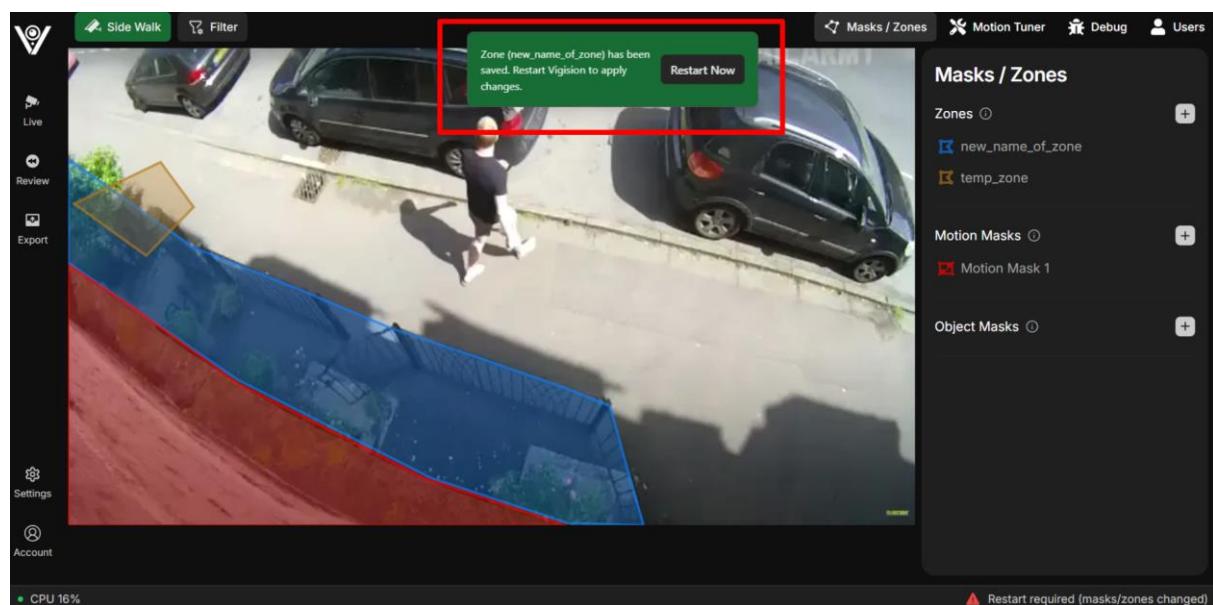


Figure 447. Manage zones - Update zone

You can double check by finding the updated zone information as below.

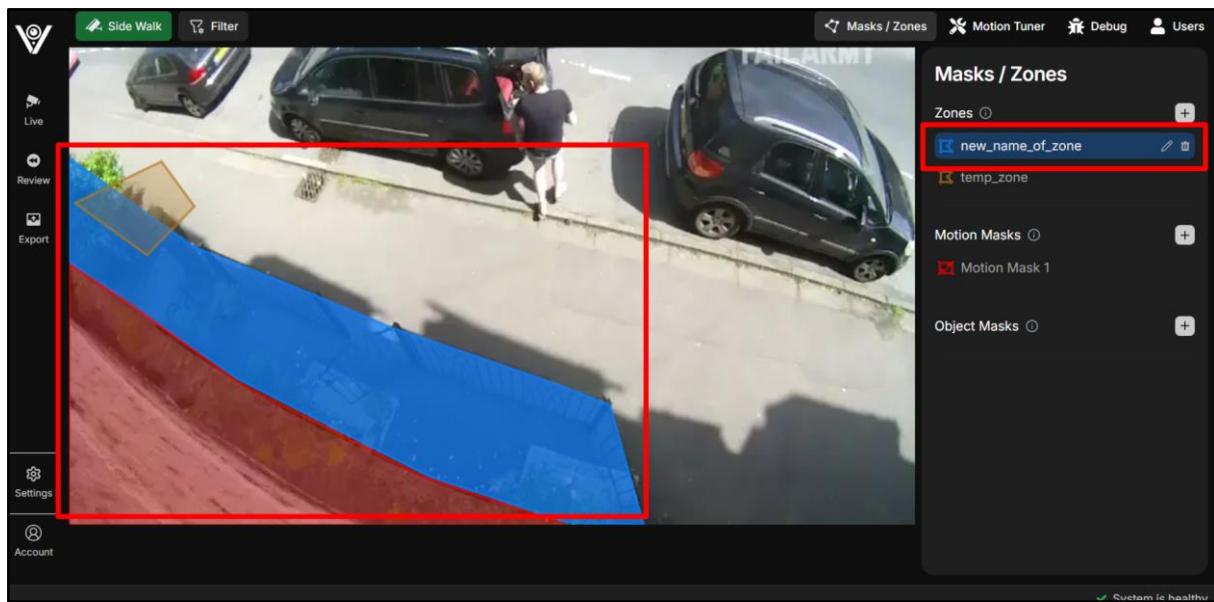


Figure 448. Manage zones - Update zone

3.3.2.16.3 Delete zone

- Step 1: To delete zone, please click on button “Delete” on the row of the zone to be deleted.

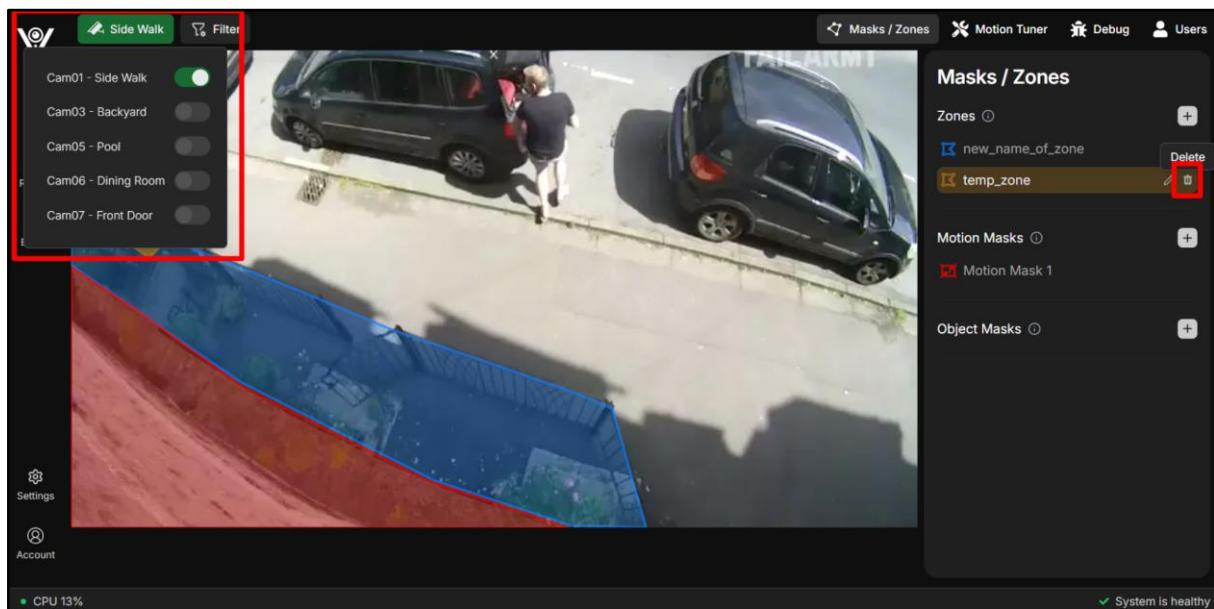


Figure 449. Manage zones - Delete zone

- Step 2: Confirm delete.

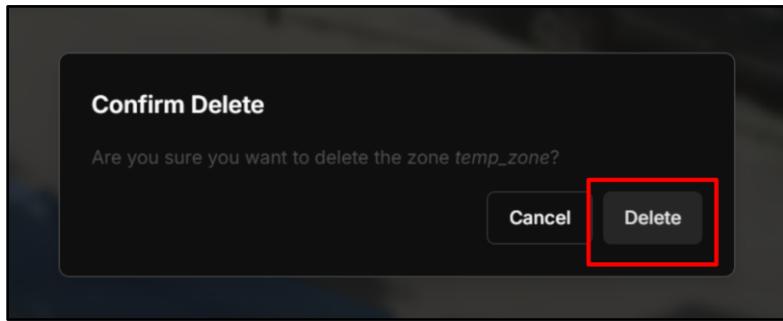


Figure 450. Manage zones - Delete zone

- Step 3: After delete zone, the system displays a message indicating that zone has been deleted and restart is required.

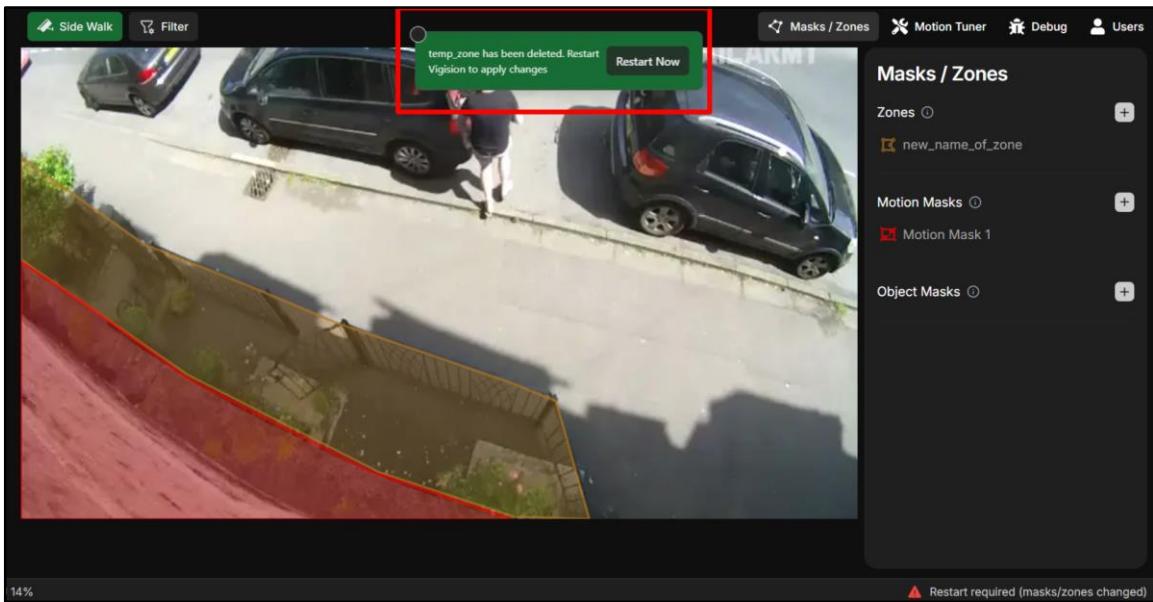


Figure 451. Manage zones - Delete zone

The image below shows that temp_zone has been deleted.

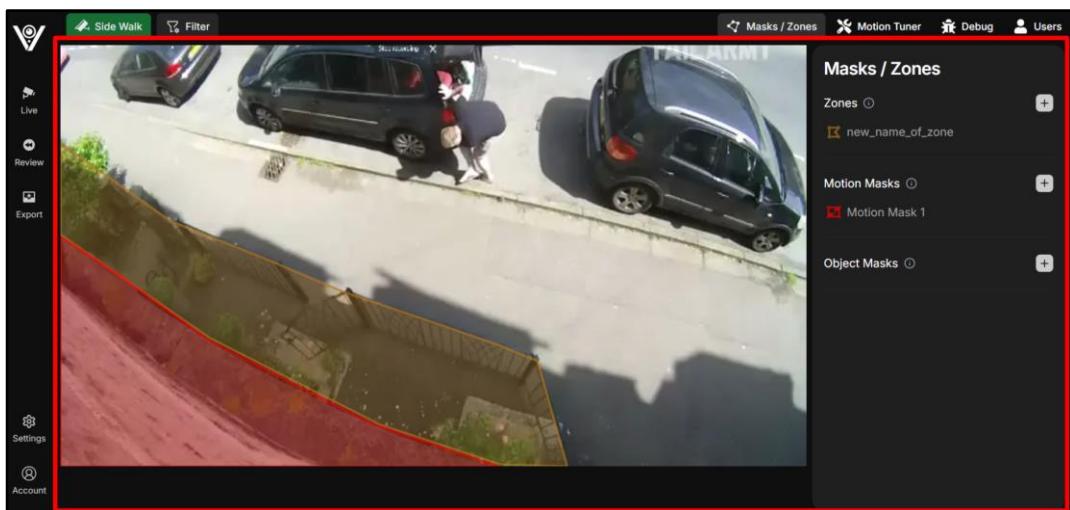


Figure 452. Manage zones - Delete zone

3.3.2.17 Manage object masks

3.3.2.17.1 Create object mask

- Step 1: Click green button to change to the camera you want to create object mask for. Then click on button “Add Object Mask”

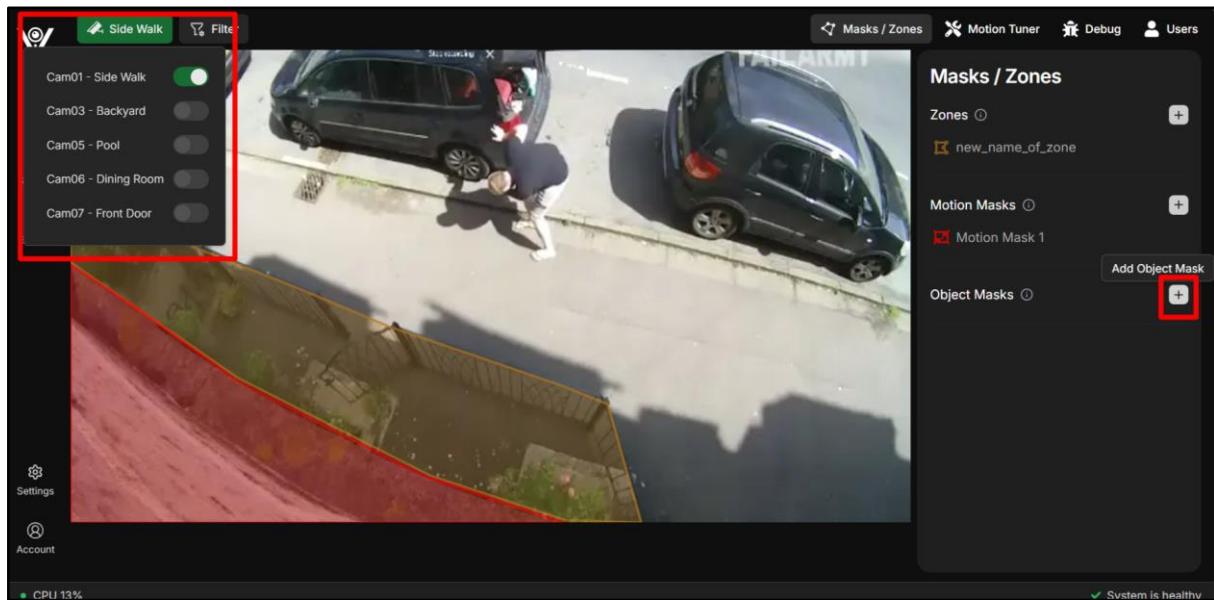


Figure 453. Manage object masks - Create object mask

After that, “New Motion Mask” panel appears.

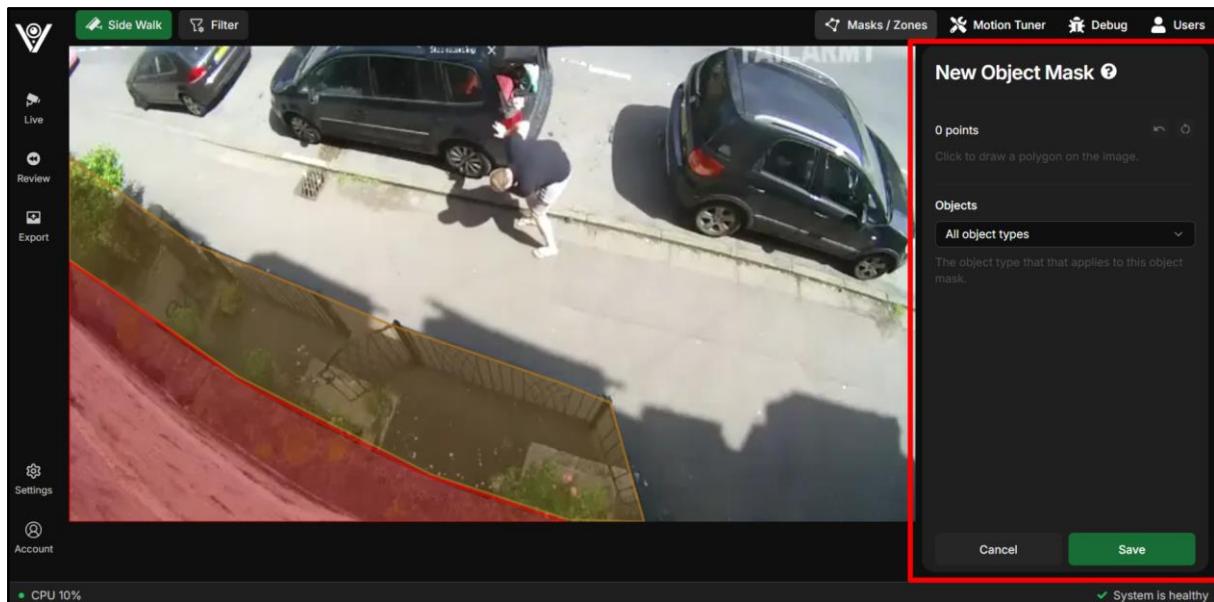


Figure 454. Manage object masks - Create object mask

- Step 2: Draw object mask by draw a polygon that covers the object mask area. You can also Undo or Reset this motion mask by clicking button “Undo” and “Reset”.

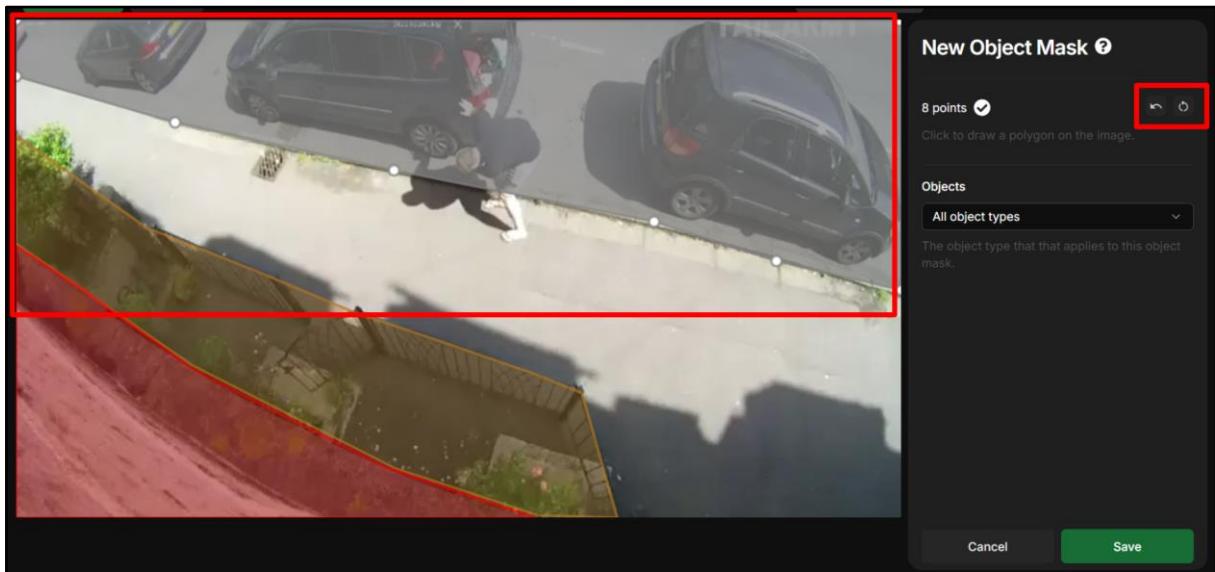


Figure 455. Manage object masks - Create object mask

- Step 3: Select objects

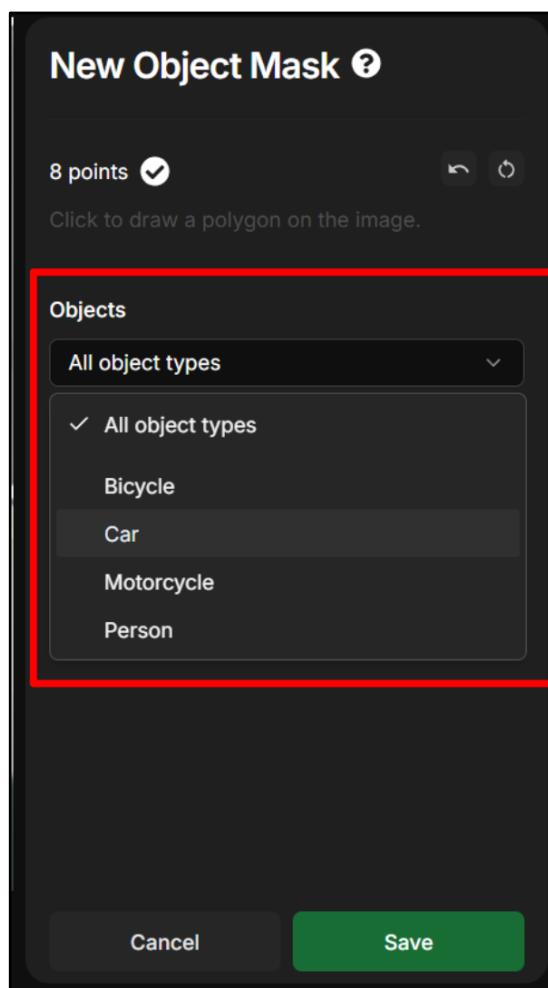


Figure 456. Manage object masks - Create object mask

- Step 4: Click on button “Save” to save new object mask.

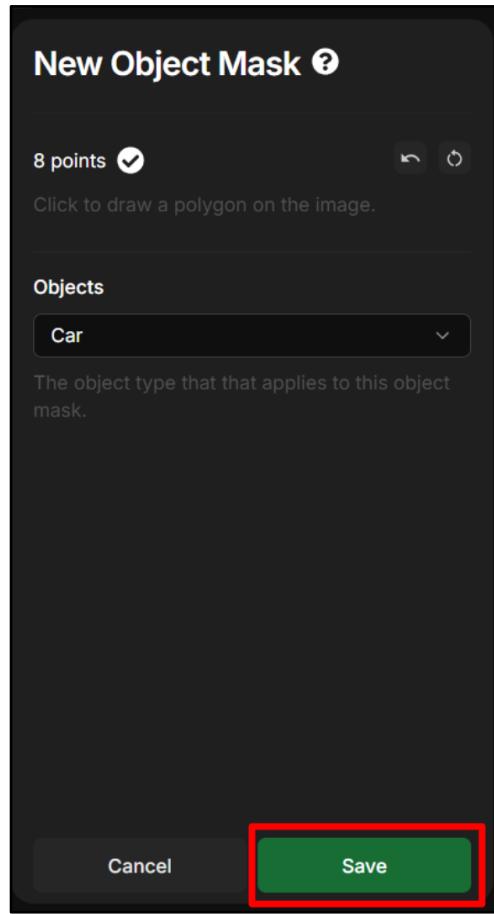


Figure 457. Manage object masks - Create object mask

- Step 5: After update object mask, the system displays a message indicating that object mask has been and restart is required.

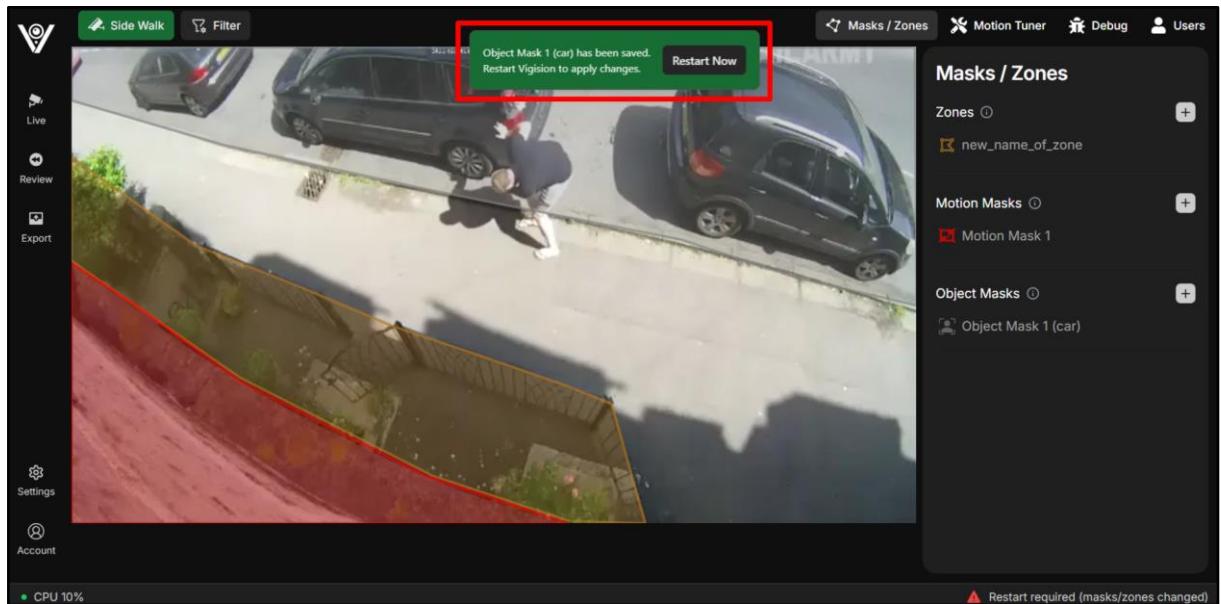


Figure 458. Manage object masks - Create object mask

You can double check by finding the updated object mask information as below.

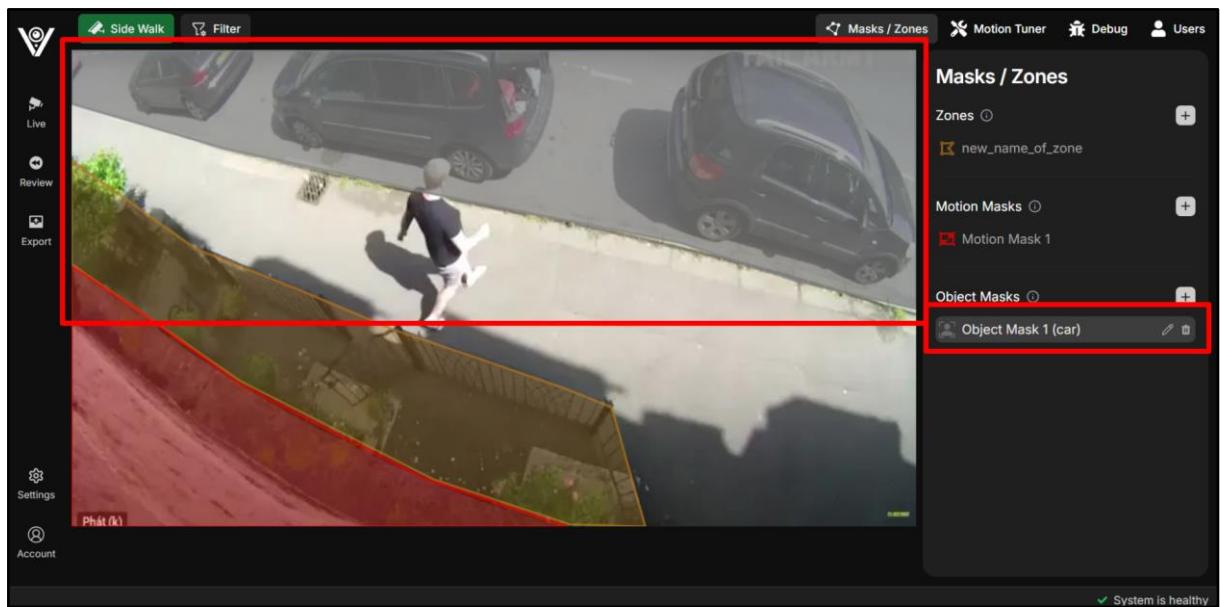


Figure 459. Manage object masks - Create object mask

3.3.2.17.2 Update object mask

- Step 1: Click green button to change to the camera you want to create object mask for. Then, click on button “Edit” on the row of the object mask(ex: gate) to be updated.

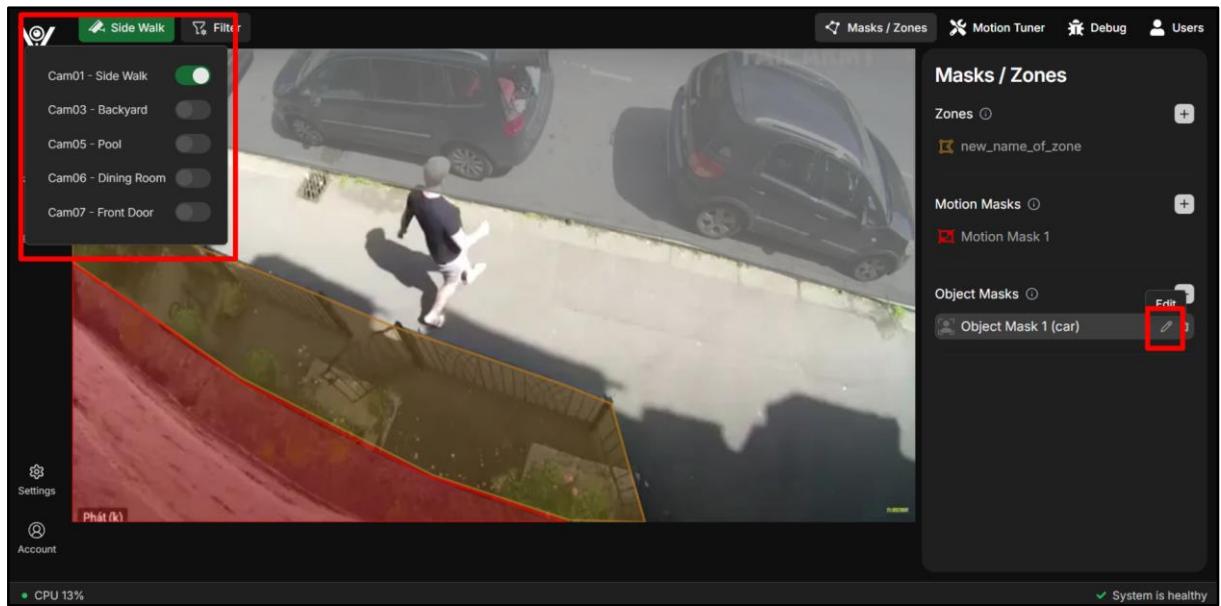


Figure 460. Manage object masks - Update object mask

- Step 2: Update object mask's attributes such as its polygon, objects. And finally, click on button “Save”.

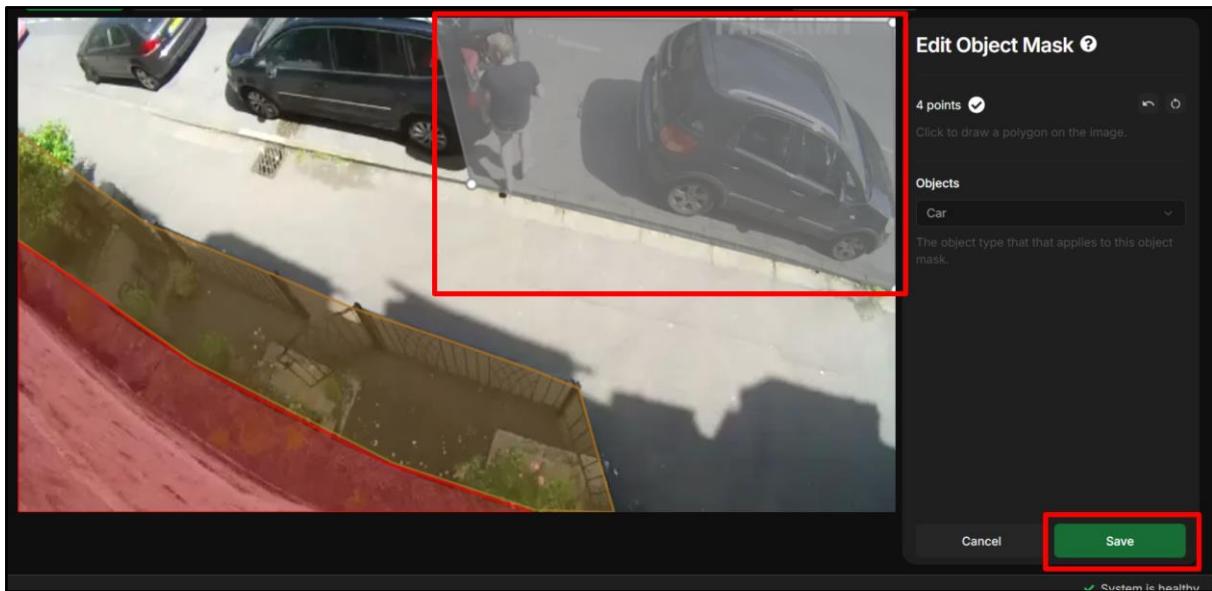


Figure 461. Manage object masks - Update object mask

- Step 3: After update object mask, the system displays a message indicating that object mask has been saved and restart is required.

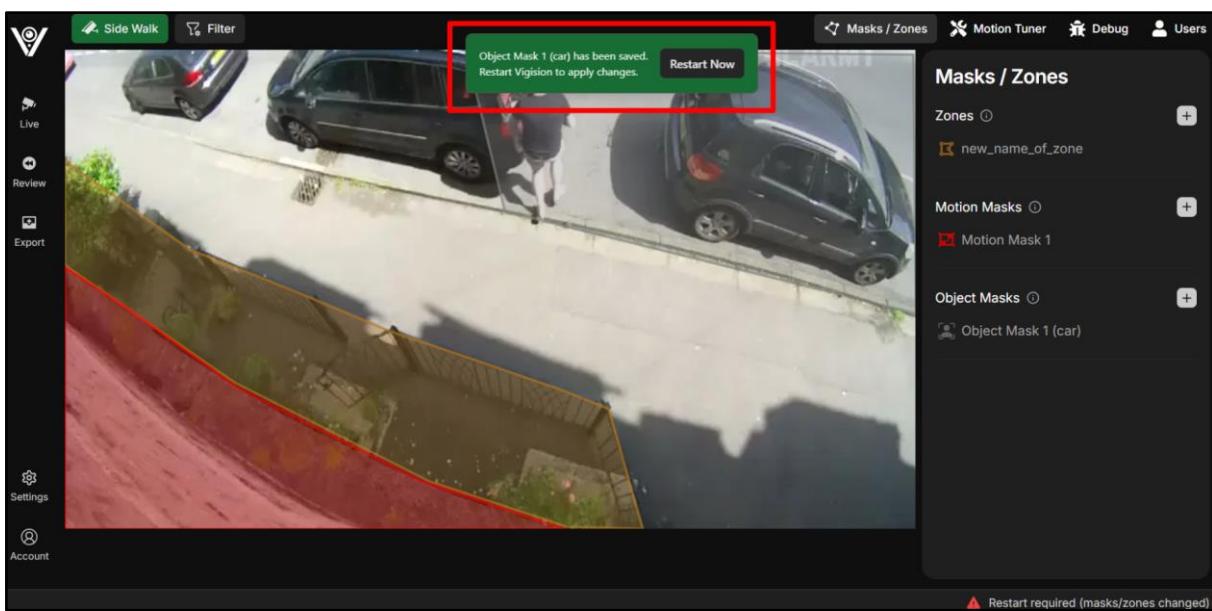


Figure 462. Manage object masks - Update object mask

You can double check by finding the updated object mask as below. Object mask has been changed.

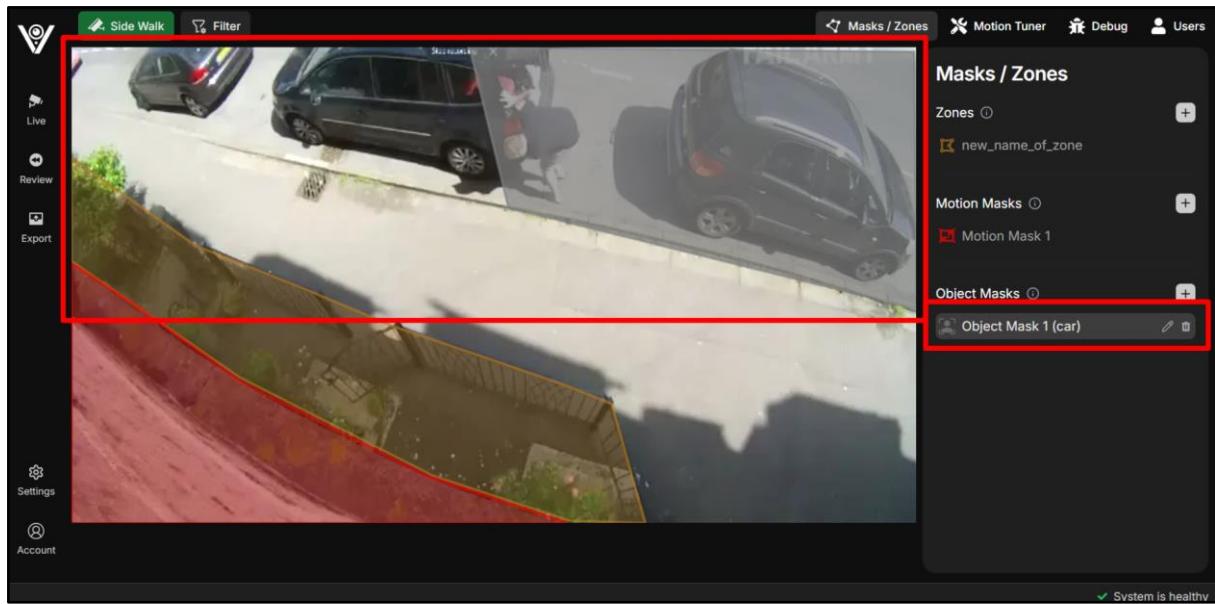


Figure 463. Manage object masks - Update object mask

3.3.2.17.3 Delete object mask

Step 1: To delete object mask, please click on button “Delete” on the row of the object mask to be deleted.

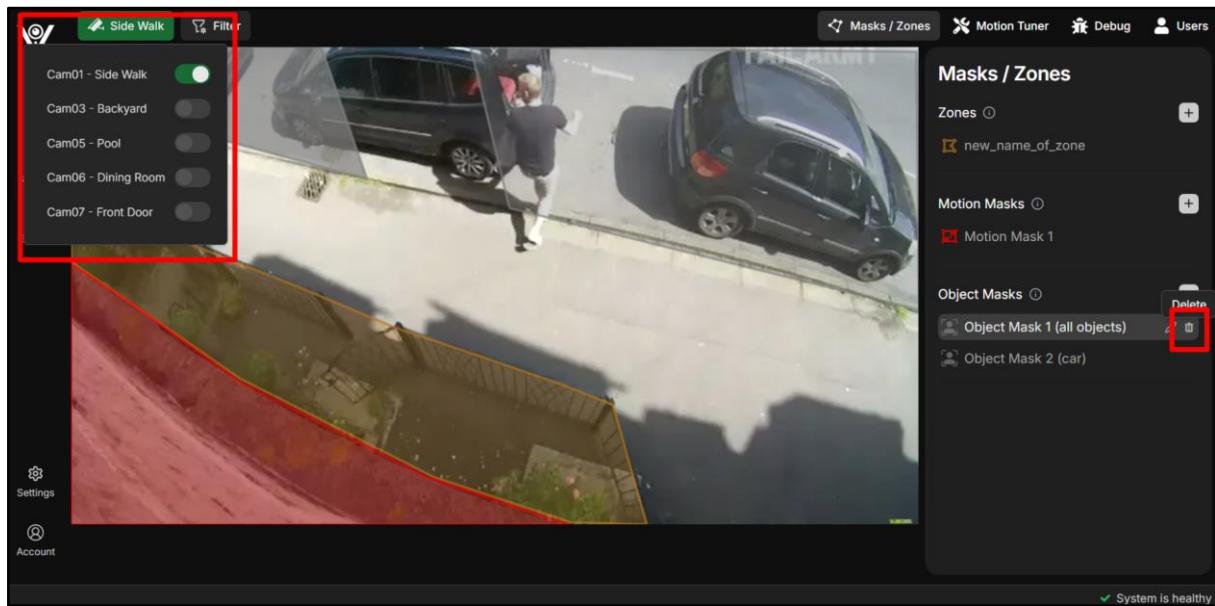


Figure 464. Manage object masks - Delete object mask

- Step 2: Confirm delete.

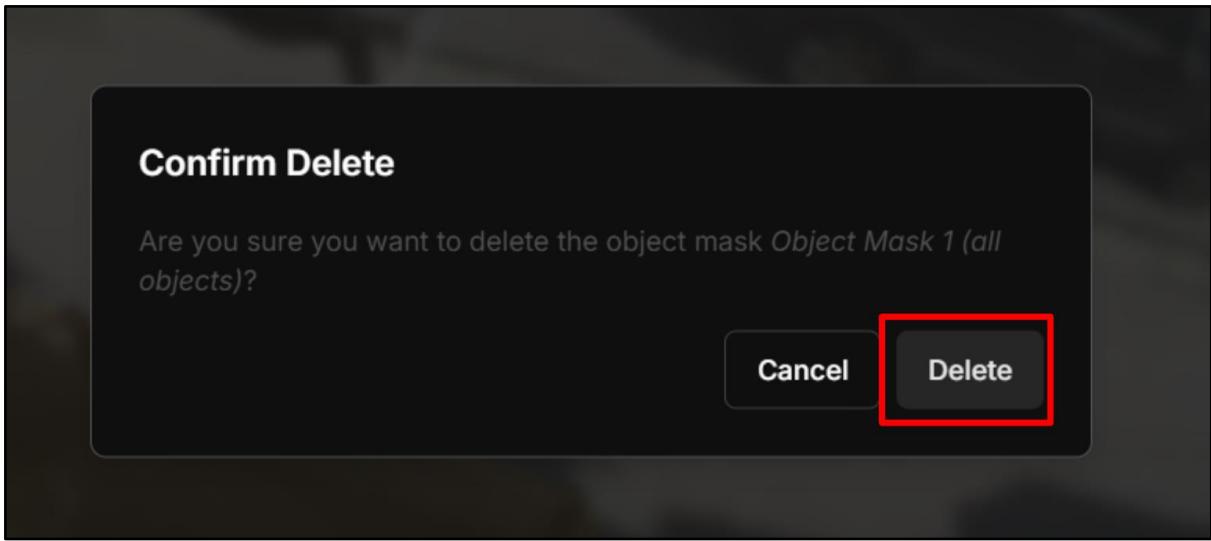


Figure 465. Manage object masks - Delete object mask

- Step 3: After delete object mask, the system displays a message indicating that object mask has been deleted and restart is required.

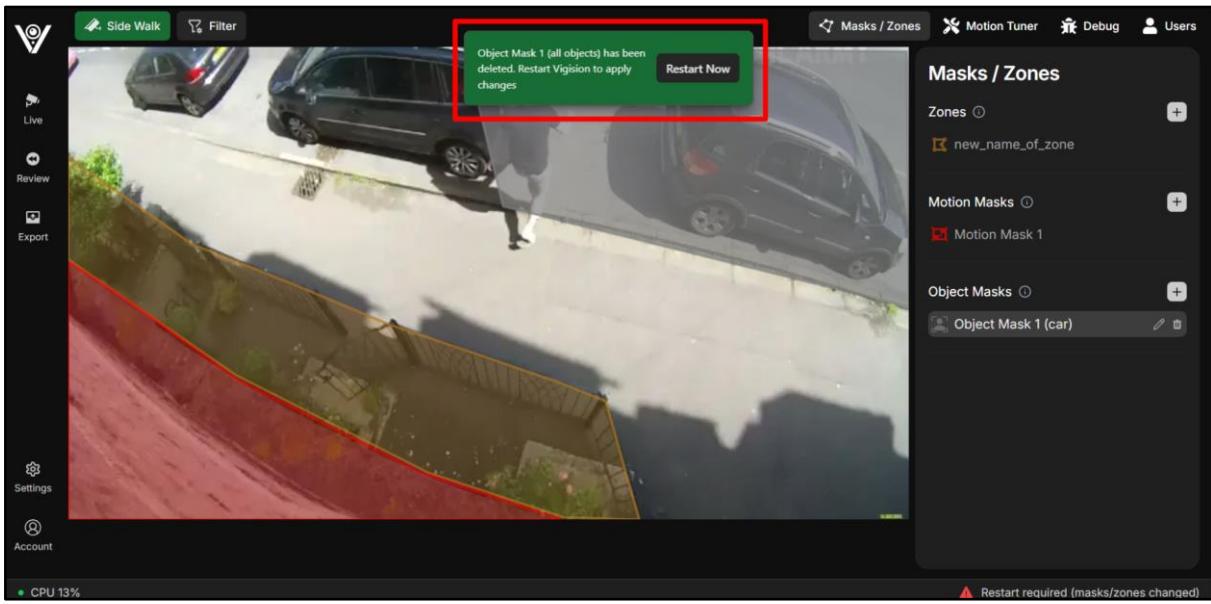


Figure 466. Manage object masks - Delete object mask

The image below shows that object mask with **all object** has been deleted. Note that **Object Mask 1 (car)** is renamed from **Object Mask 2 (car)** right after the first object mask is deleted.

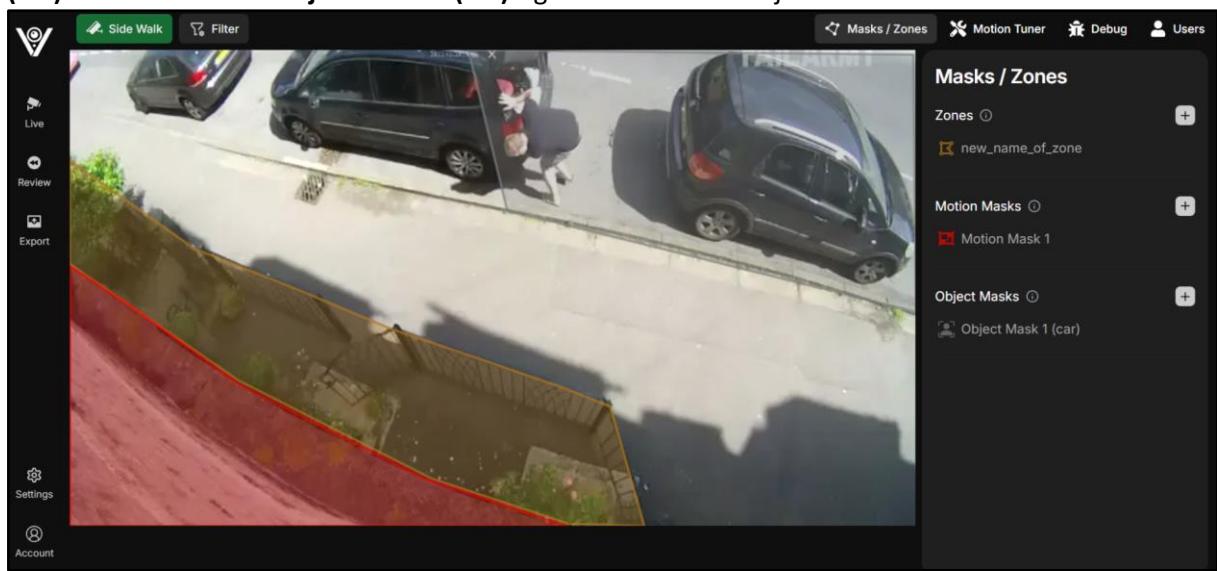


Figure 467. Manage object masks - Delete object mask

3.3.2.18 Fine-tune motion detection

To fine-tune motion detection, firstly, click on button “Settings” in the sidebar.

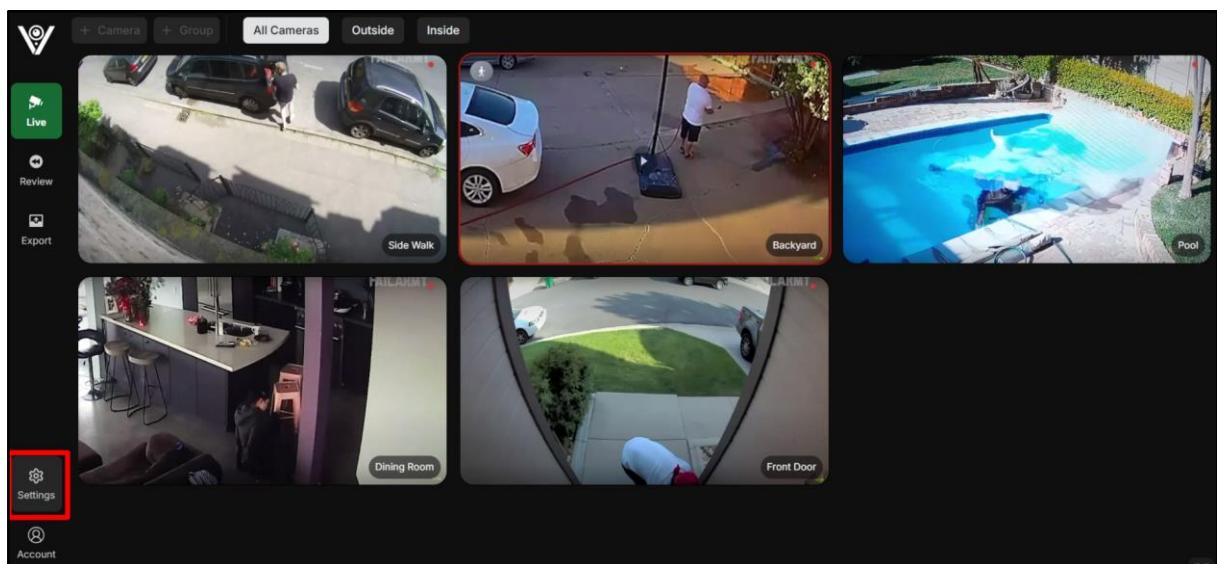


Figure 468. Fine-tune motion detection

Then, click “Settings”.

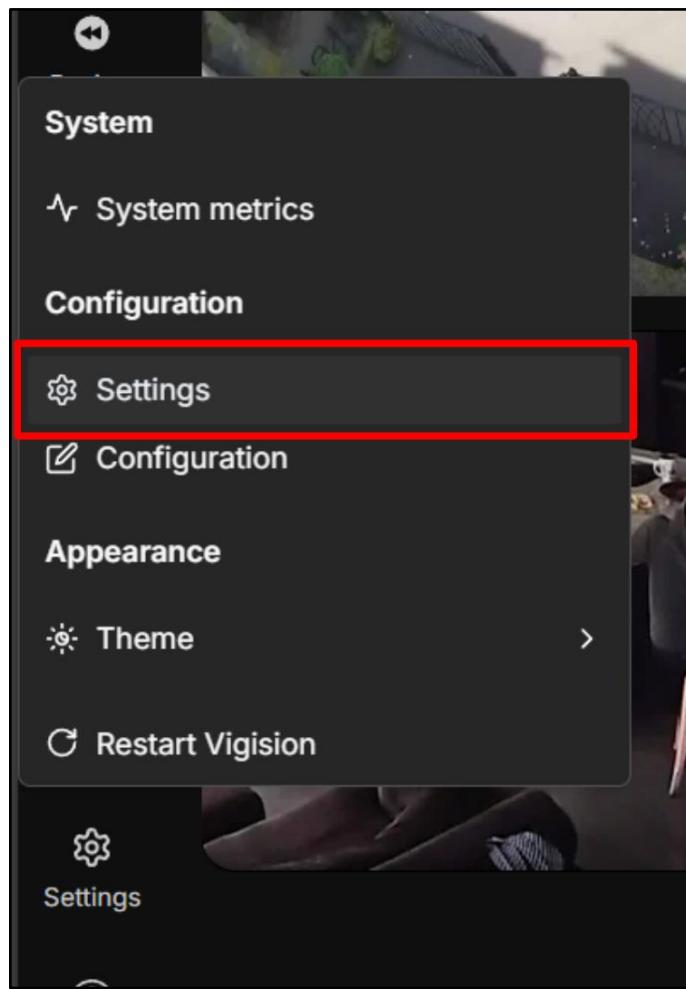


Figure 469. Fine-tune motion detection

Next, click on the button “Motion Tuner” on the top-right corner.

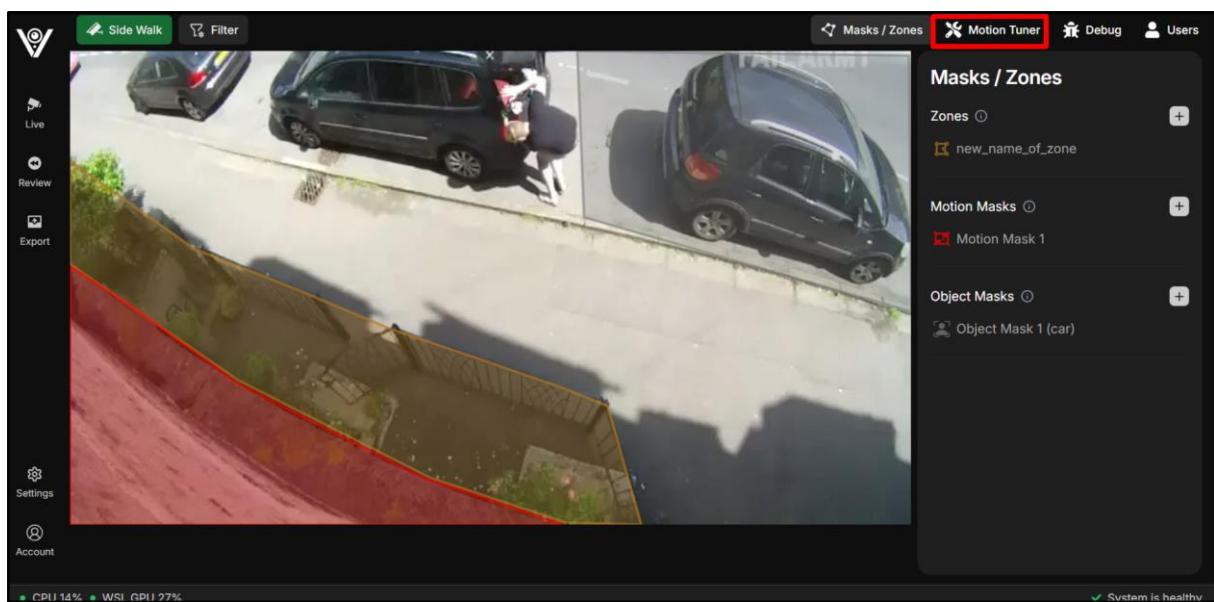


Figure 470. Fine-tune motion detection

The Motion Tuner panel is shown on the left side of the screen.

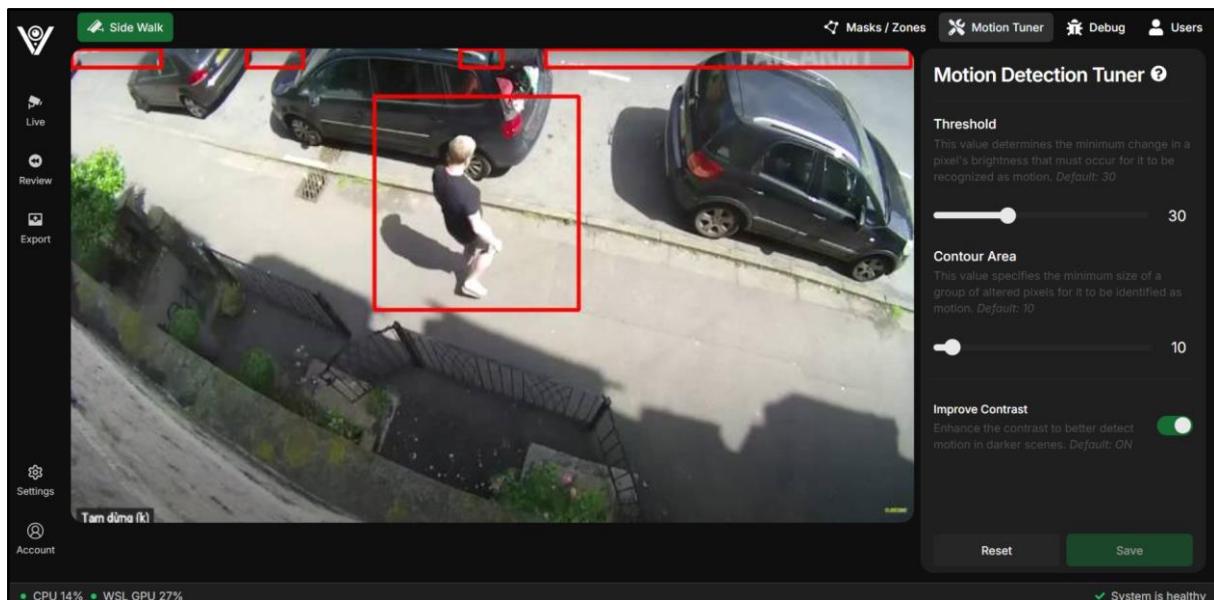


Figure 471. Fine-tune motion detection

Select the camera you want to fine tune.

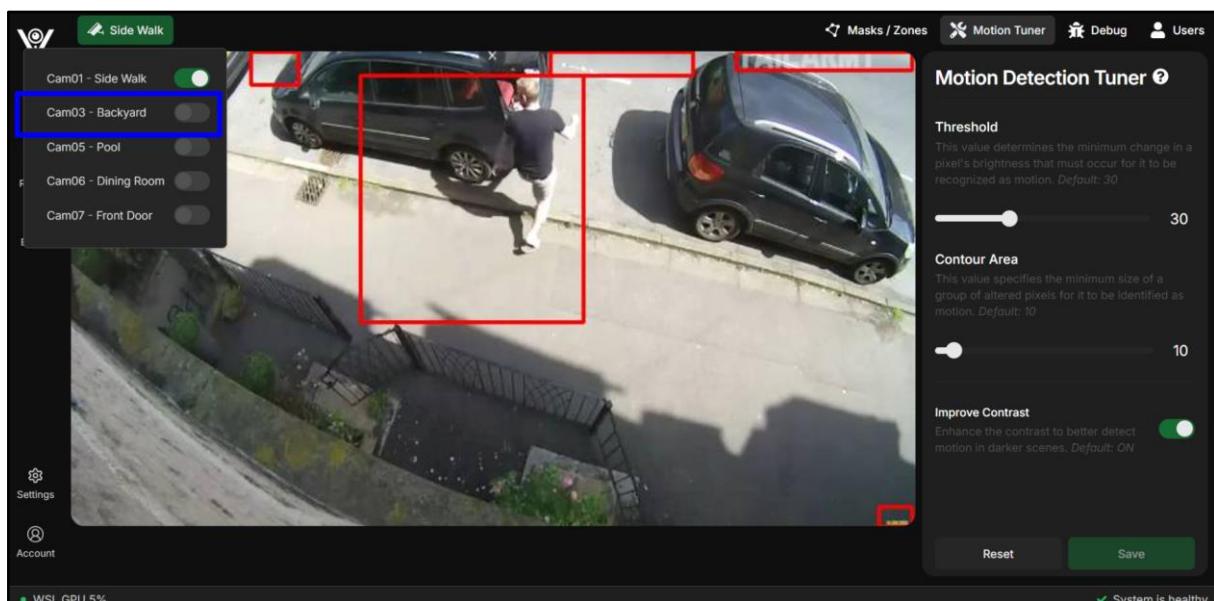


Figure 472. Fine-tune motion detection

Now, the selected camera livestream is shown.



Figure 473. Fine-tune motion detection

Now, start tuning threshold by sliding the threshold bar

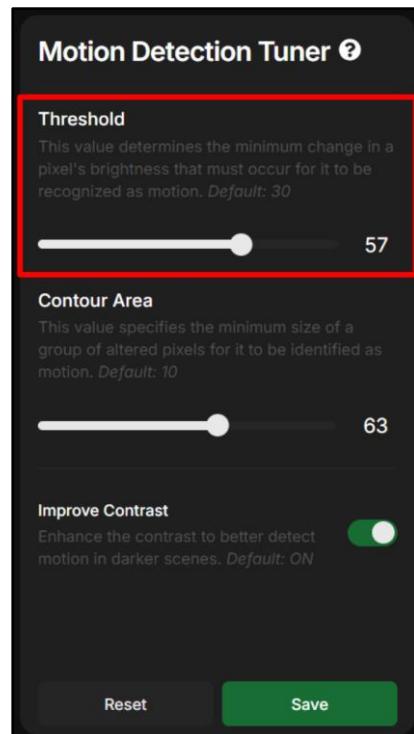


Figure 474. Fine-tune motion detection

Similarly, slide contour area bar to fine tune contour area. Then, you can also toggle on/off **Improve Contrast**.

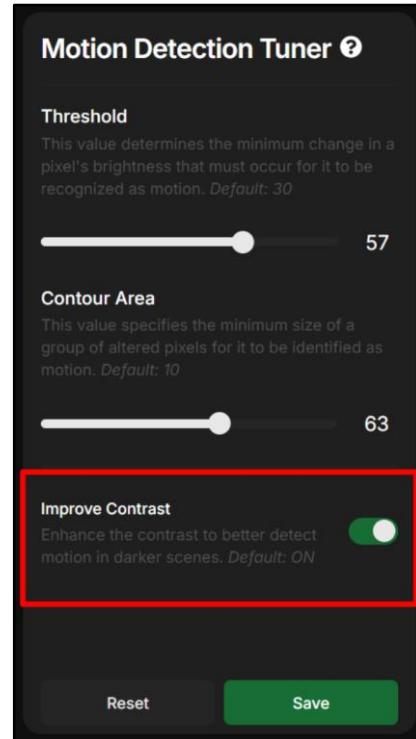


Figure 475. Fine-tune motion detection

If you want to reset as its previous state, simply click on the button “Reset”

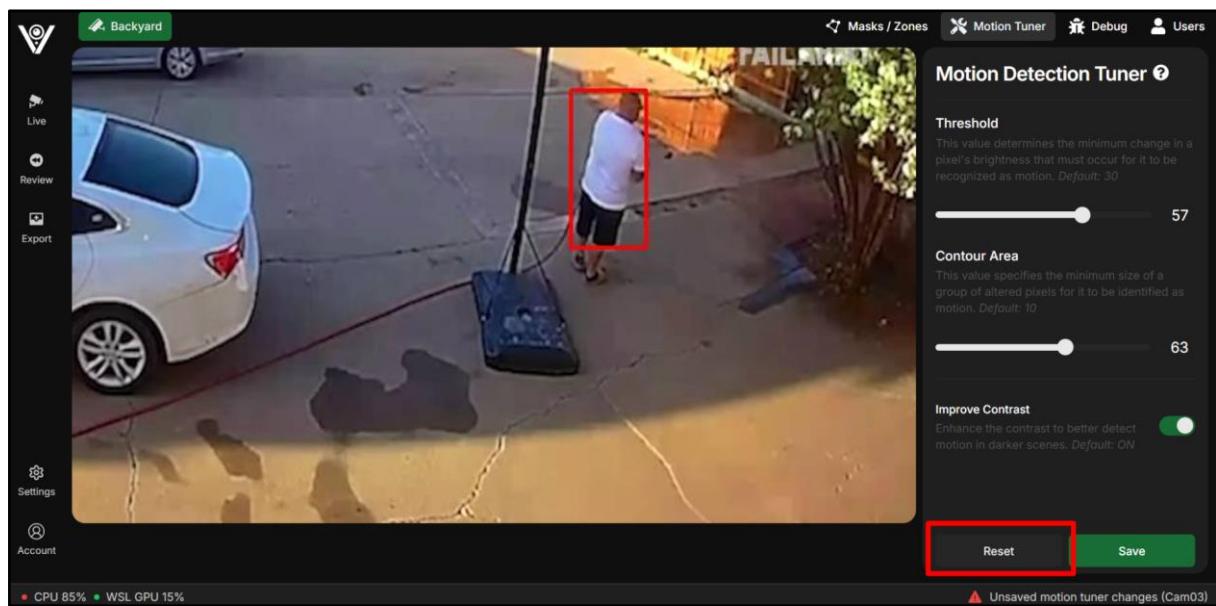


Figure 476. Fine-tune motion detection

As you see below, previous state is restored.



Figure 477. Fine-tune motion detection

If you want to save new motion detection tuner, click on the button “Save” at the bottom.



Figure 478. Fine-tune motion detection

Then, the system send a message informing you that motion setting has been saved.

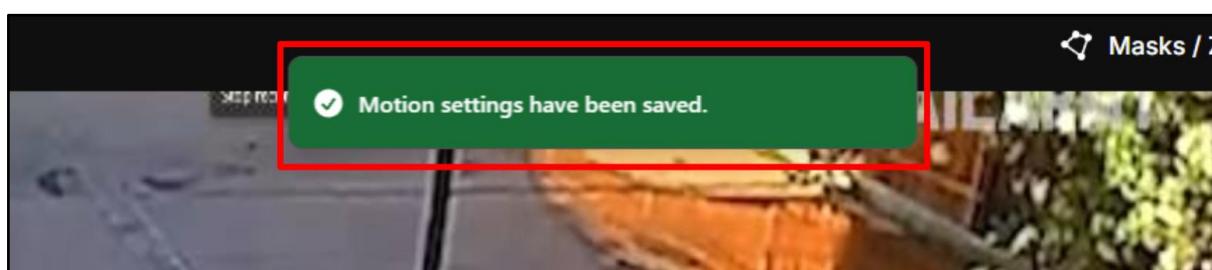


Figure 479. Fine-tune motion detection

Save successfully.



Figure 480. Fine-tune motion detection

3.3.2.19 Debug

- Step 1: On Live Dashboard page, click on “Settings”.

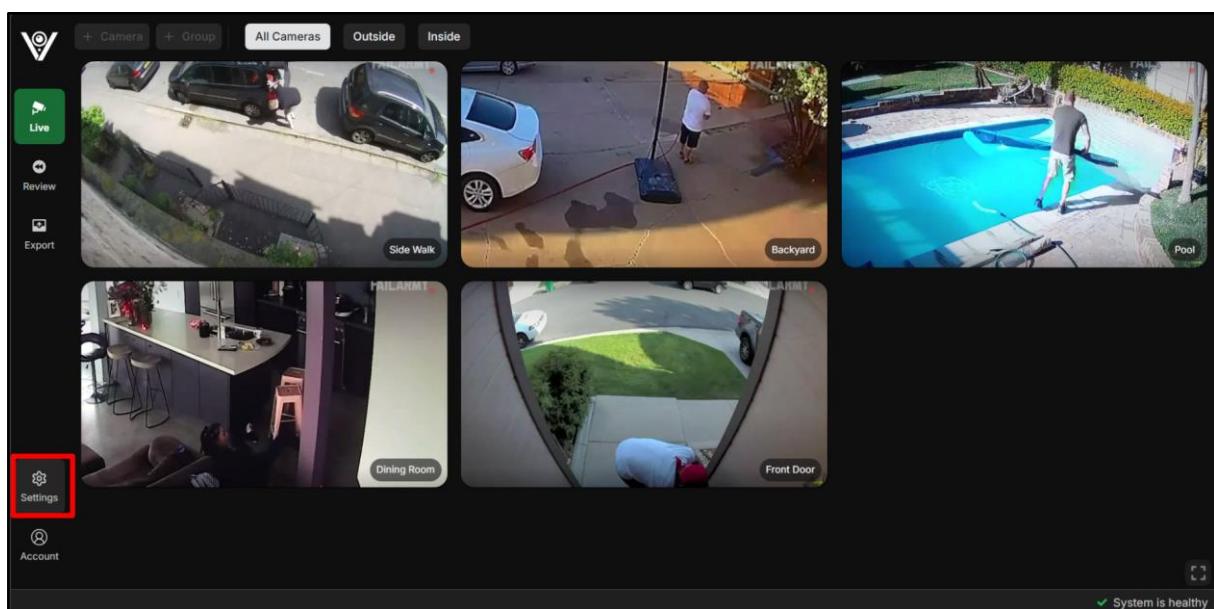


Figure 481. Debug

- Step 2: Click on “Settings”

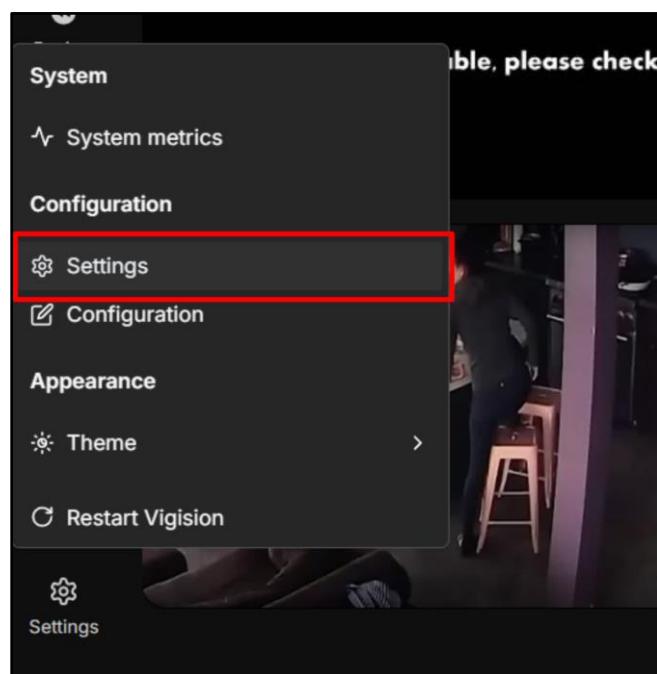


Figure 482. Debug

- Step 3: Click on “Debug” icon to view Debug tab.

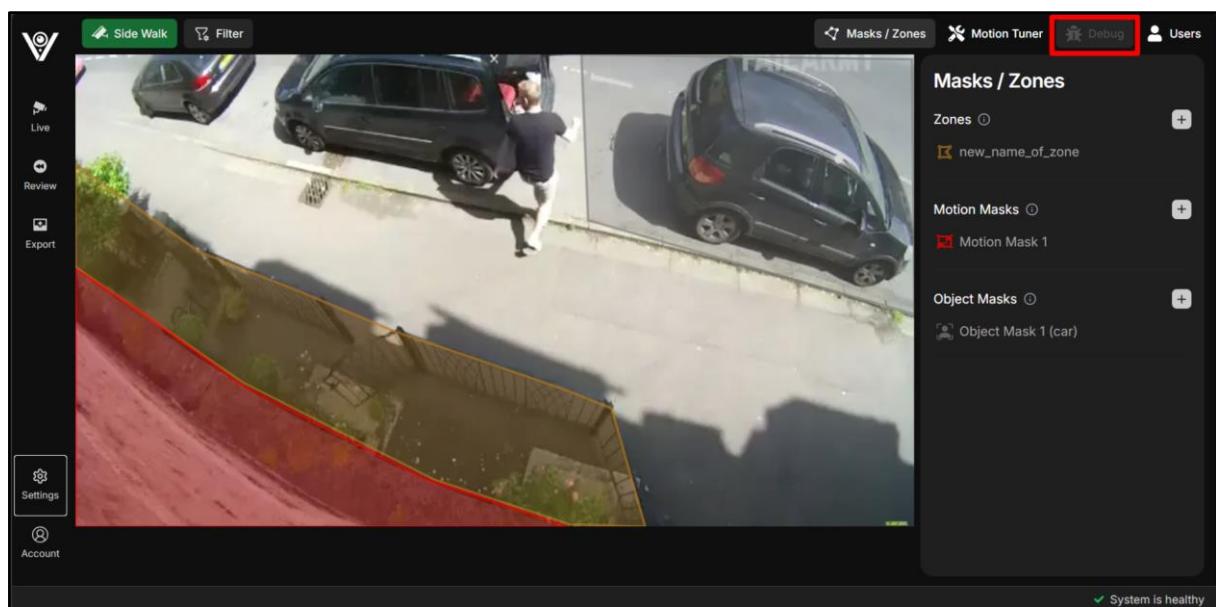


Figure 483. Debug

- Step 4: Click green button to change to the camera you want to debug for.

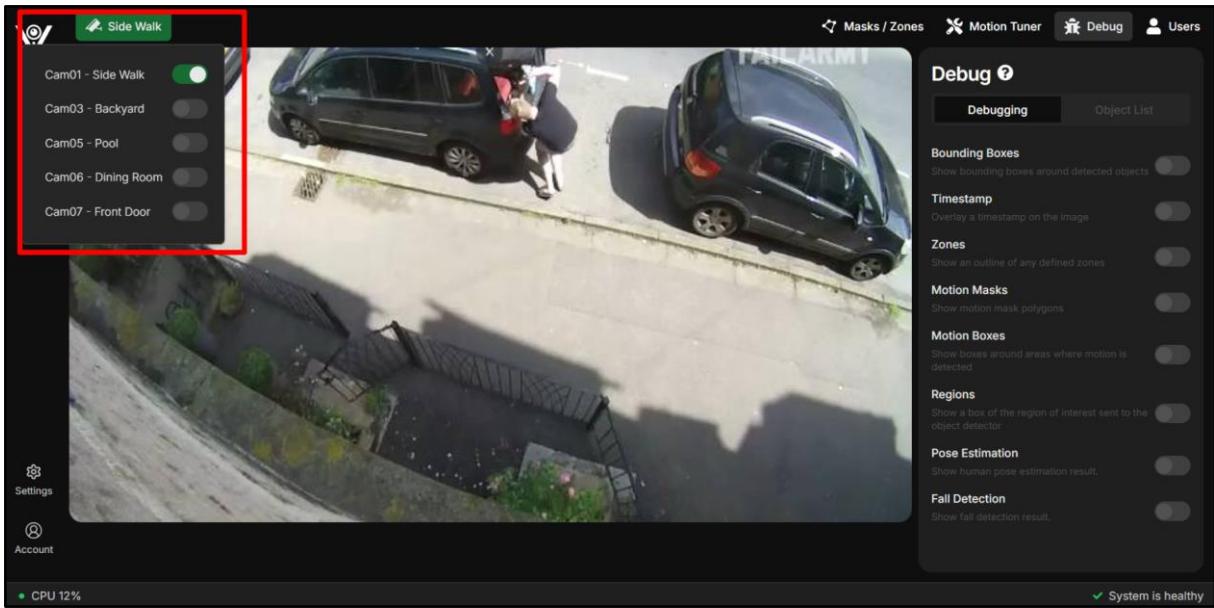


Figure 484. Debug

- Step 5: Click on “Cam03 - Backyard” (or any camera you want)

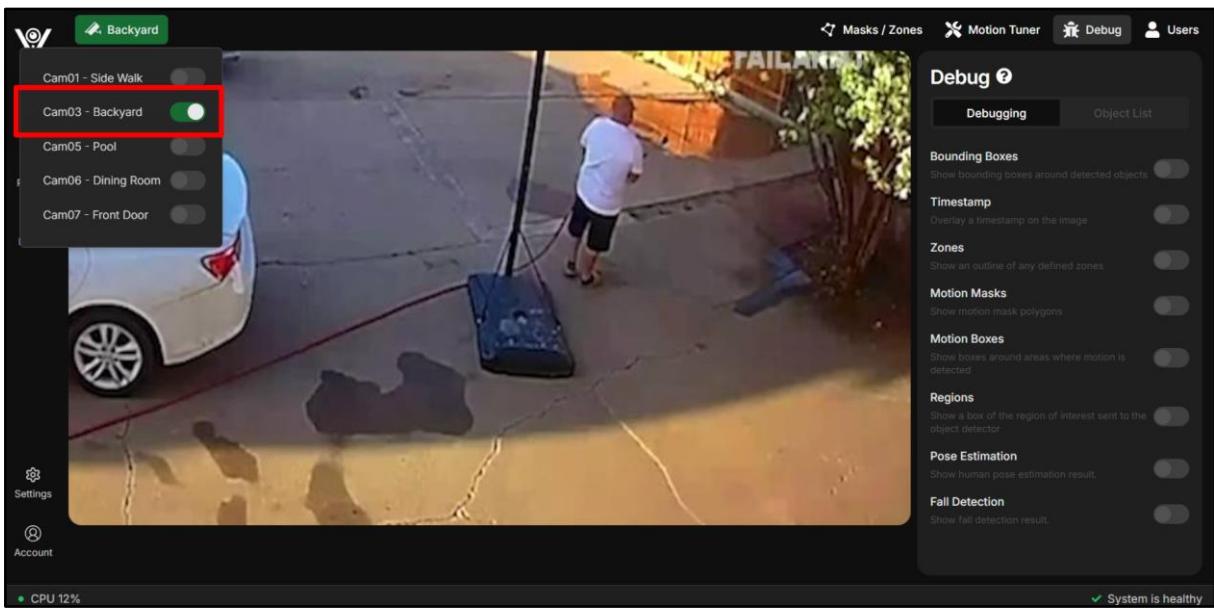


Figure 485. Debug

- Step 6: Start to debug

Case 1: Enable “Bounding Boxes” to view bounding boxes



Figure 486. Debug

Case 2: Enable “Timestamp” to view timestamp

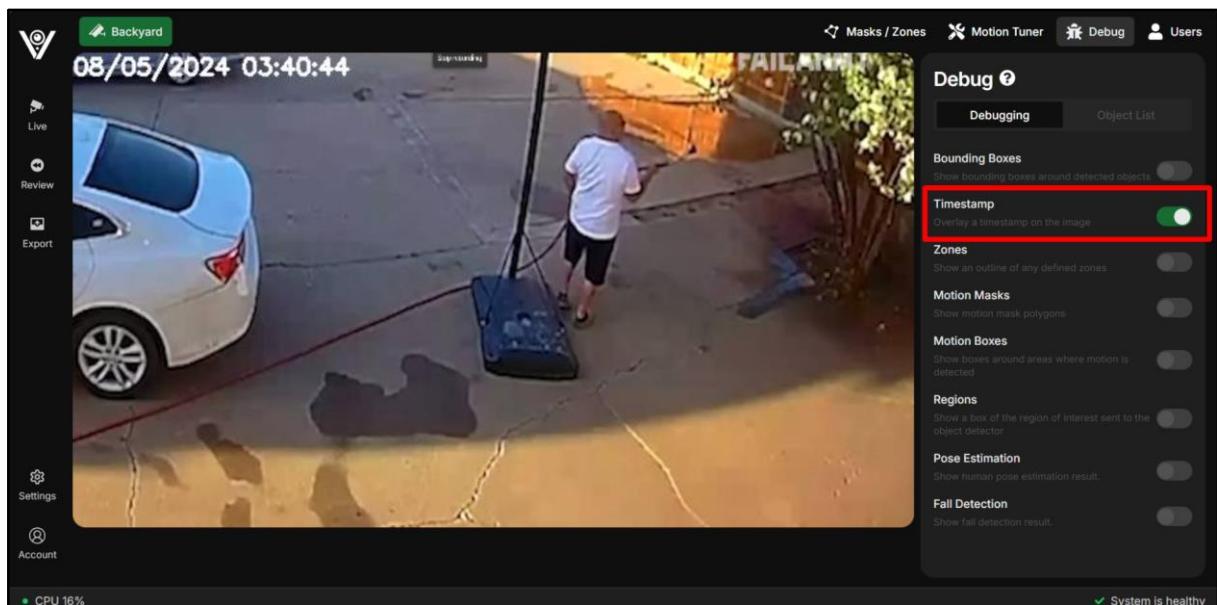


Figure 487. Debug

Case 3: Enable “Zones” to view zones

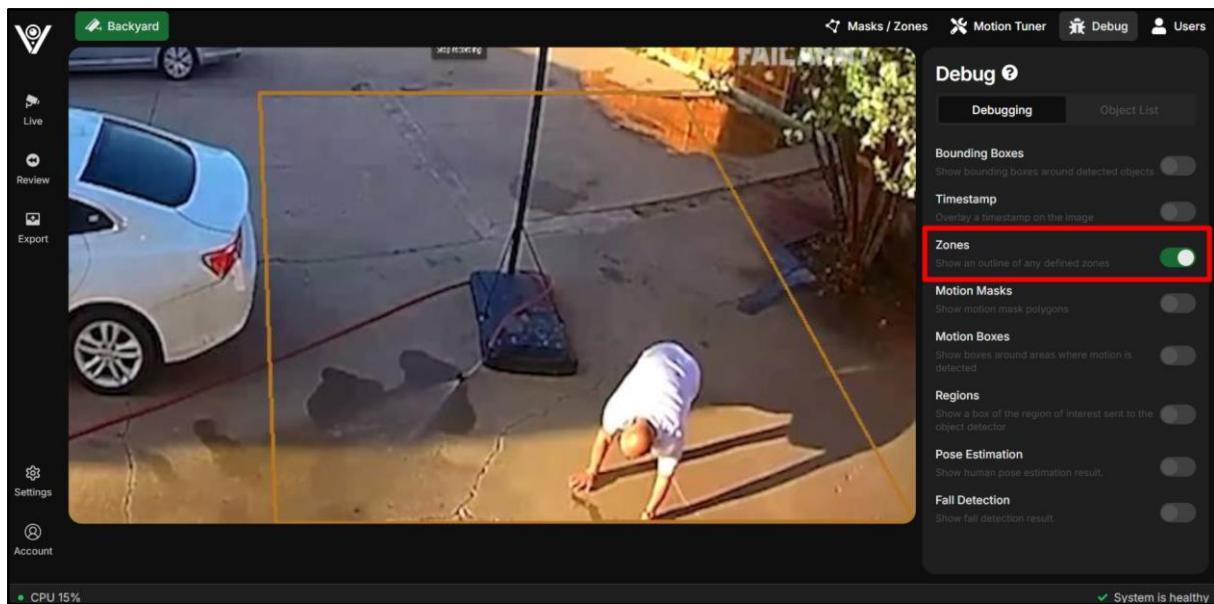


Figure 488. Debug

Case 4: Enable “Motion Masks” to view motion masks

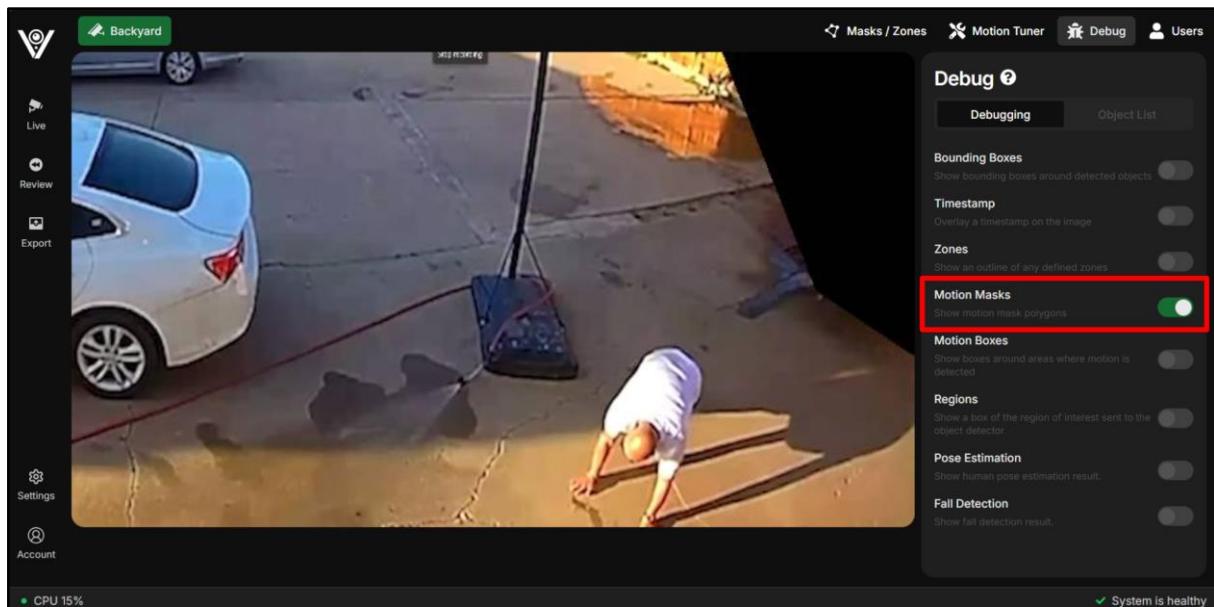


Figure 489. Debug

Case 5: Enable “Motion Boxes” to view motion boxes.



Figure 490. Debug

Case 6: Enable “Regions” to view regions.



Figure 491. Debug

Case 7: Enable “Pose Estimation” to view pose estimation.



Figure 492. Debug

Case 8: Enable “Fall Detection” to view fall detection.

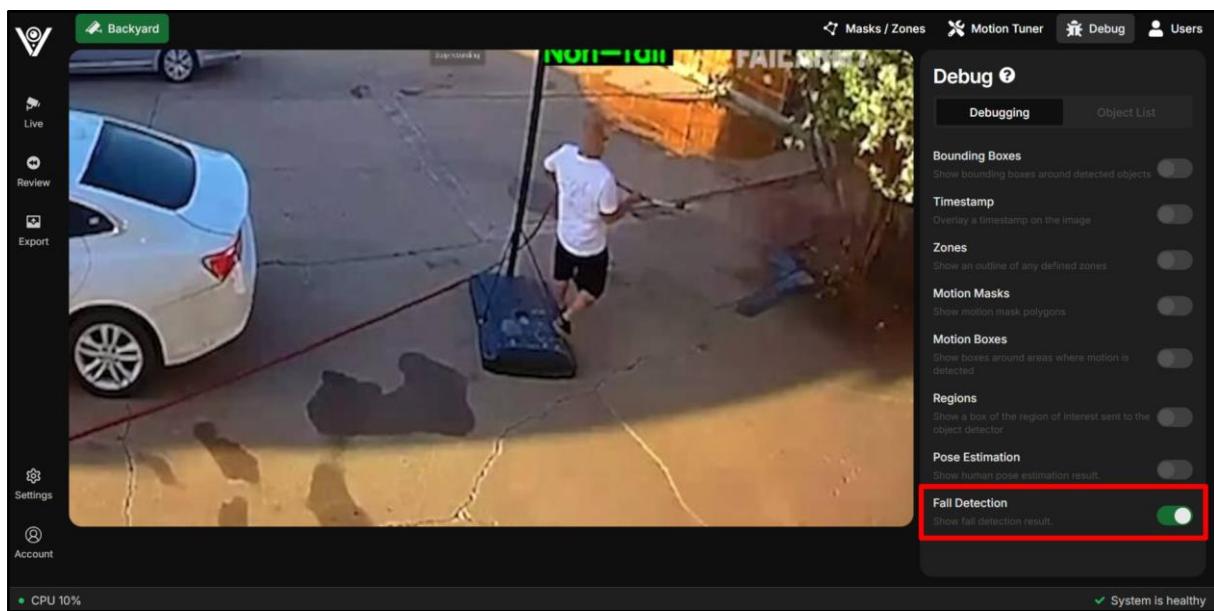


Figure 493. Debug

Case 9: Enable “Bounding Boxes”, “Pose Estimation”, “Fall Detection” to view bounding boxes, pose estimation, fall detection.



Figure 494. Debug

You can open “Object List” to see the object Person has been detected.



Figure 495. Debug

3.3.2.20 View alerts/detections

To view alerts/detections, the first step is to set up: enable record, detection, and select some objects to track (in this example: person, car). Remember to click “Save” and restart to apply changes.

Note: You can check “Update camera” feature to know how to set up camera.

Below image is an example of a camera set up for alerts/detections.

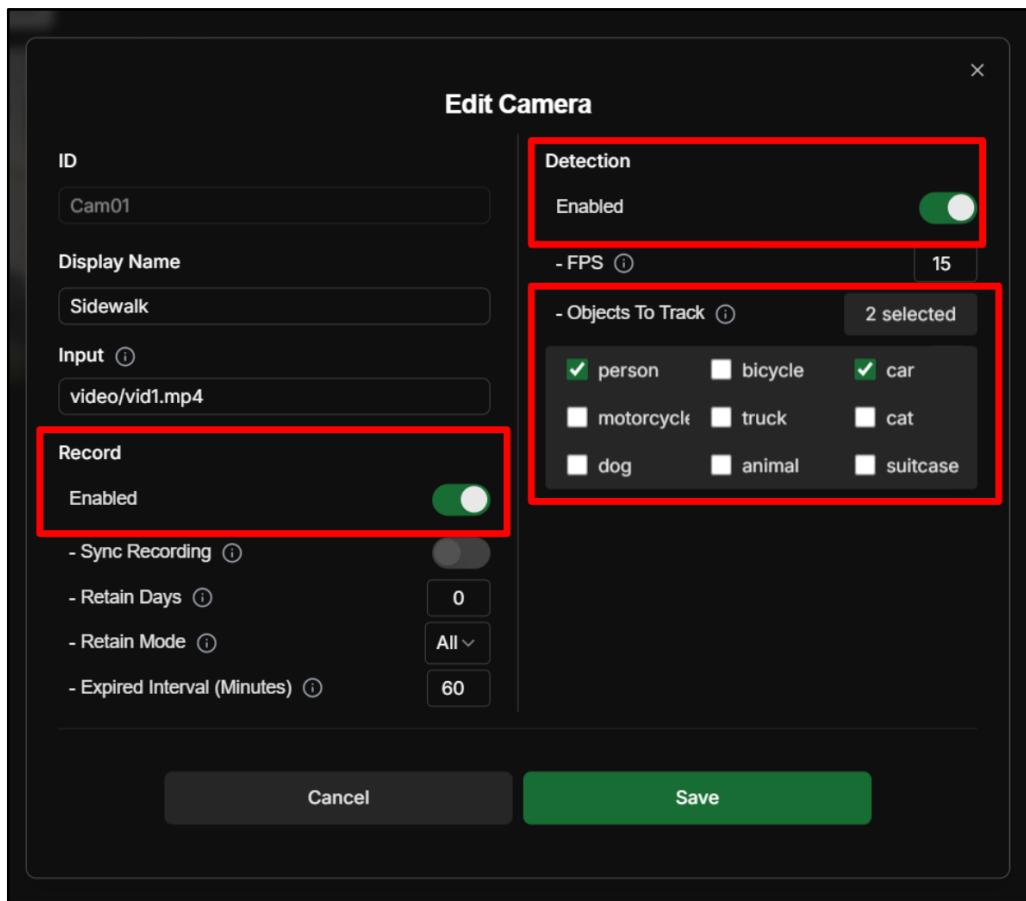


Figure 496. View alerts/detections

Detail steps for function View alerts/detections:

- Step 1: On Live Dashboard page, click on “Review”.

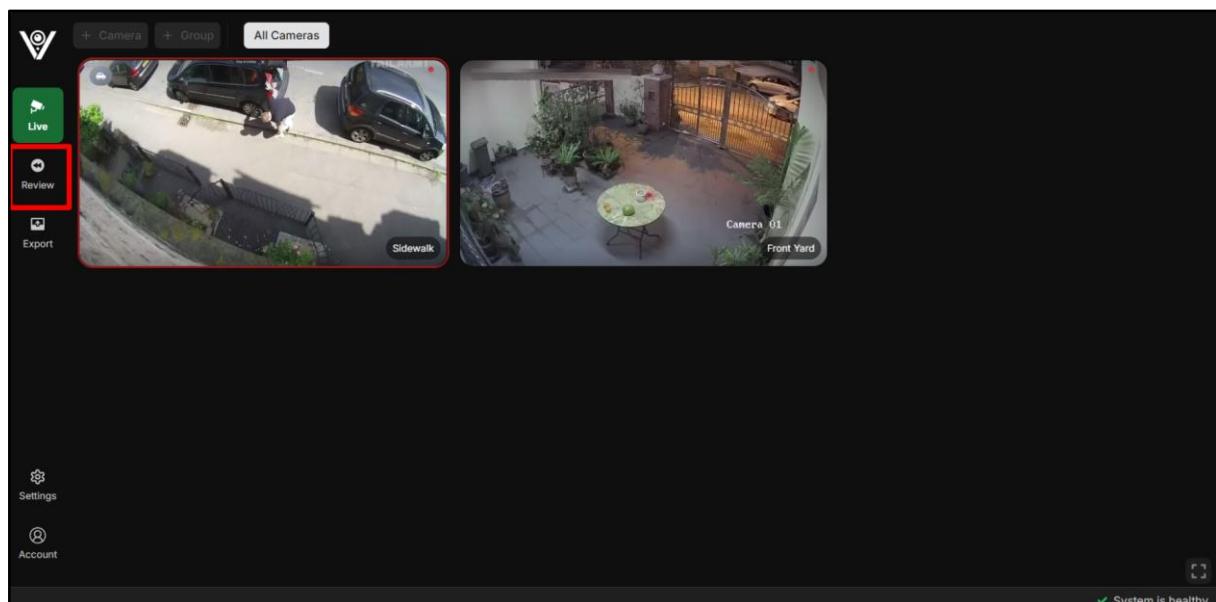


Figure 497. View alerts/detections

Assume that there is an people enter the frame, an alert is recorded. Then, you can see an alert appears as shown below.

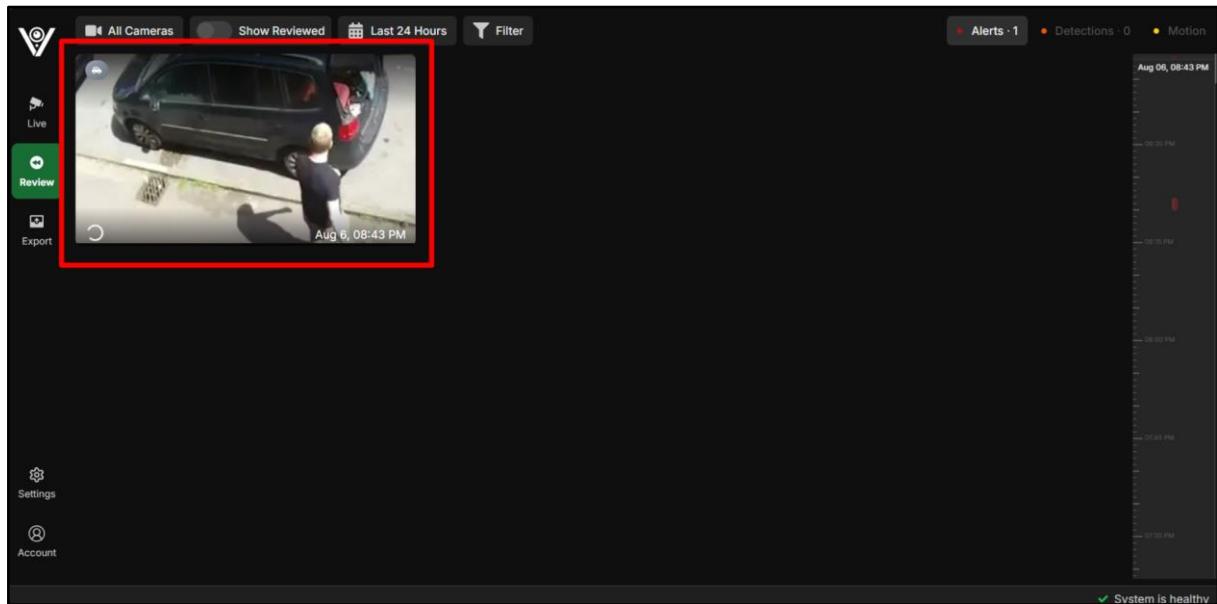


Figure 498. View alerts/detections

Simultaneously, an alert email is sent to user email.

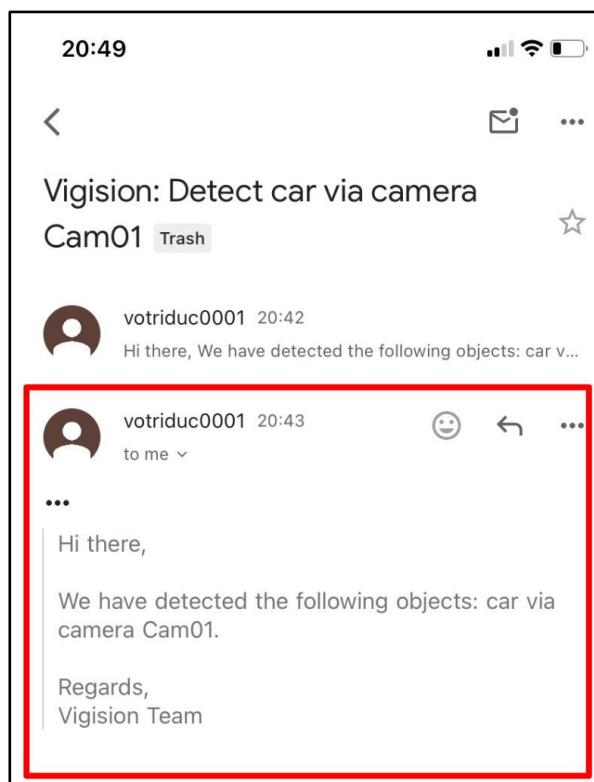


Figure 499. View alerts/detections

If you see a message “New Items To Review”, this means a new event is happening. Just click on it to refresh the page and new alerts/detections will appear.

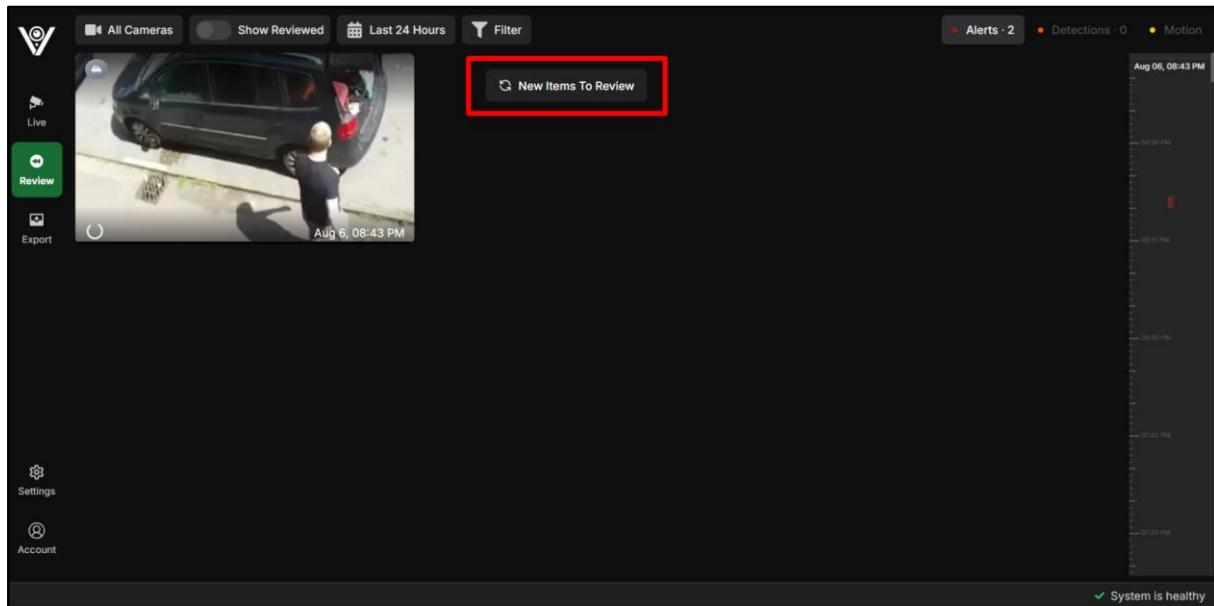


Figure 500. View alerts/detections

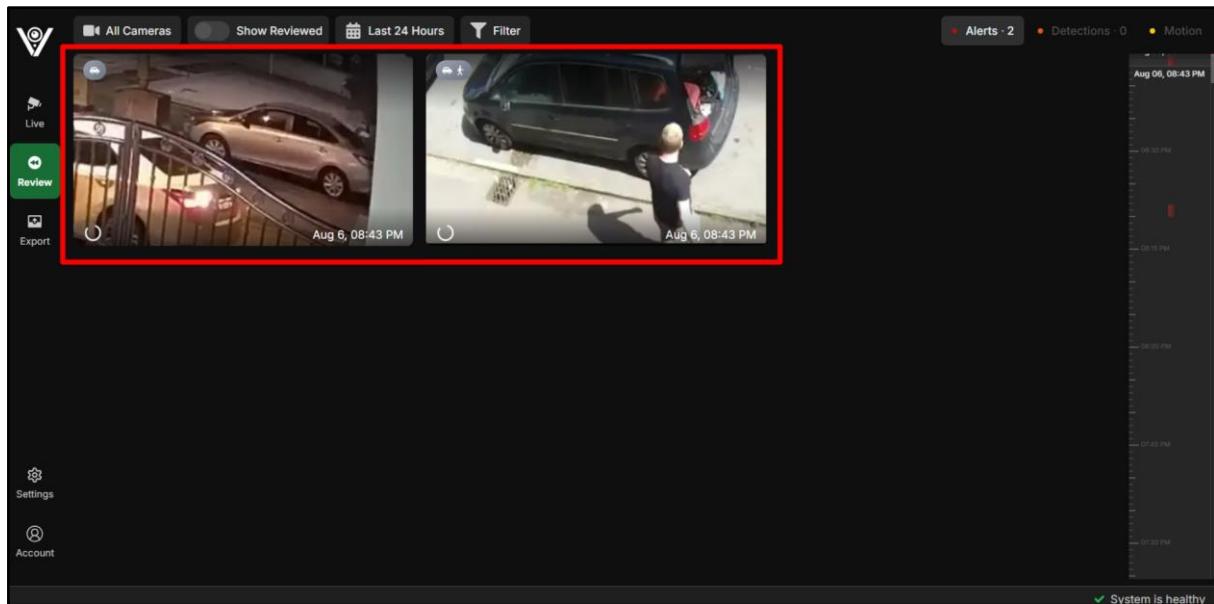


Figure 501. View alerts/detections

As mentioned above, when an event happens, an notification is sent to email of user.

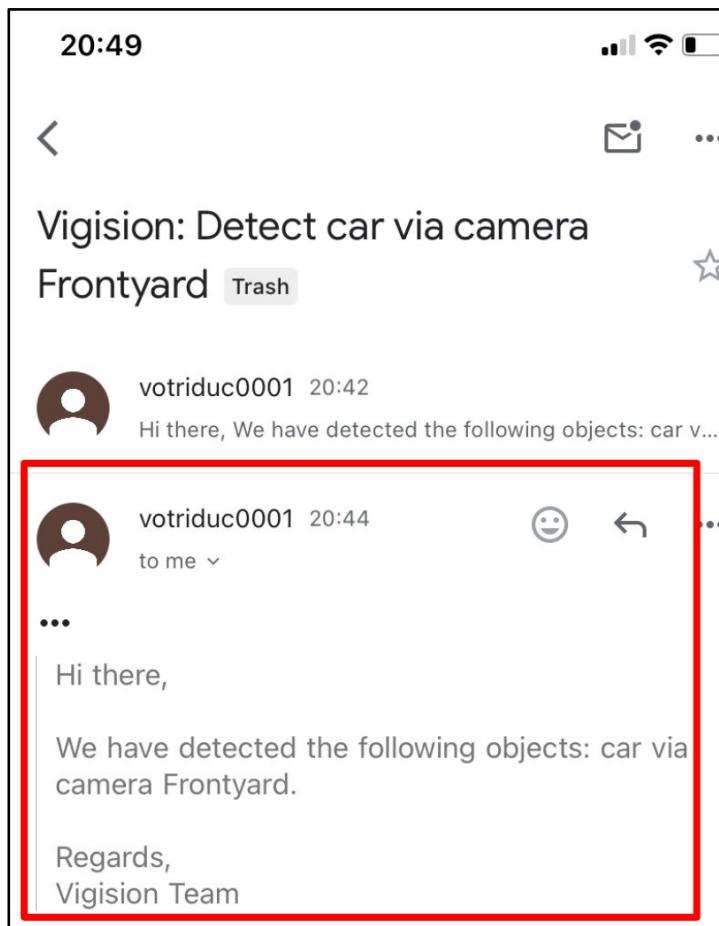


Figure 502. View alerts/detections

You can go to tab “Detections” to see detections. Currently there is no detections so the screen displays “There are no detections to review” - as in the image below.

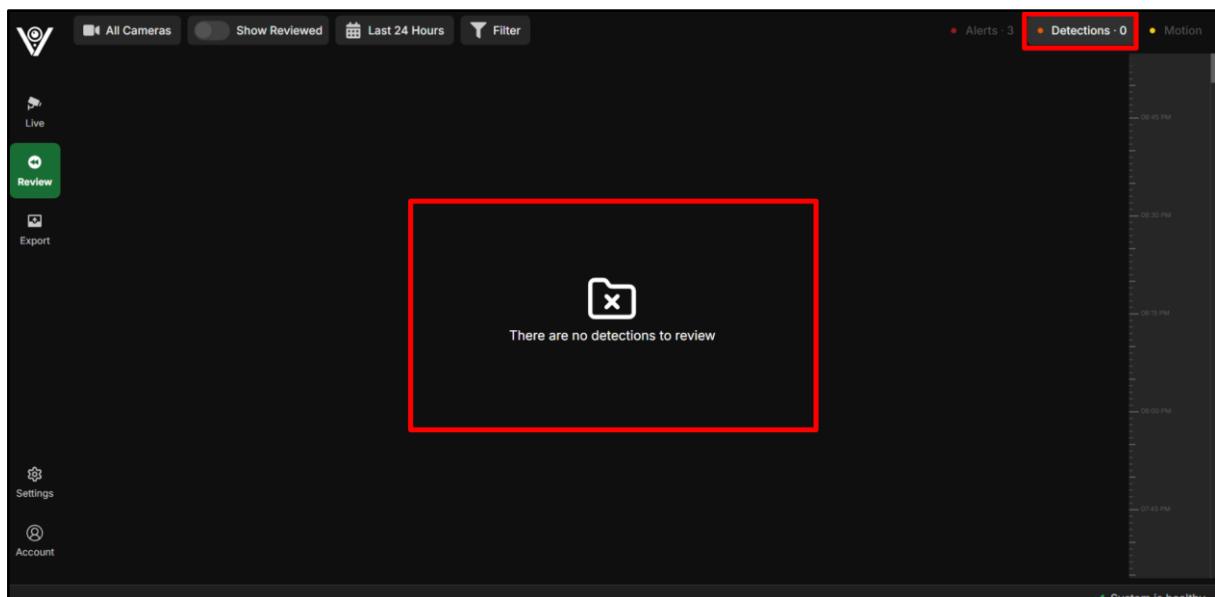


Figure 503. View alerts/detections

Now, go back to the Alerts tab by clicking on “Alerts”, you can see that 2 more new alerts have been recorded.

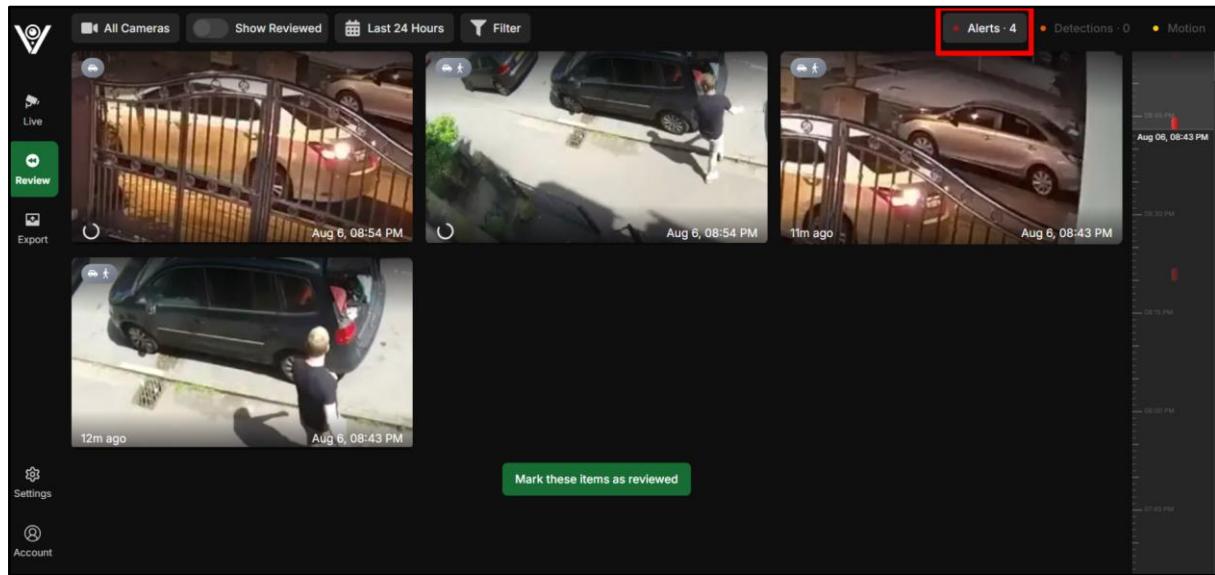


Figure 504. View alerts/detections

If you want to view alert of some specific cameras/a specific camera, click on the button at top-left corner (in this image its name is All Cameras).

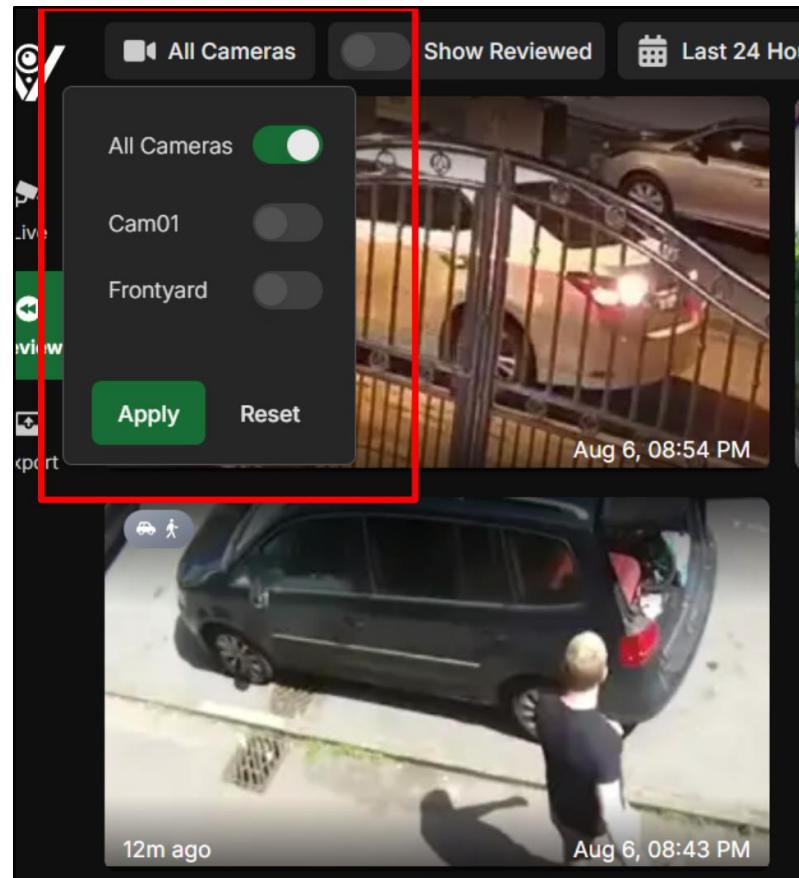


Figure 505. View alerts/detections

Then, a list of cameras is shown, you can click on the camera(s) you want to view events - as show in the image below. Finally, click “Apply” to view.

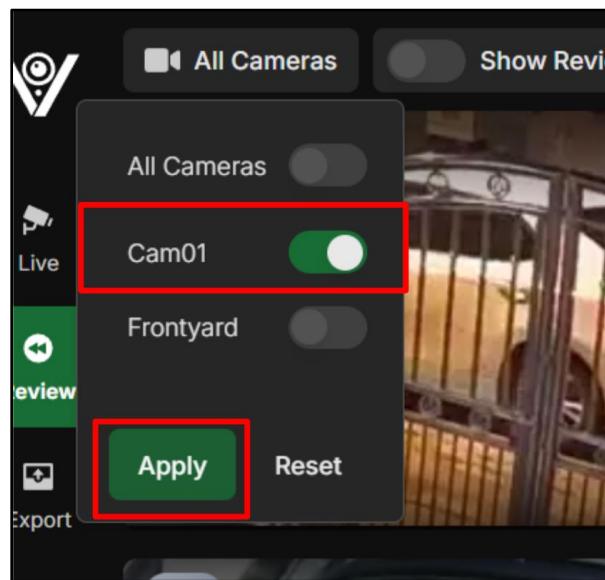


Figure 506. View alerts/detections

Now, you can see the alerts recorded by the selected cameras (example: “Cam01”).

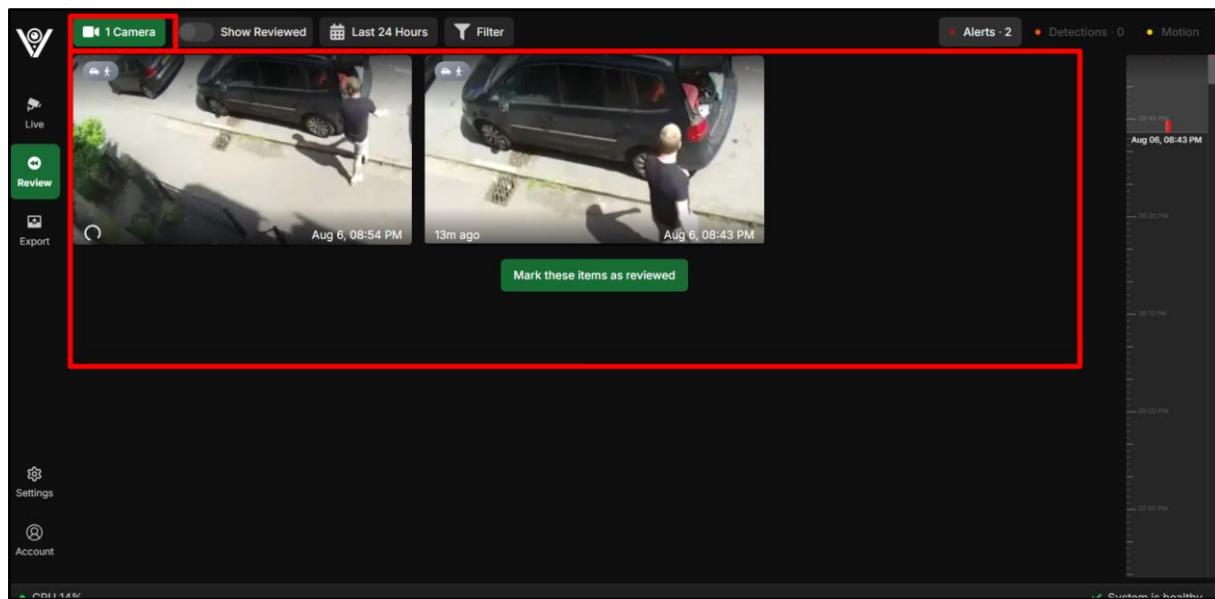


Figure 507. View alerts/detections

If you want to reset this filter to default, you can click button “Reset”.

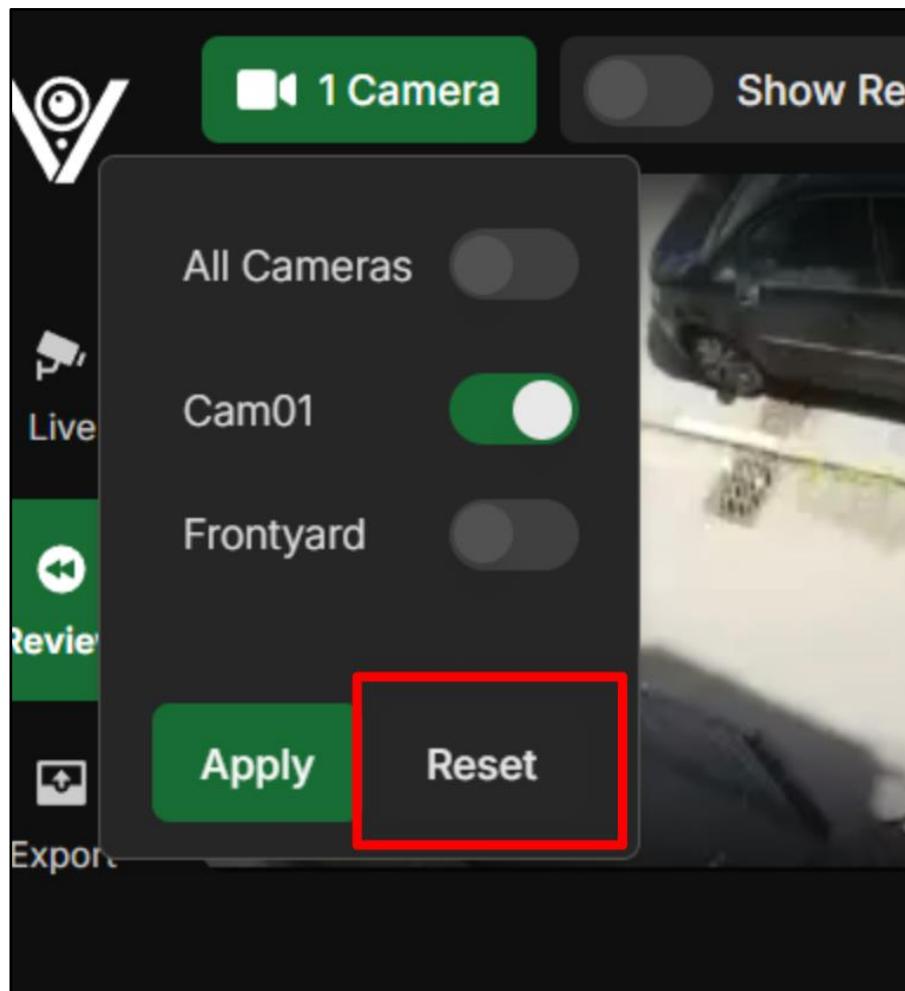


Figure 508. View alerts/detections

Then, the “Review” page returns to show event of all cameras again.

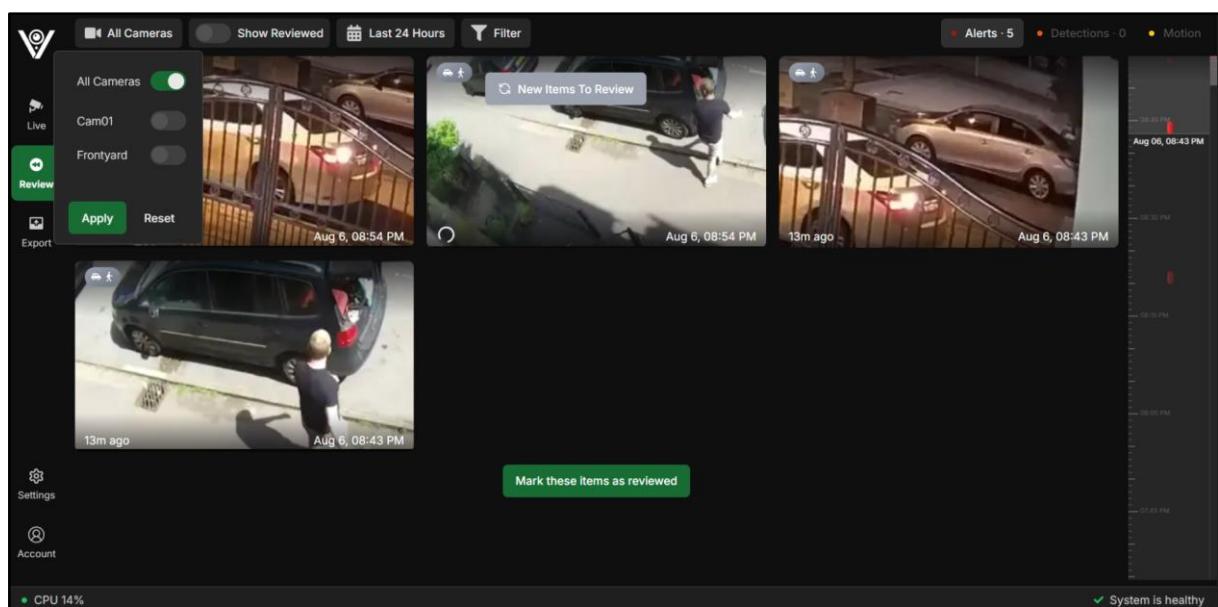


Figure 509. View alerts/detections

Next, click on button “Mark these items as reviewed”

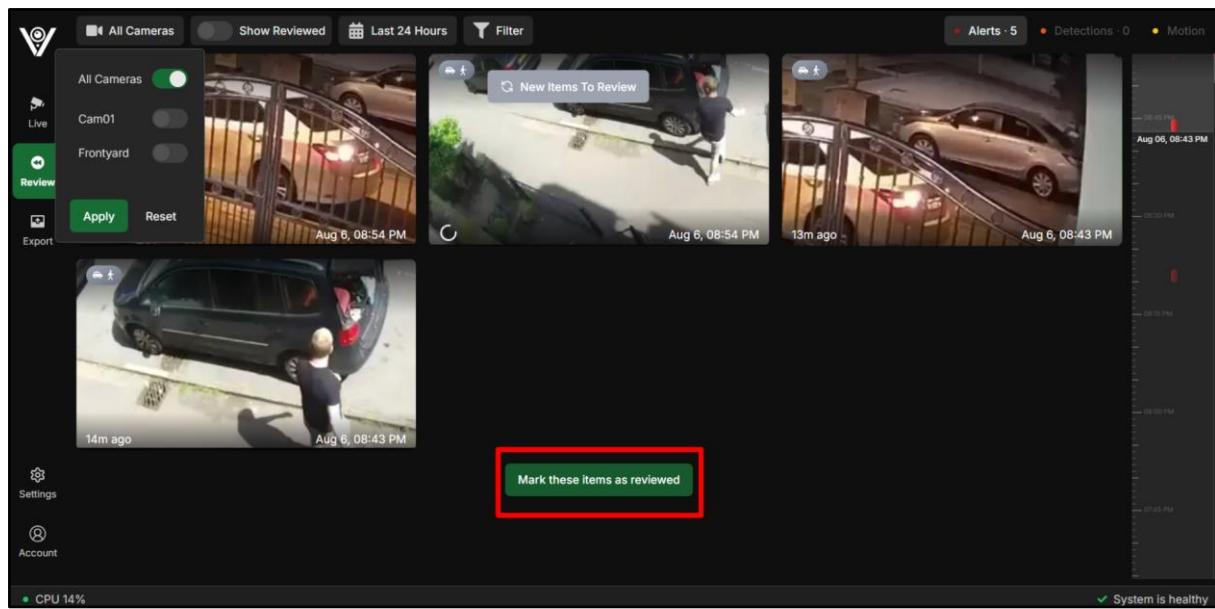


Figure 510. View alerts/detections

The result is current items on the screen is marked as reviewed and hidden - make place for upcoming items.

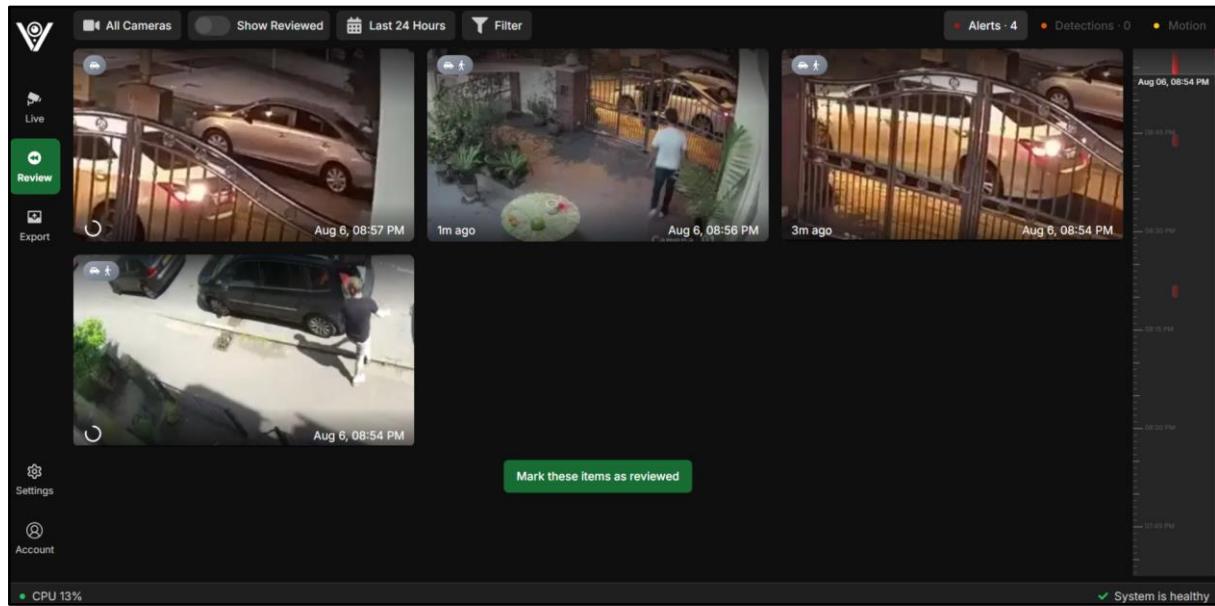


Figure 511. View alerts/detections

To view reviewed items, toggle on button “Show Reviewed”

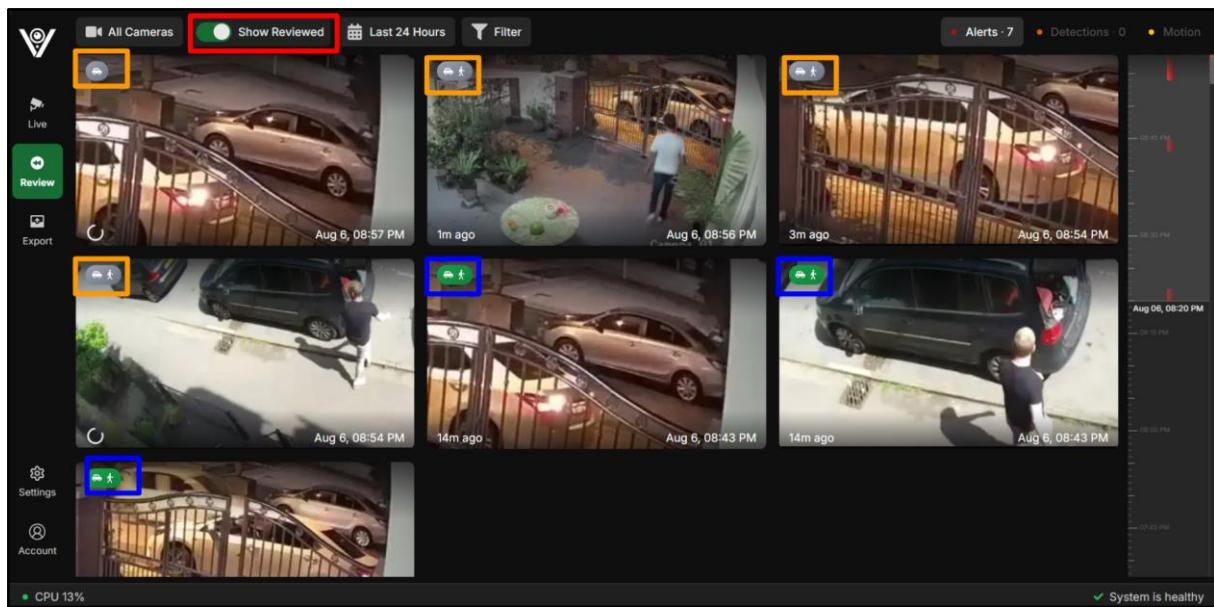


Figure 512. View alerts/detections

You can view items by date. A red round point below each date indicates there exists item(s) on that date. You can click on that date to see its items.

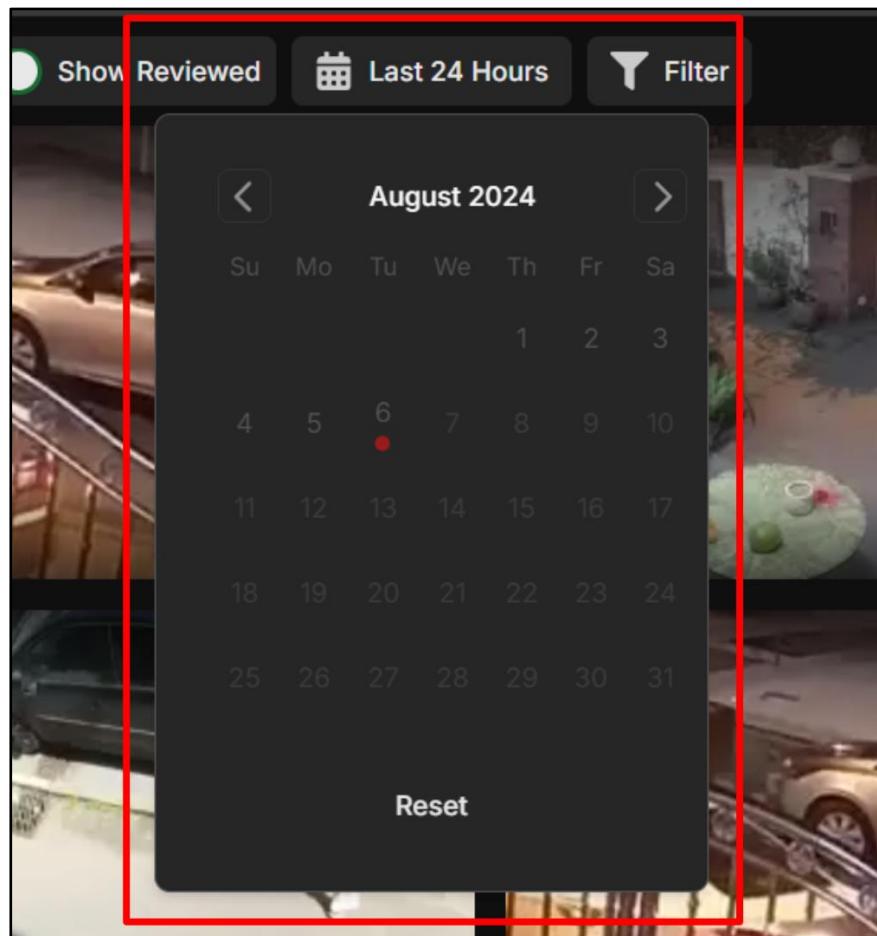


Figure 513. View alerts/detections

If you want to view items of only some objects, some zones, ... you can click on button “Filter” and set up your filter. Then, click “Apply” to finish.

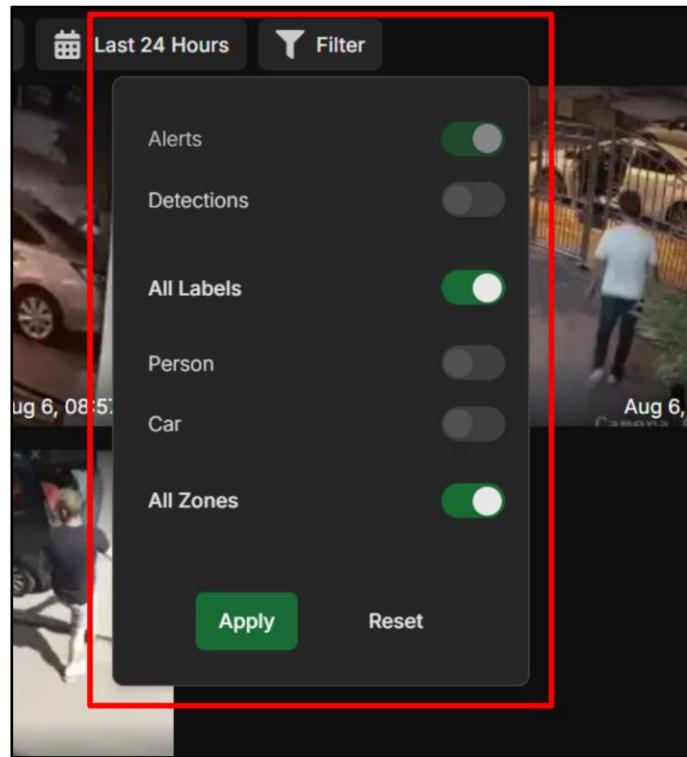


Figure 514. View alerts/detections

For example, if you want to filter events that have object car and all zones, you can set up as in the image below. Then, click “Apply”.

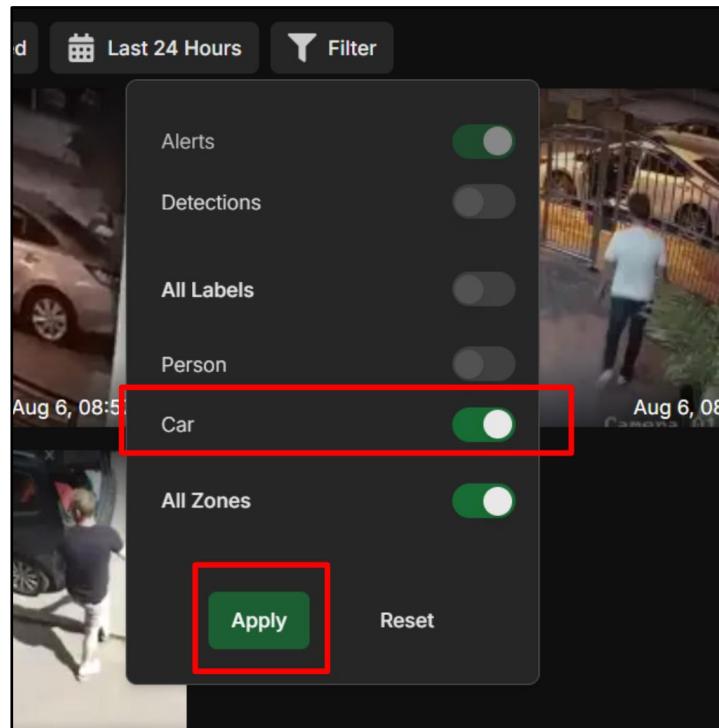


Figure 515. View alerts/detections

Here is the result when filter applied:

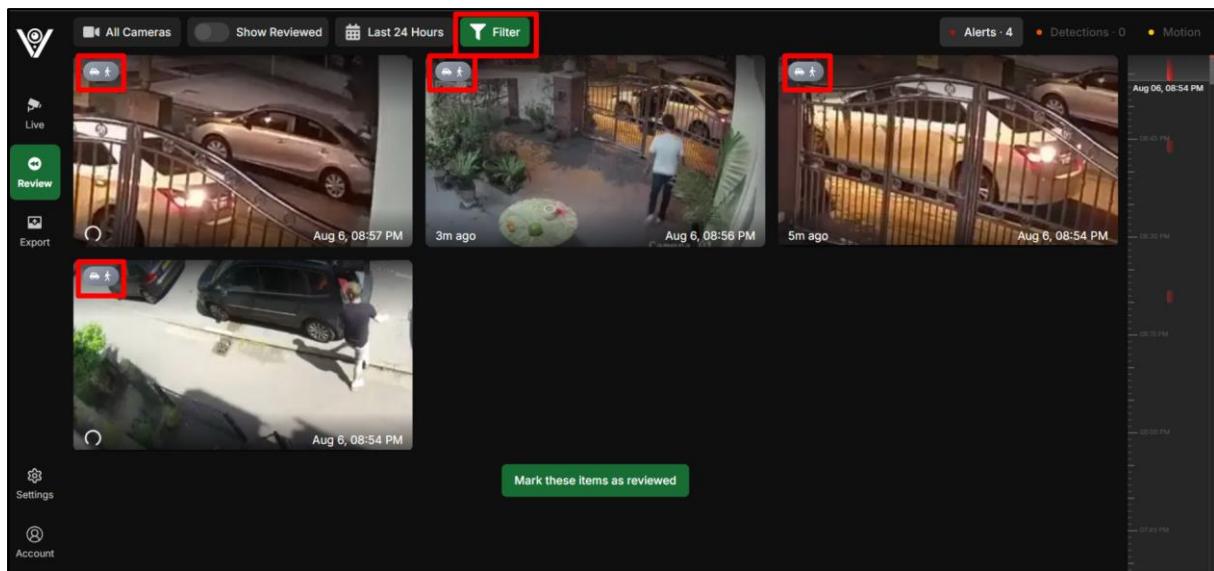


Figure 516. View alerts/detections

While hover on an item, you can watch the snapshot of that event.

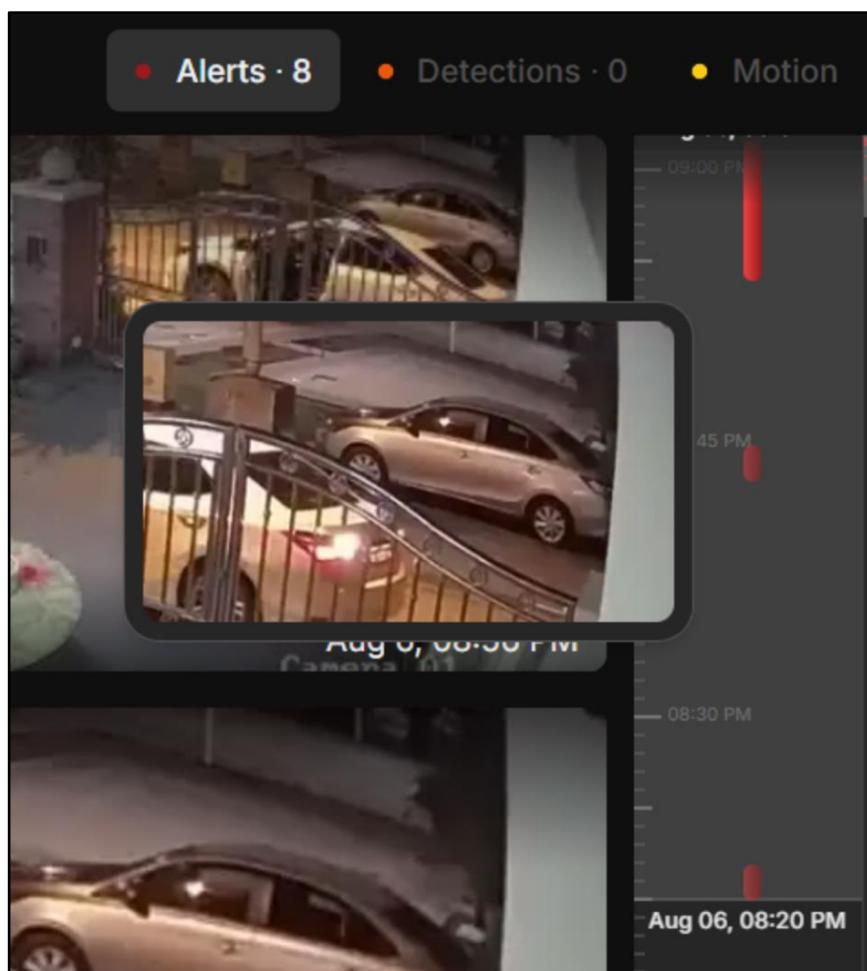


Figure 517. View alerts/detections

While hovering over an item, you can watch the preview of that event.

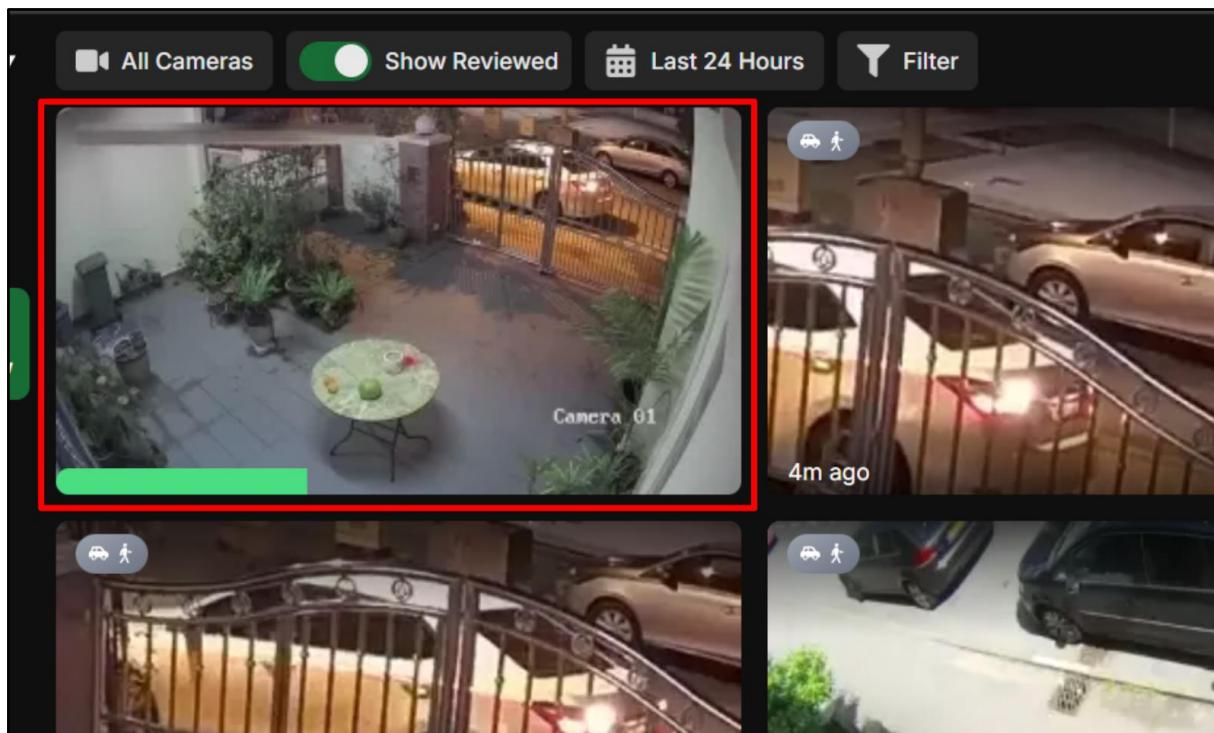


Figure 518. View alerts/detections

3.3.2.21 View motion events

On the “Review” page, in tab “Motion”, you can view some motions captured by the cameras.

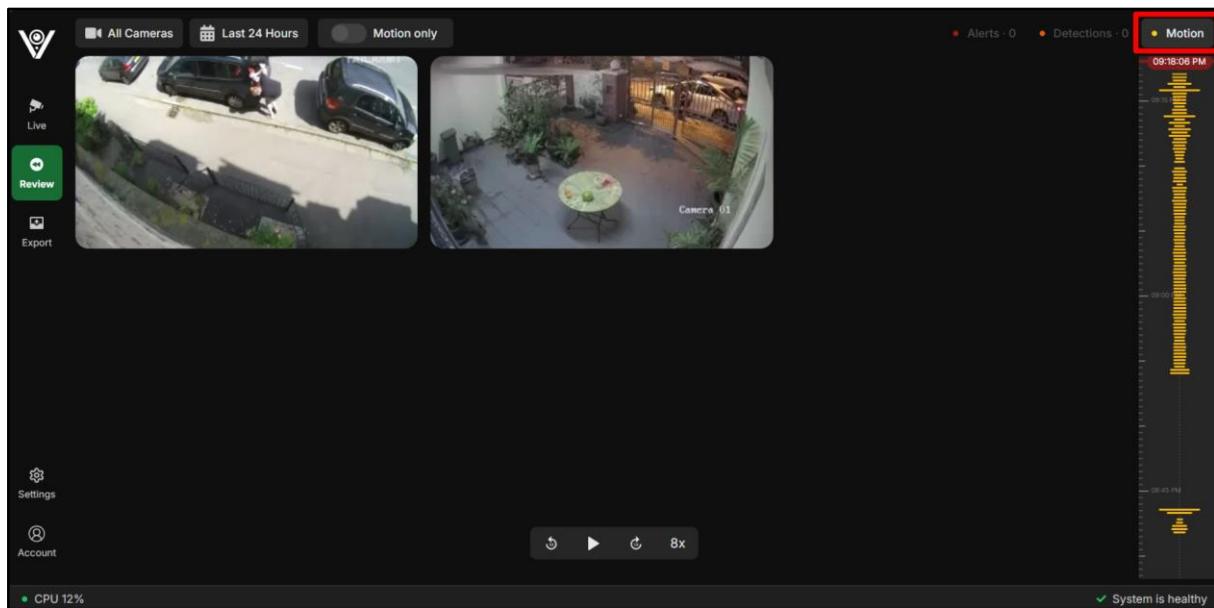


Figure 519. View motion events

You can view motion in different speed (4x, 8x, 12x, 16x), view by time.

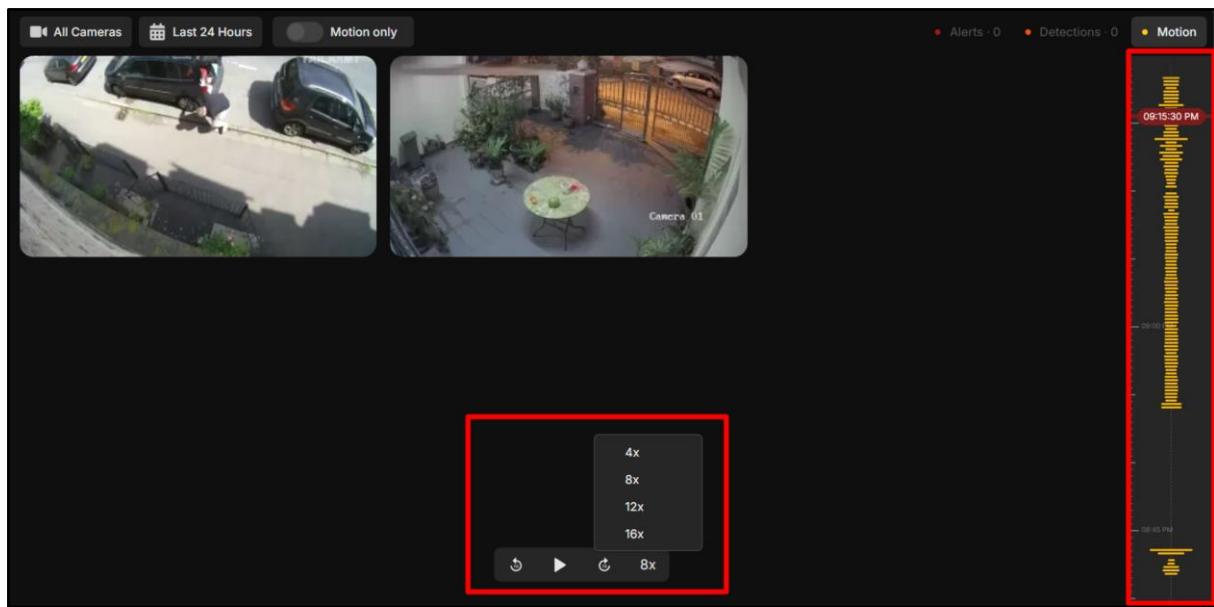


Figure 520. View motion events

When enable “Motion only” button - as in image below, it will automatically hide all non-motion frames from video. So that you can see the gaps in motion timeline disappears. This function make user scan all motions faster.

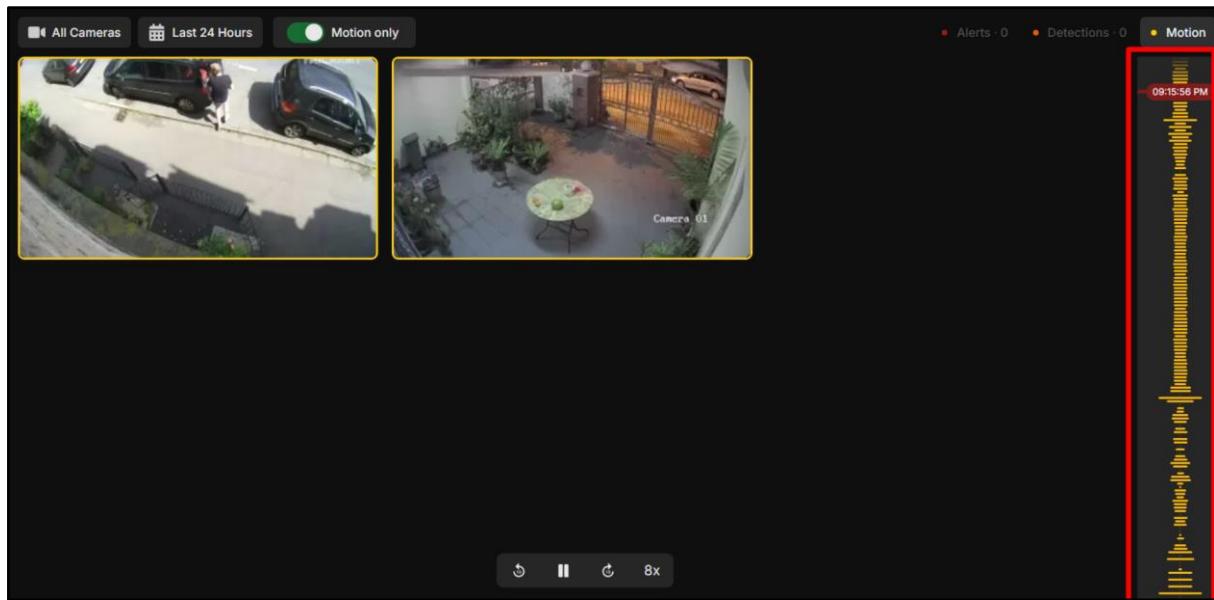


Figure 521. View motion events

3.3.2.22 View event recording

To view event recording, there are 3 ways for you

Way 1: Go to the “Review” page. You will see events listed on the screen. Click on a event thumbnail to view event recording.

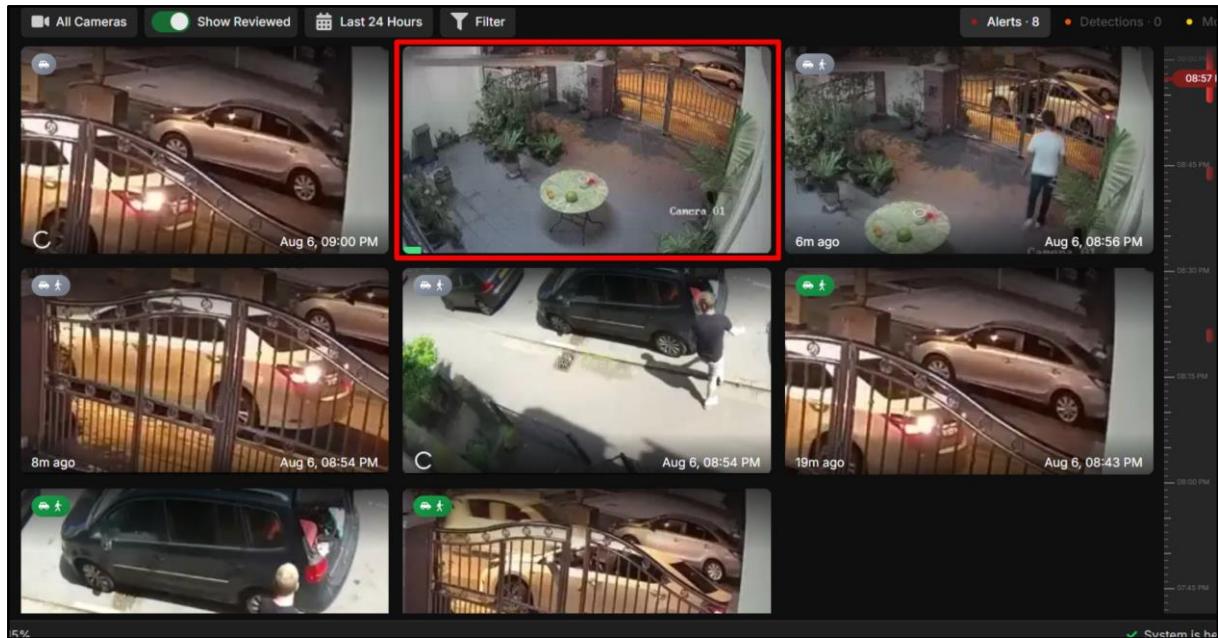


Figure 522. View event recording

Event recording is running:

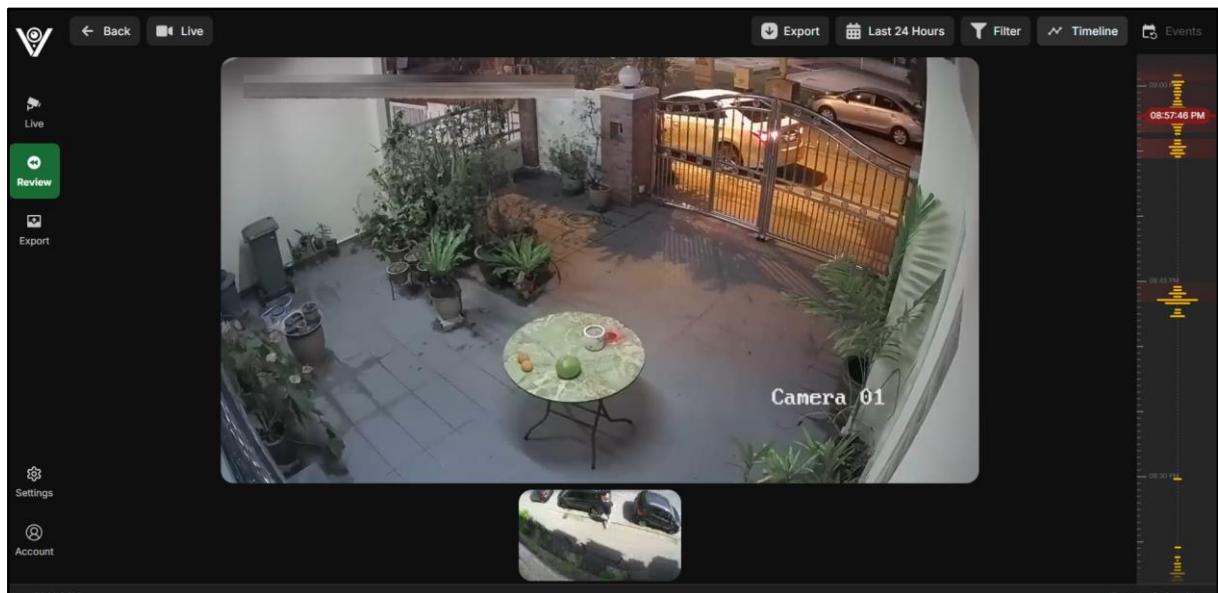


Figure 523. View event recording

Way 2: On the “Live Dashboard” page, you will see a small thumbnail above the camera live screen (as highlighted in below image). Click on that thumbnail to view event recording.

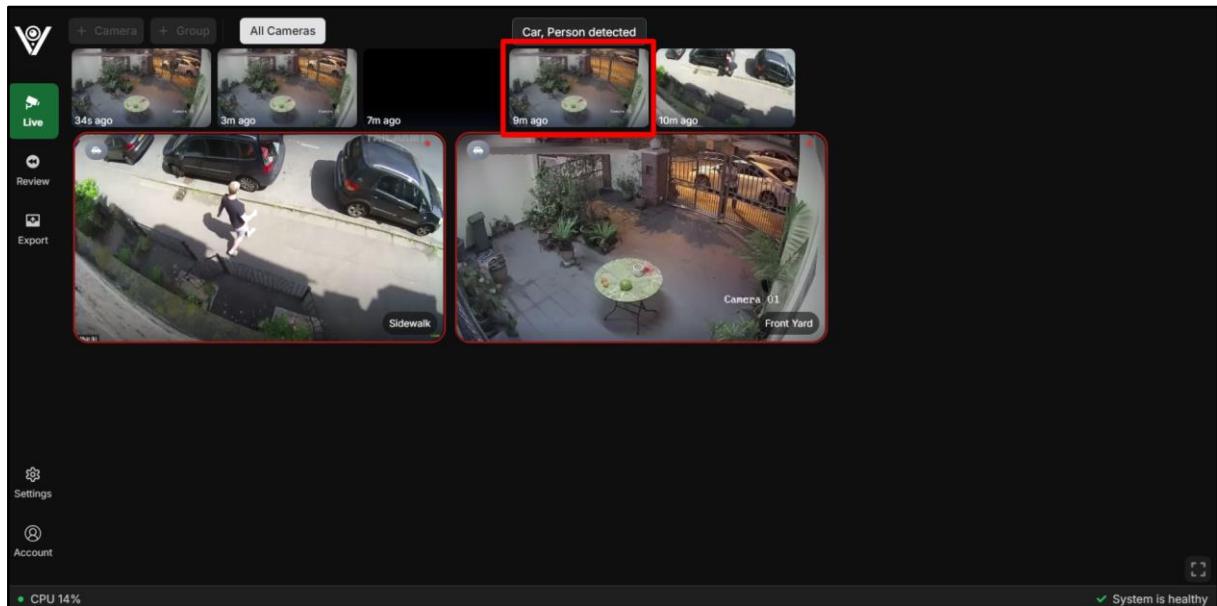


Figure 524. View event recording

Event recording is running:

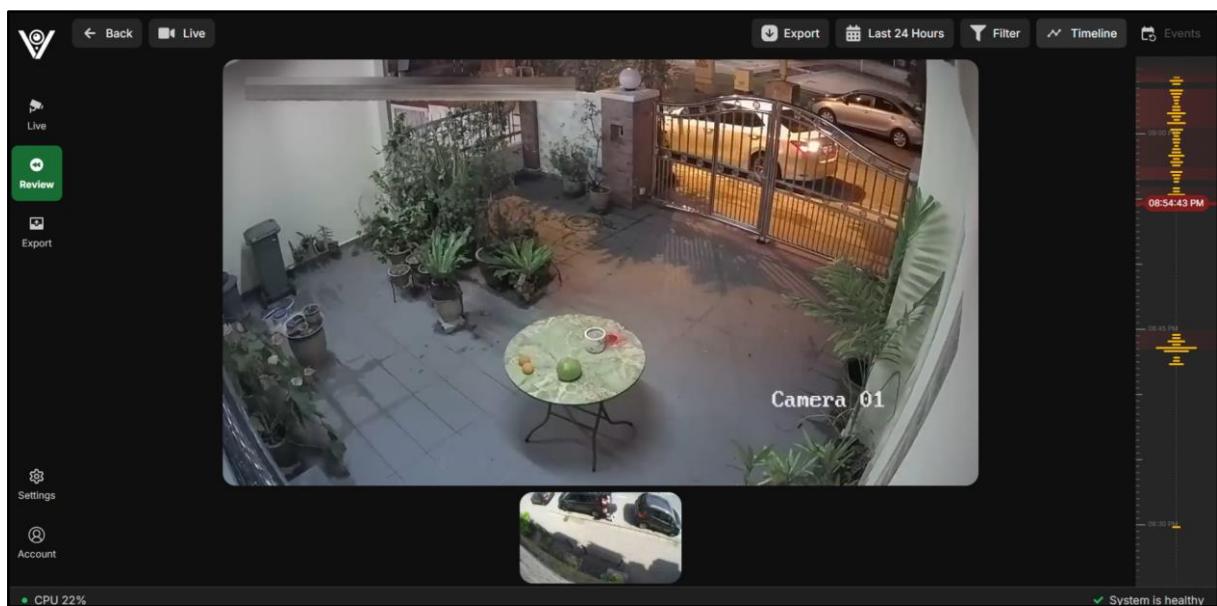


Figure 525. View event recording

Way 3: If you are viewing camera details, click on button “History” to view event recording of that camera.

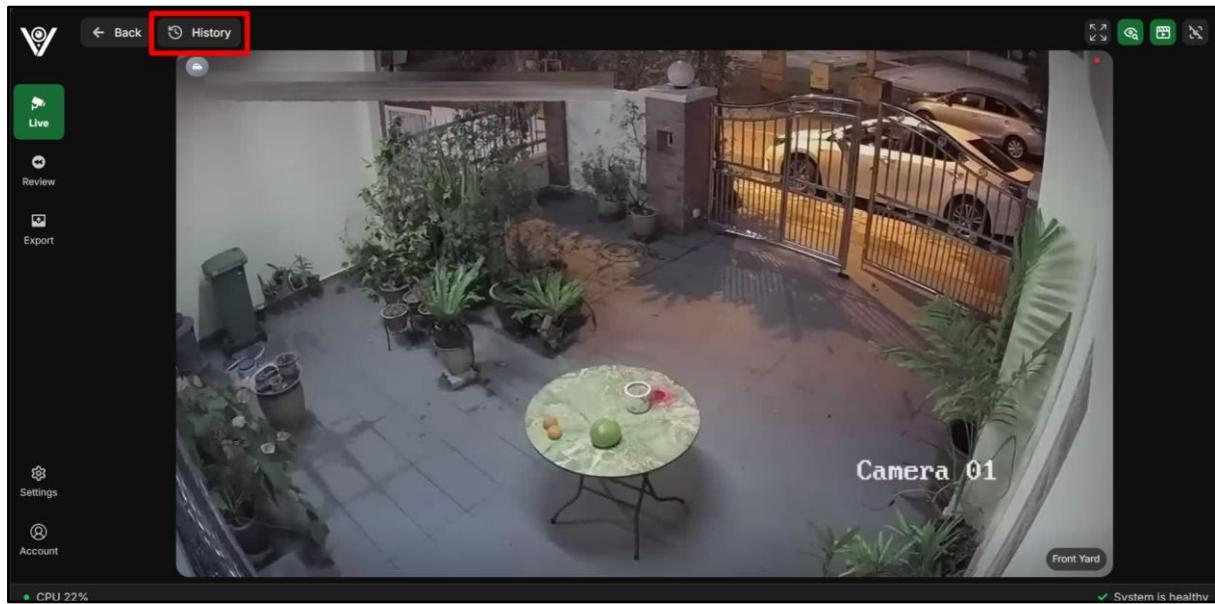


Figure 526. View event recording

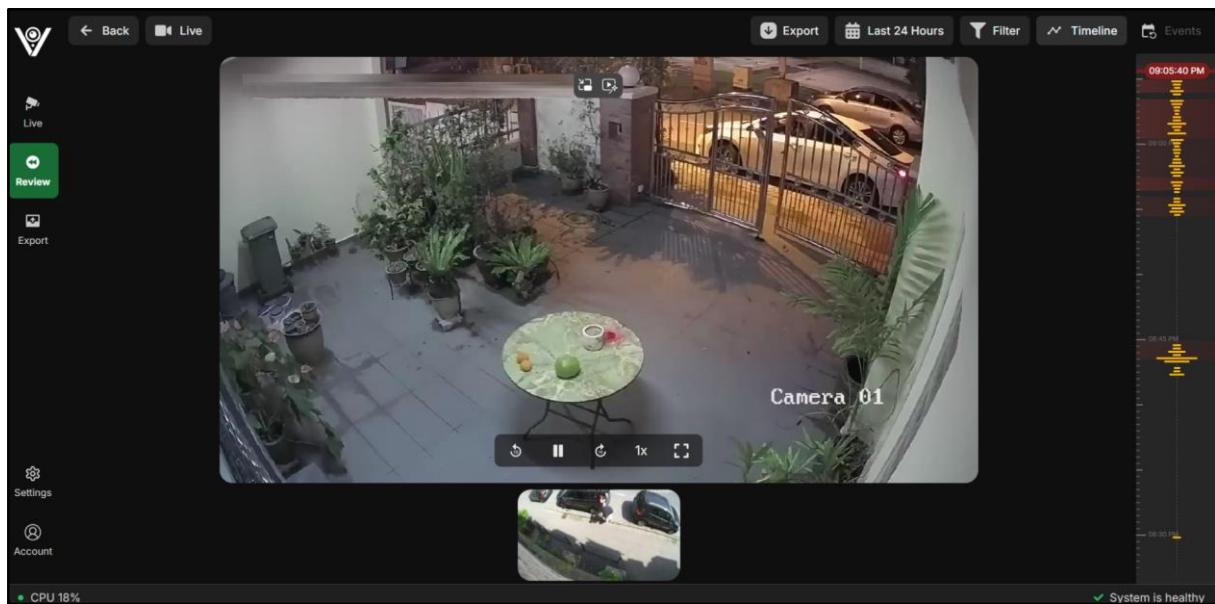


Figure 527. View event recording

If you want to view an event in a certain date, you can use a date filter by clicking on button “Last 24 Hours”.

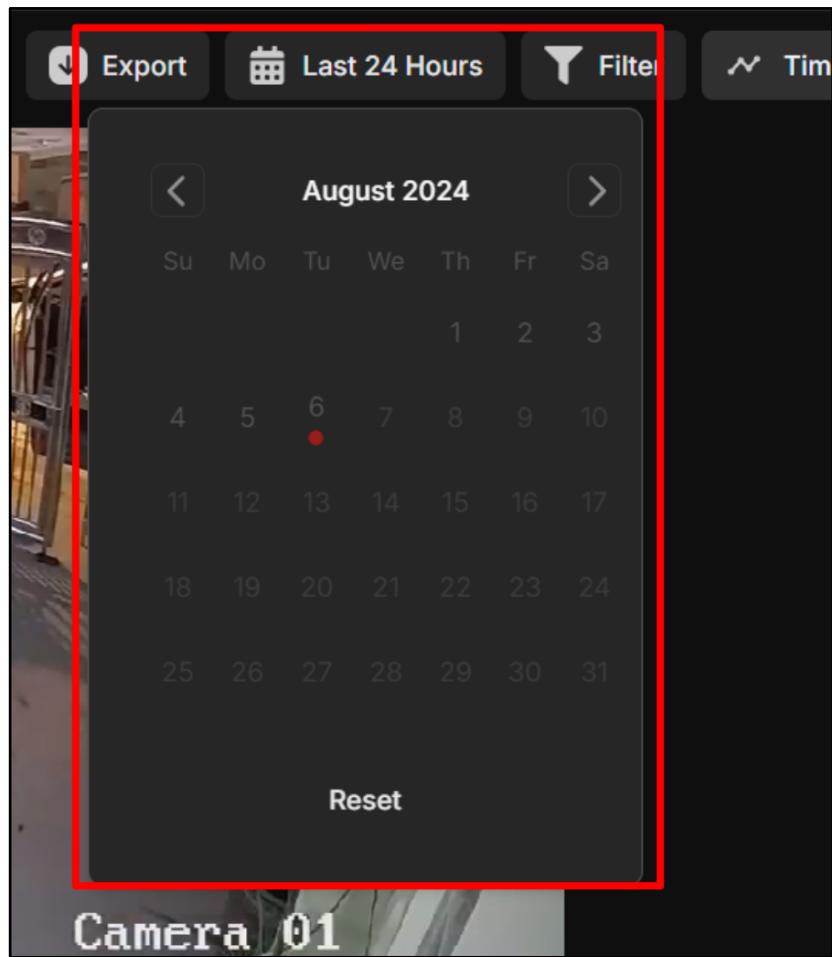


Figure 528. View event recording

You can scrub the timeline to view events by time (yellow bars stand for events)

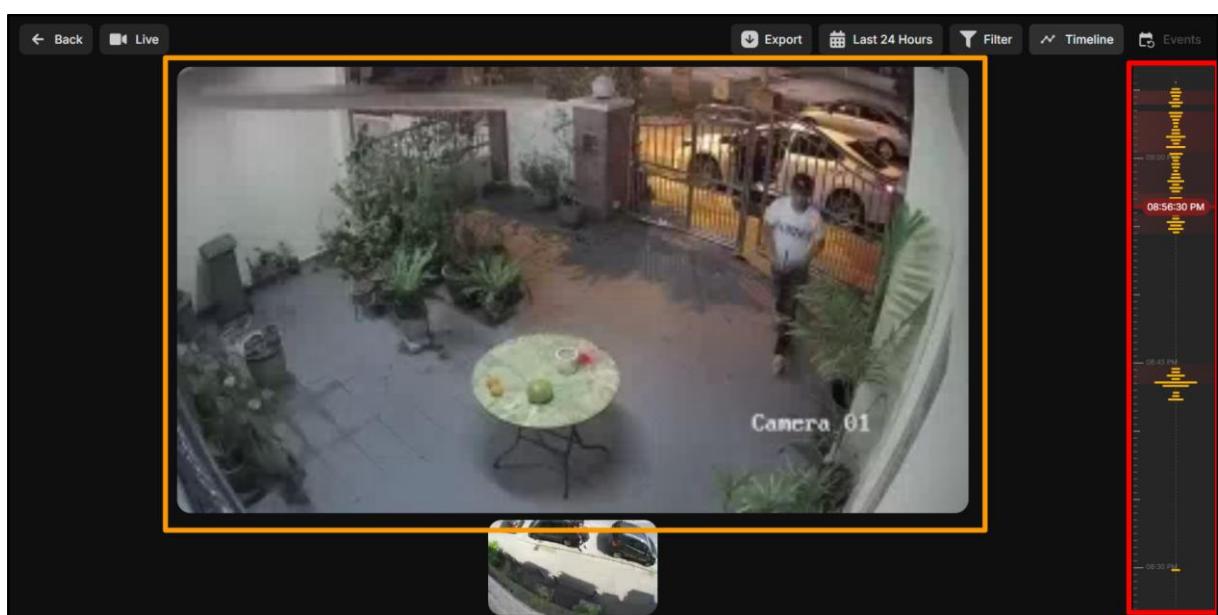


Figure 529. View event recording

You can view in different speed (4x, 8x, 12x, 16x)

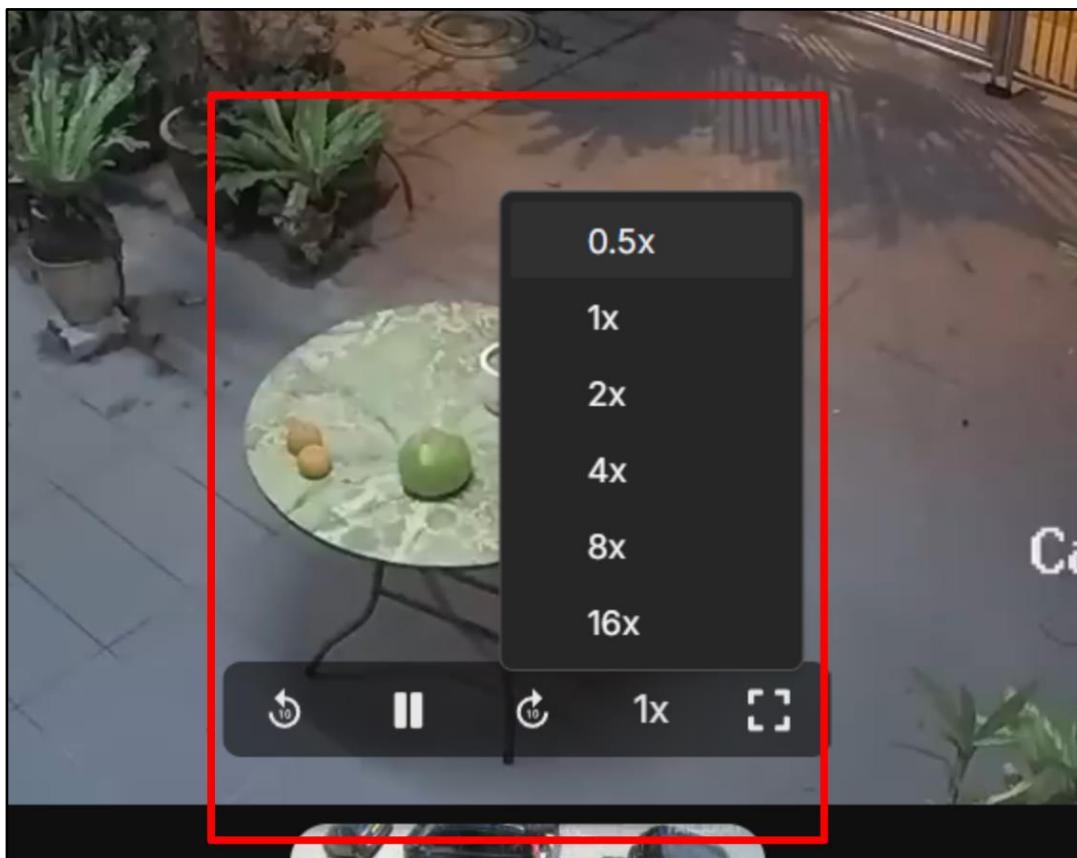


Figure 530. View event recording

You can click on the small thumbnail at the bottom of the screen (like highlighted in the image below) to switch between cameras.

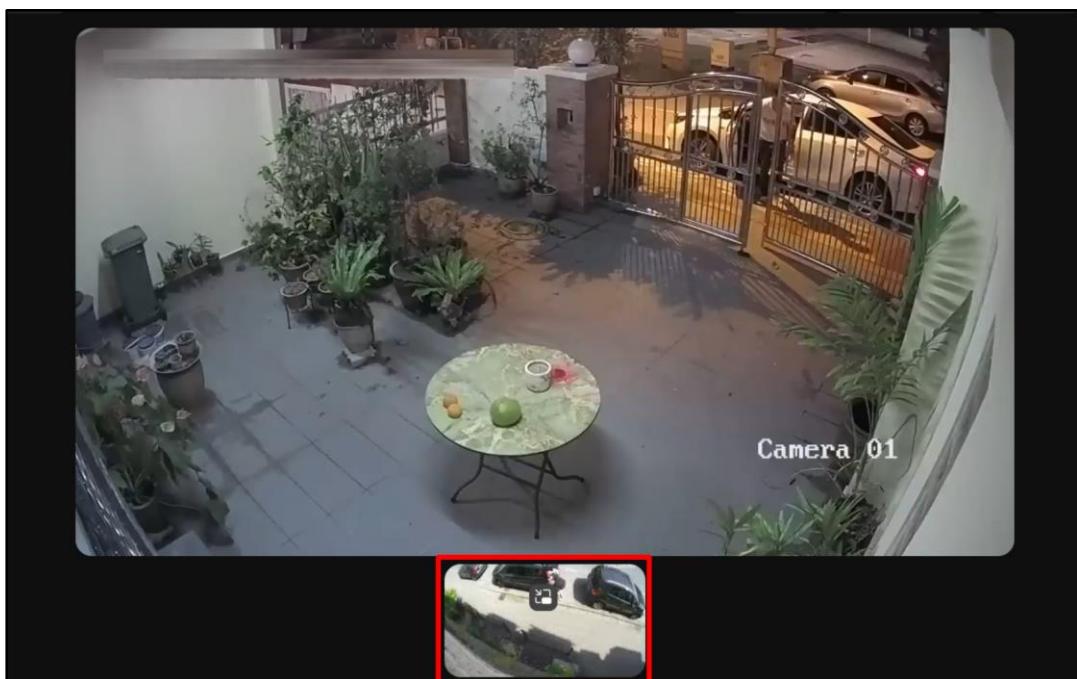


Figure 531. View event recording

Switched to another camera:

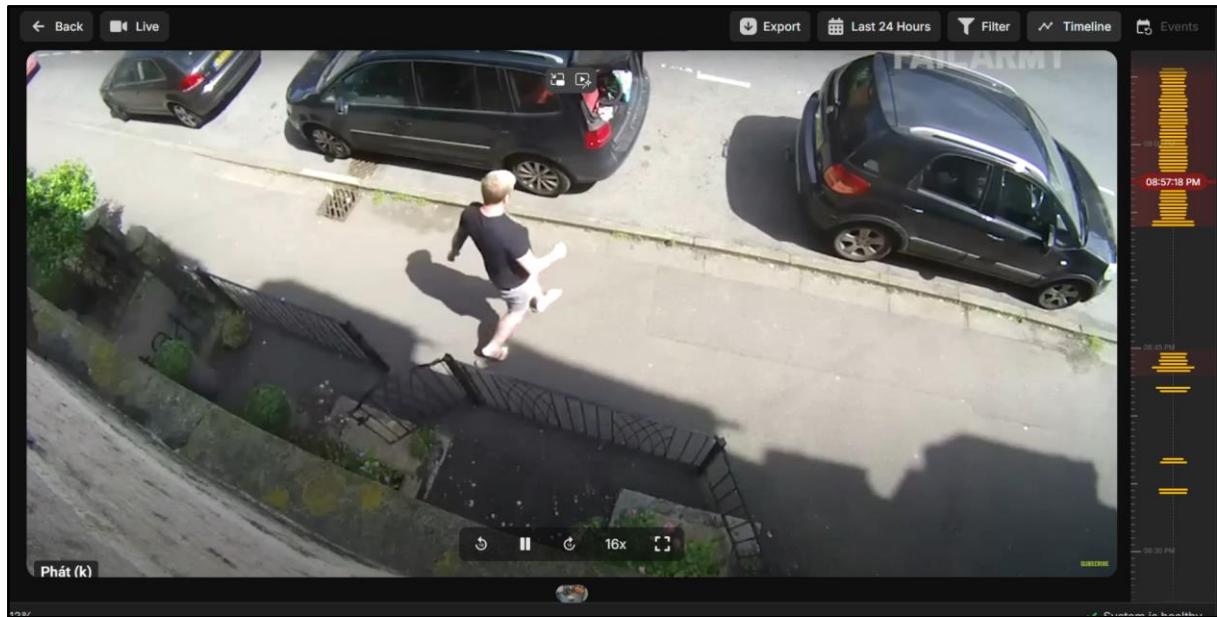


Figure 532. View event recording

Beside "Timeline", you can view event in list by clicking on button "Events", as shown below:

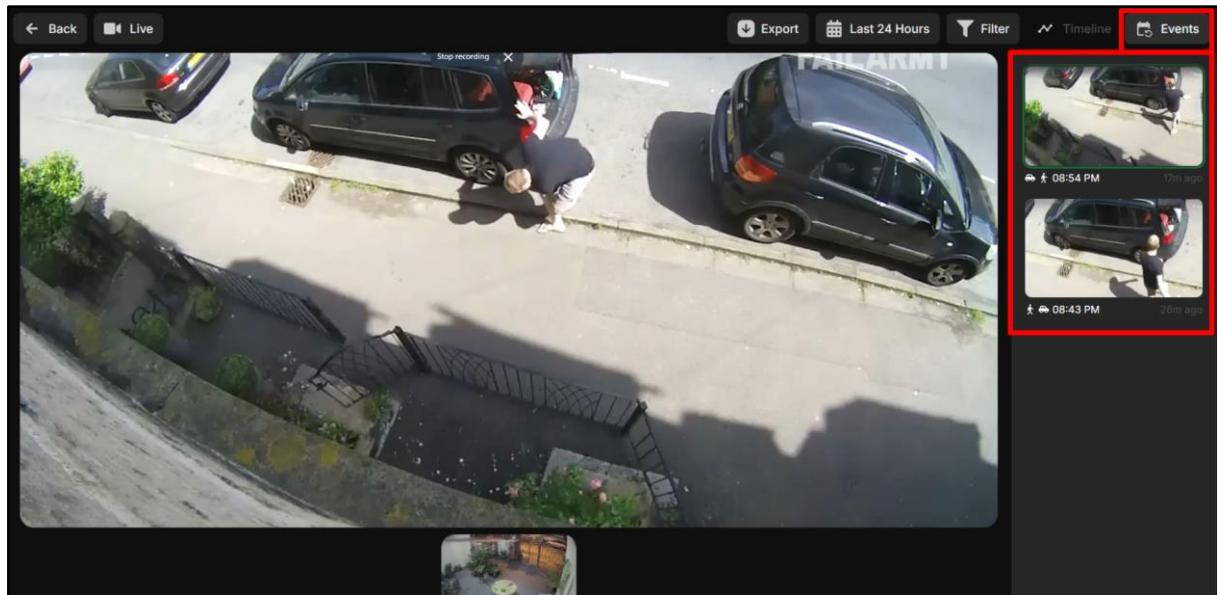


Figure 533. View event recording

Choose any event in the list to view event in detail:



Figure 534. View event recording

3.3.2.23 Mark events as reviewed

There are many ways to mark events as reviewed:

Ways 1:

On “Review” page, click on the event thumbnail you want to mark:

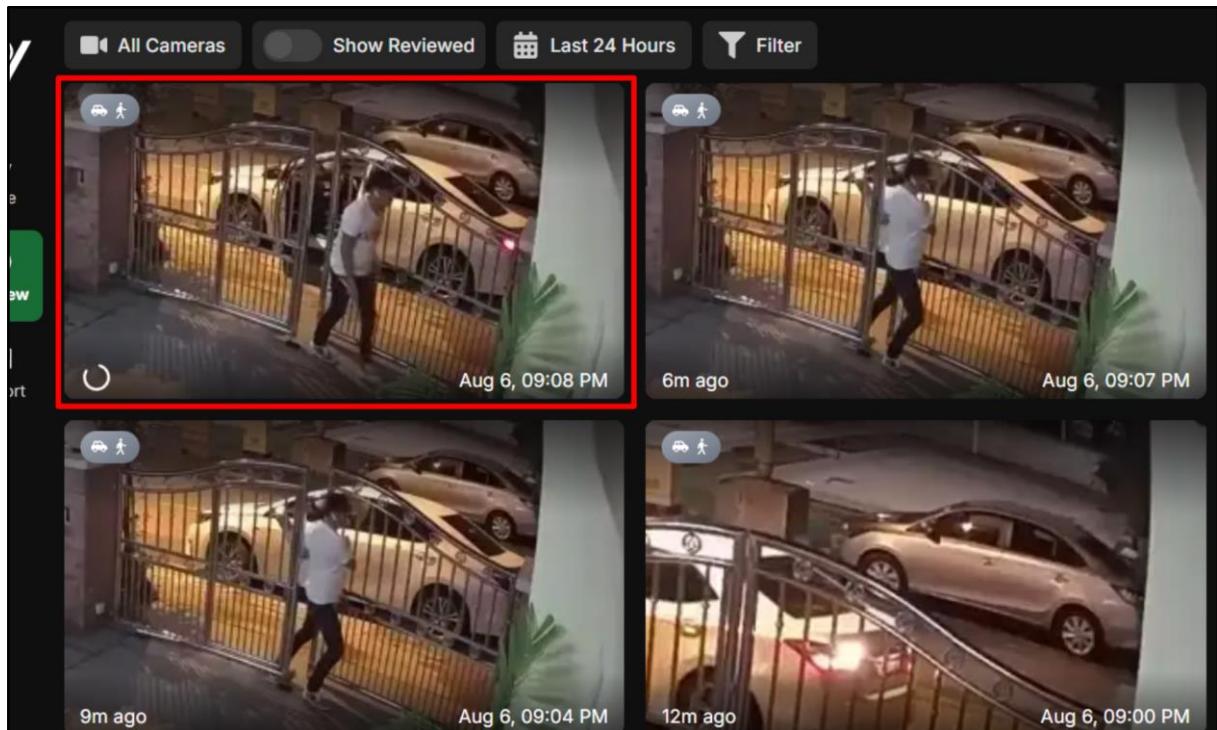


Figure 535. Mark events as reviewed

Event is show in detail:

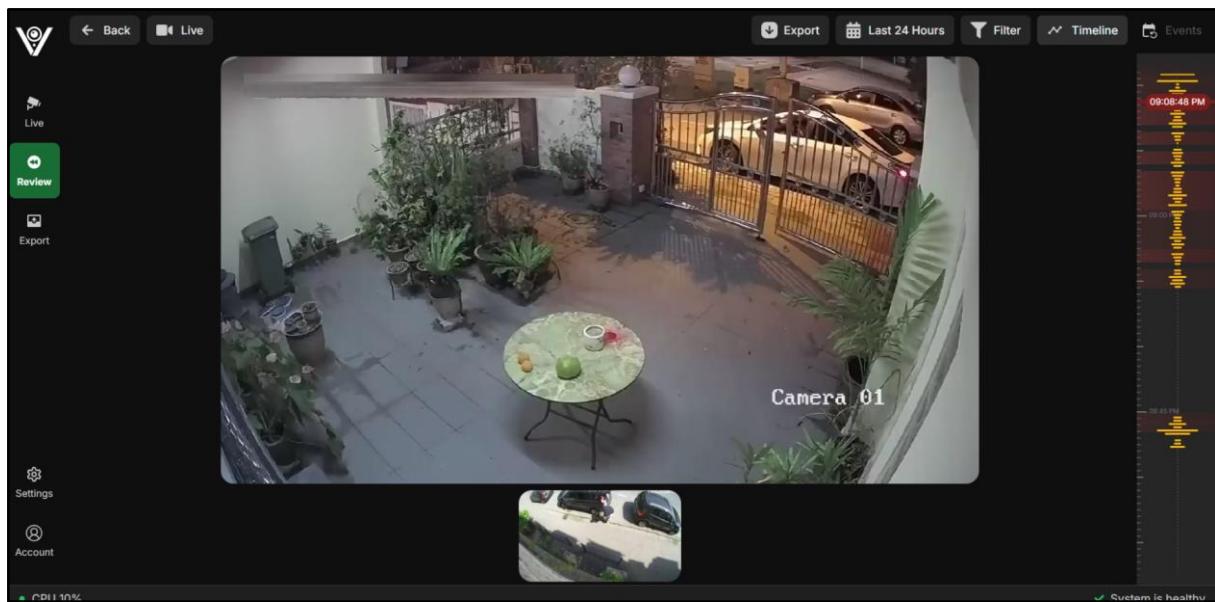


Figure 536. Mark events as reviewed

Go back to “Review” page to check, that event has been marked as reviewed:

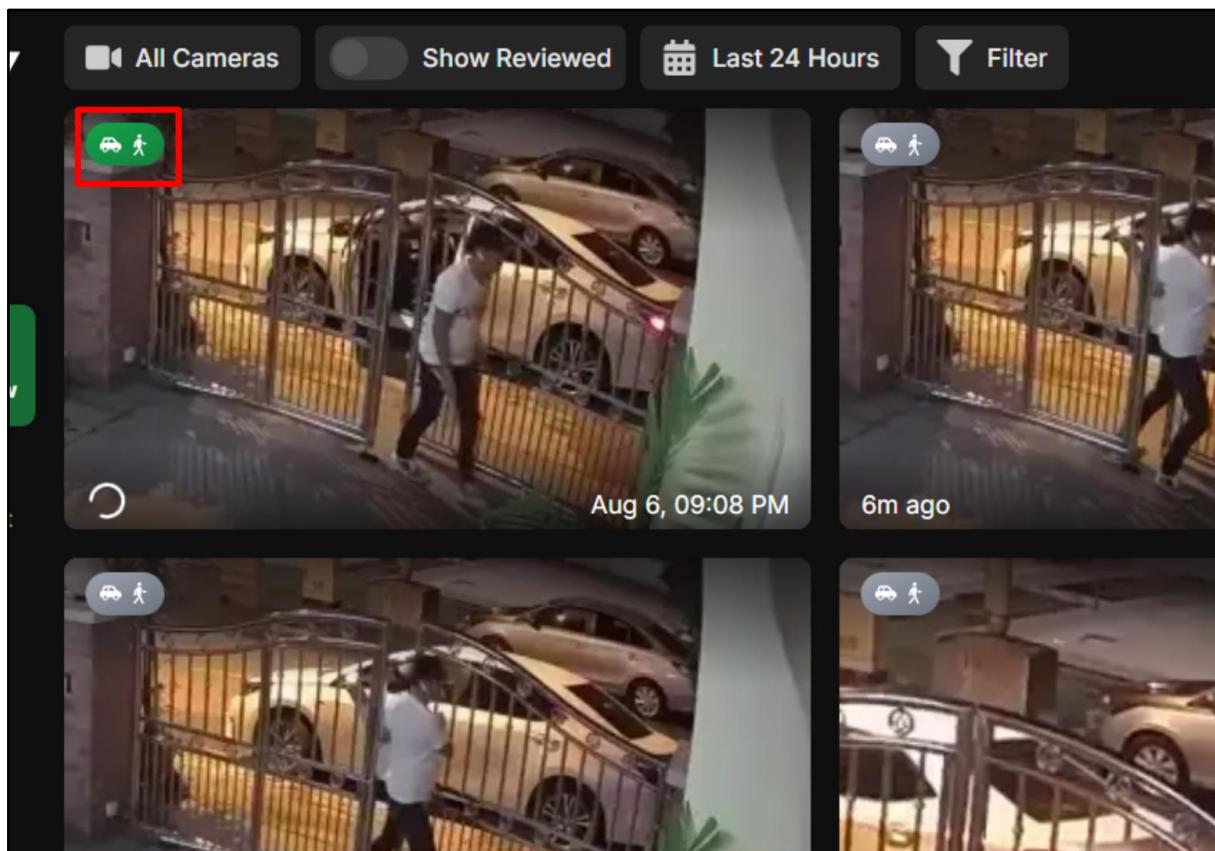


Figure 537. Mark events as reviewed

Way 2:

On “Review” page , right click to select many events at the same time:

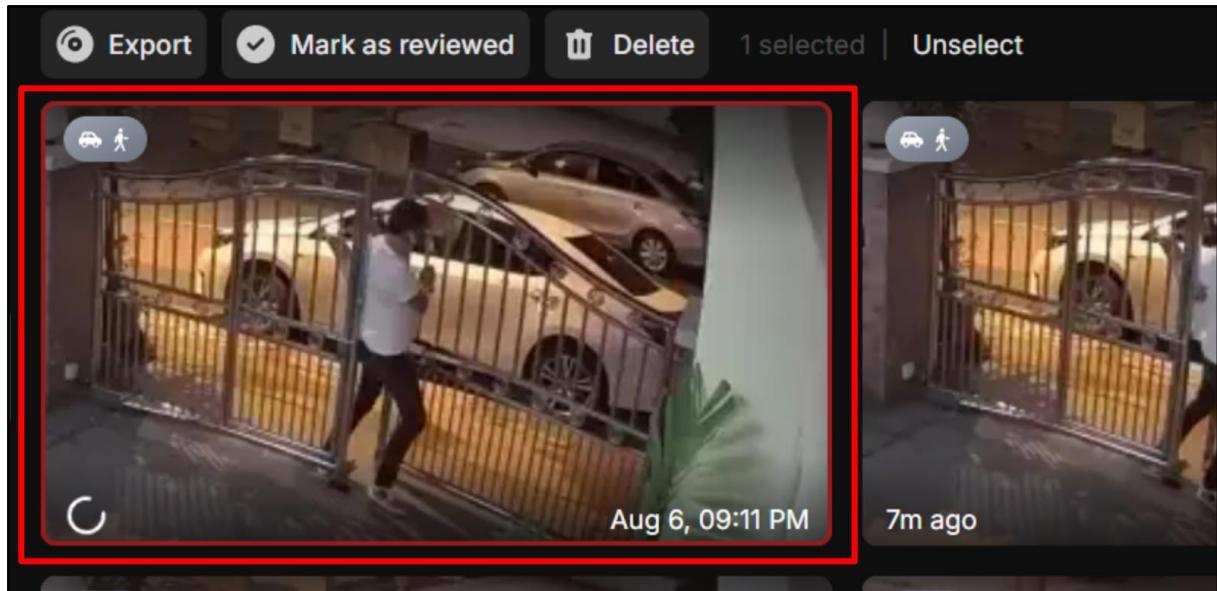


Figure 538. Mark events as reviewed

Click on “Mark as reviewed” on the top-left corner of the screen:

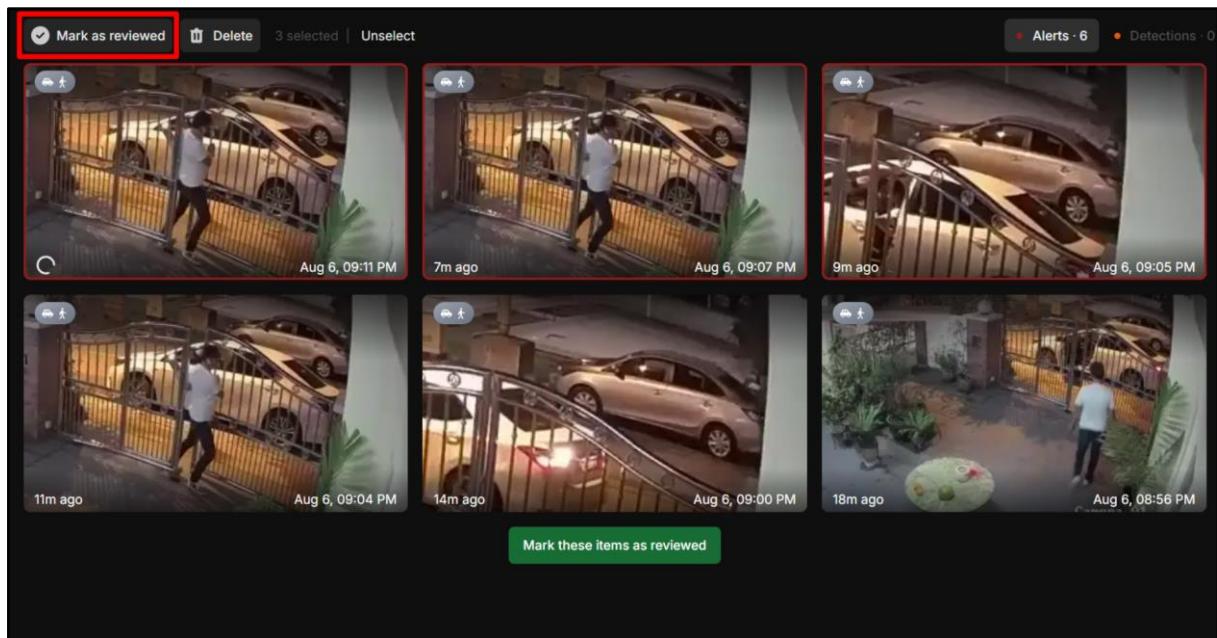


Figure 539. Mark events as reviewed

Selected events have been marked as reviewed, so that, they are hidden:

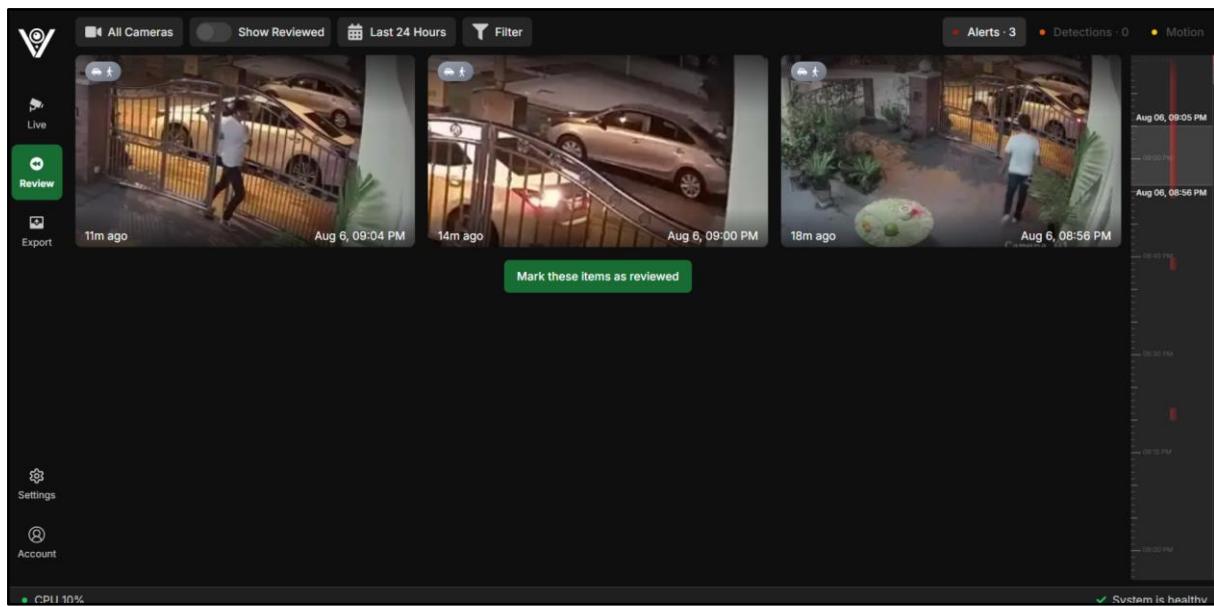


Figure 540. Mark events as reviewed

To check, you can click on button “Show Reviewed”. Then, those 3 events appears:

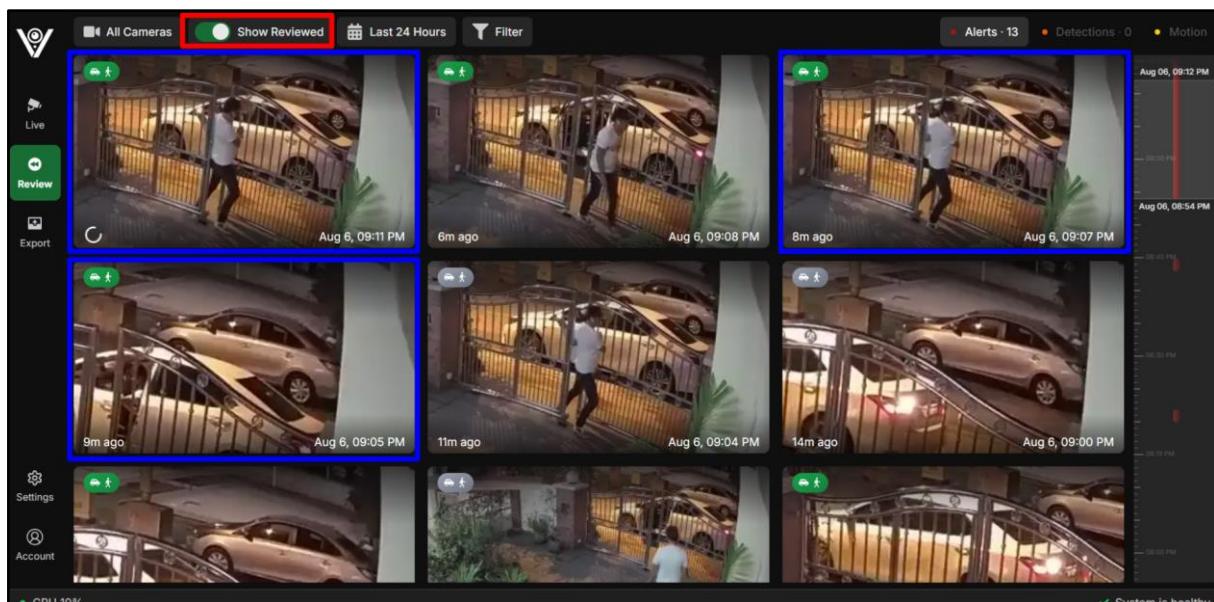


Figure 541. Mark events as reviewed

Way 3:

To mark all to-review events as reviewed, click the button “Mark these items as reviewed”, like in the image below:

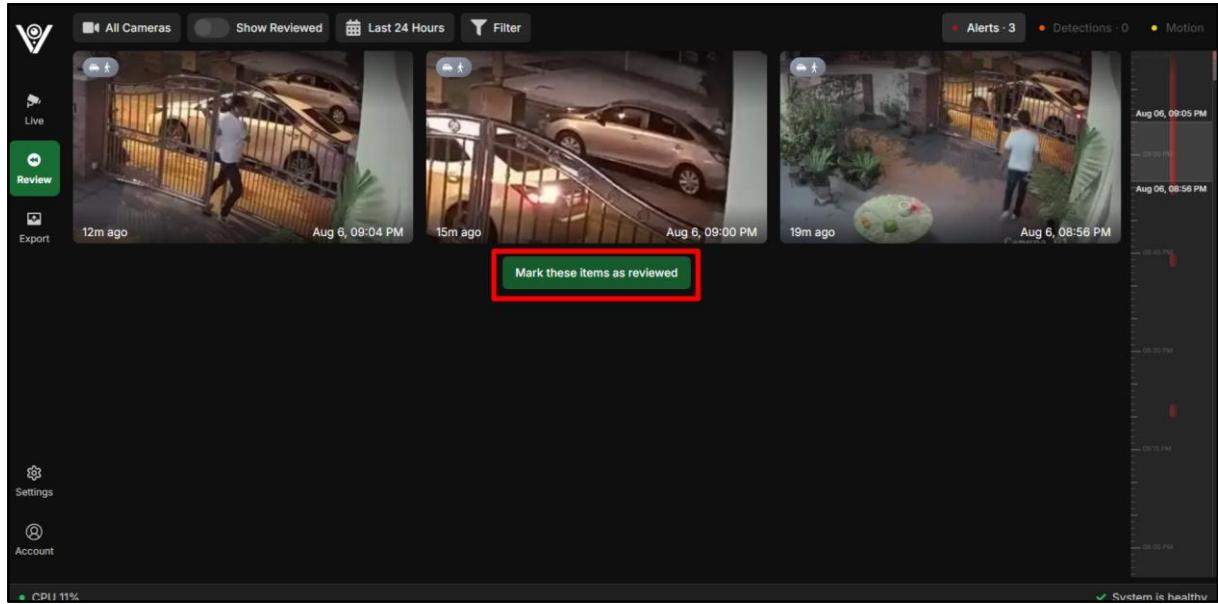


Figure 542. Mark events as reviewed

As the result, there are no alerts (events) to review:

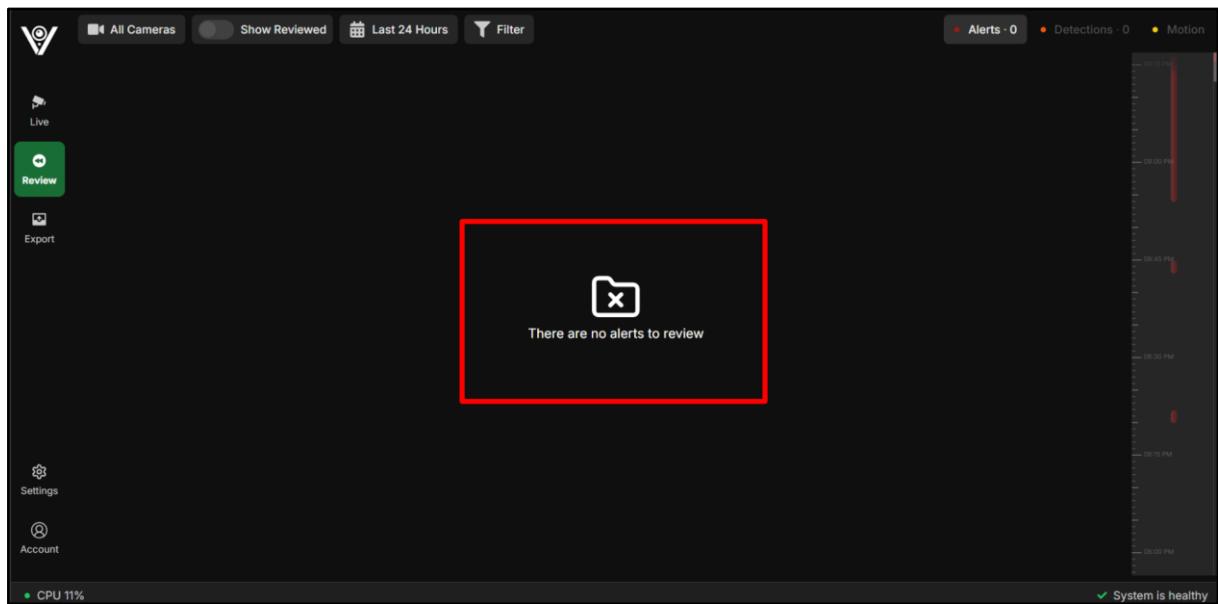


Figure 543. Mark events as reviewed

Way 4: If you are on the the Live dashboard you can click on the small thumbnail above the camera screen and view event, as shown below:

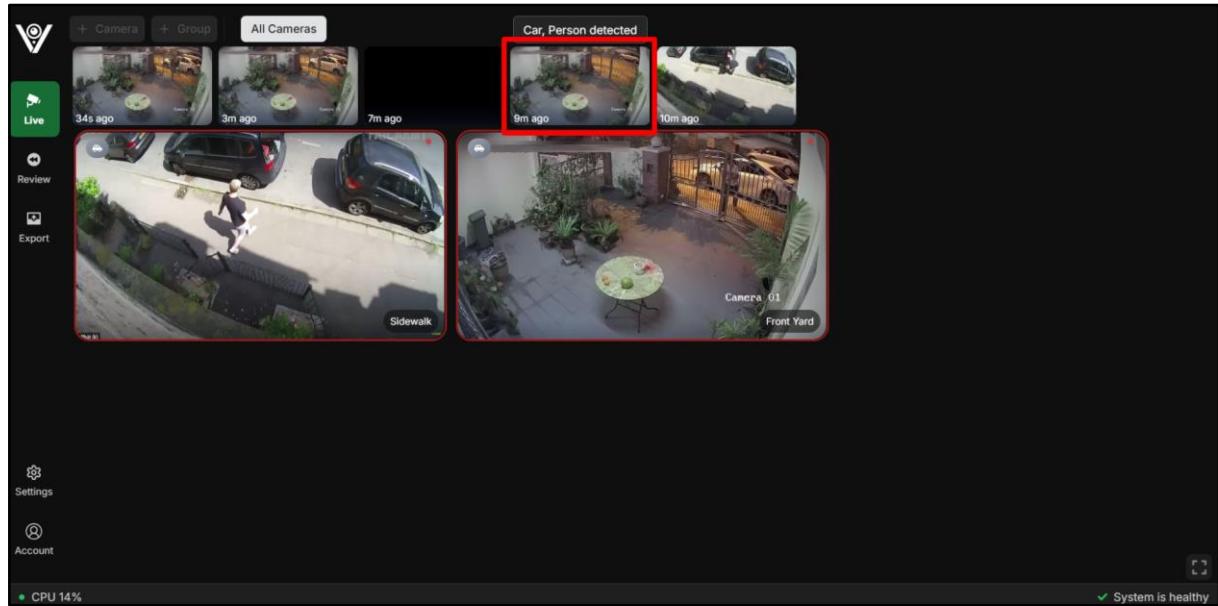


Figure 544. Mark events as reviewed

When you are able to view this screen, event is marked as reviewed:

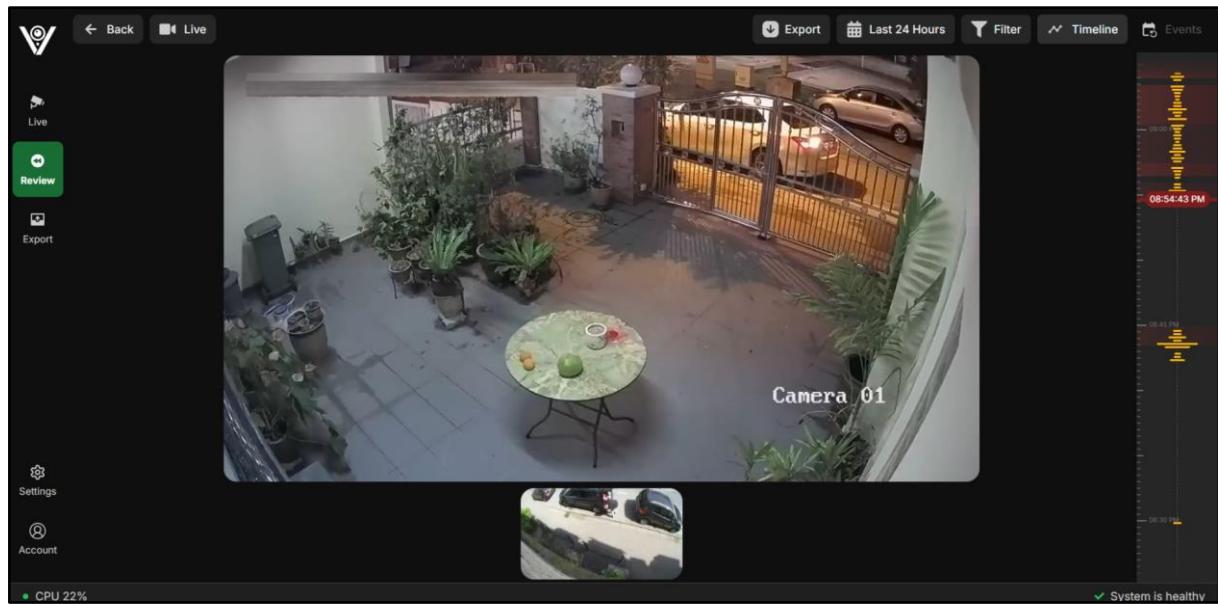


Figure 545. Mark events as reviewed

3.3.2.24 Delete events

To delete events, on the “Review” page, **right click** the event(s) you want to delete and click on the “Delete”. In the below example, 6 events have been selected, the number of alerts is now 13.

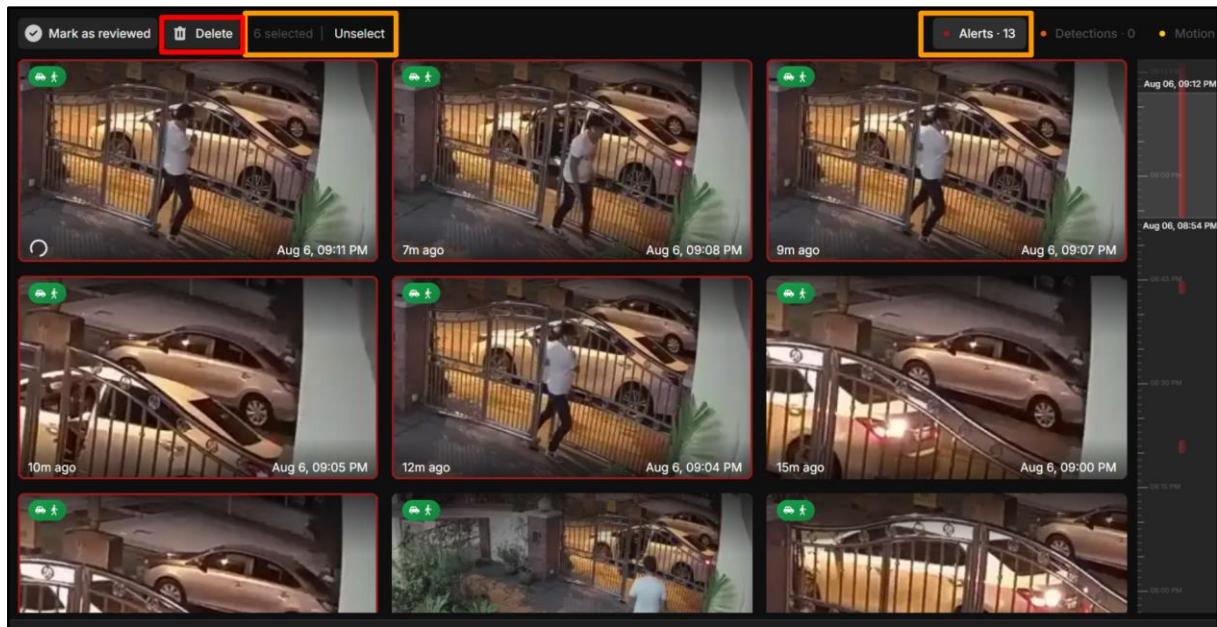


Figure 546. Delete events

After clicking on button “Delete” there are 7 alerts left.

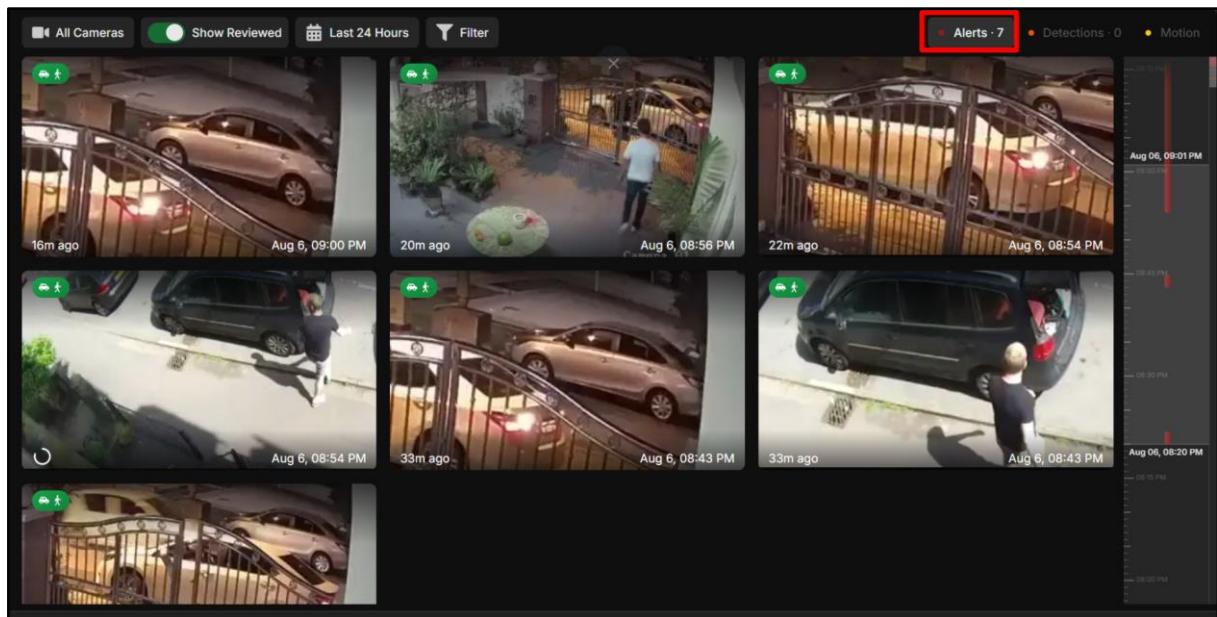


Figure 547. Delete events

3.3.2.25 View exports

To view export, click on button “Export” to go to “Export” page:

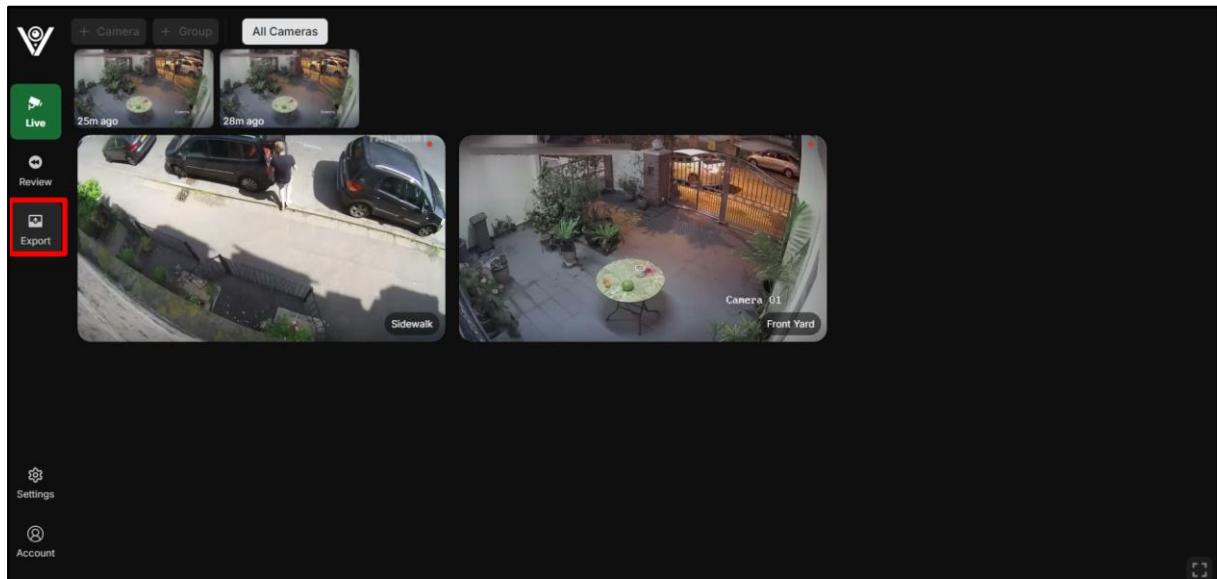


Figure 548. View exports

Case 1 - There exists no export: Nothing to export

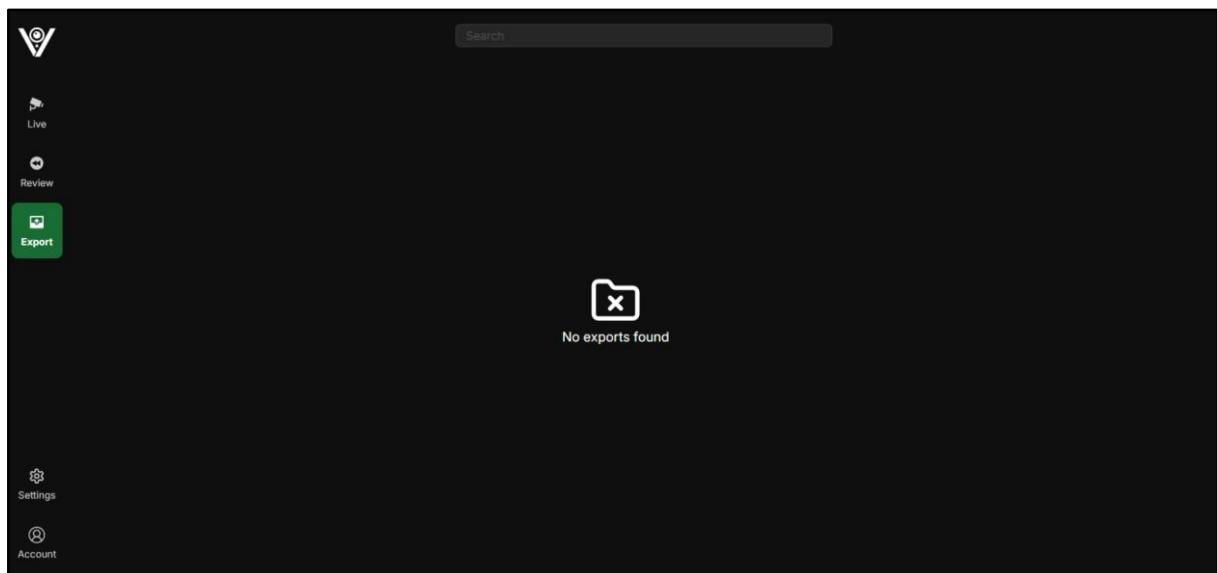


Figure 549. View exports

Case 2 - These exists exports

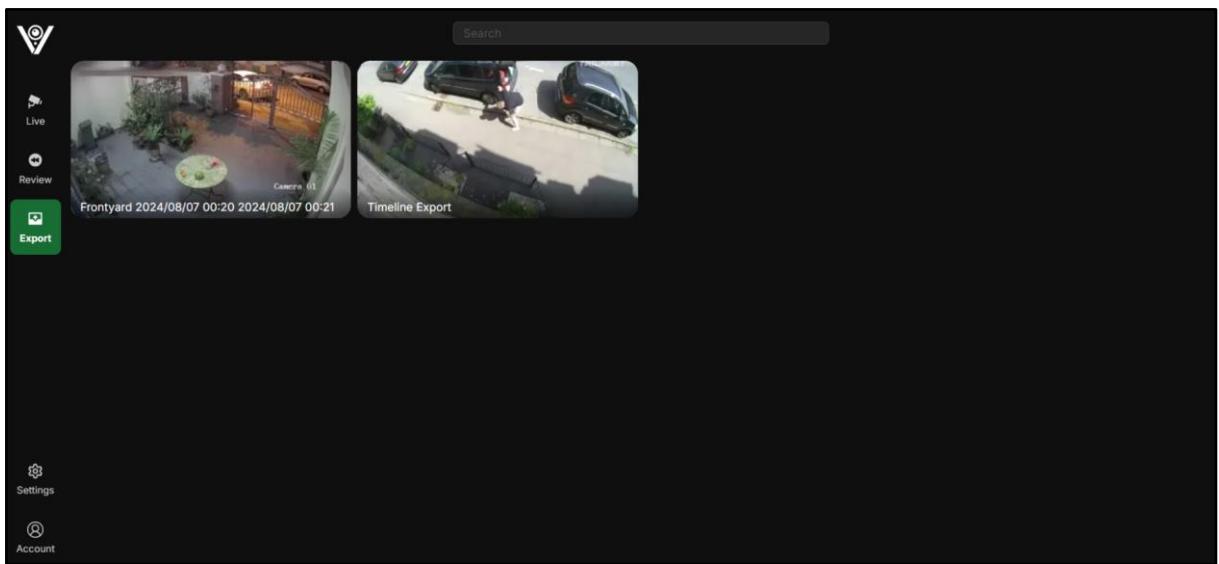


Figure 550. View exports

You can search export by name:

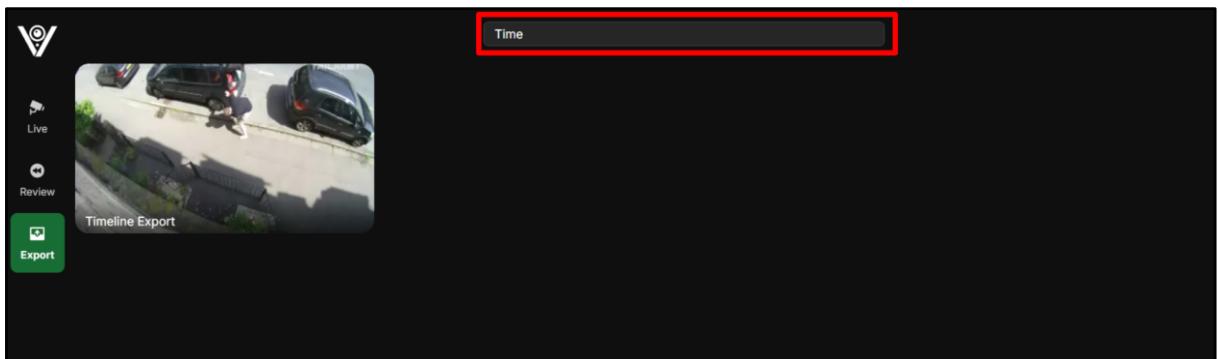


Figure 551. View exports

There exists no export name including search key "hello":

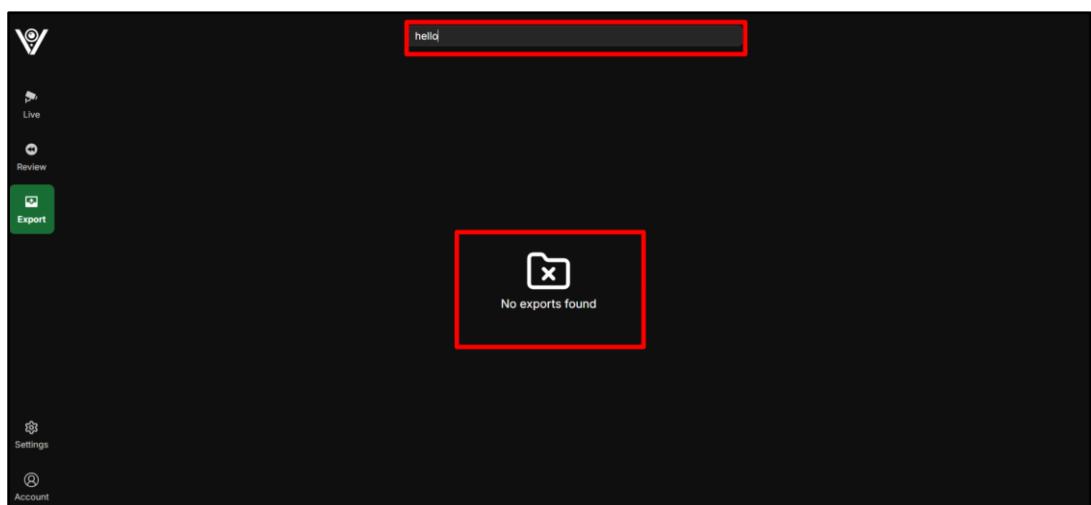


Figure 552. View exports

3.3.2.26 View export detail

To view an export in detail, click on thumbnail of that export:

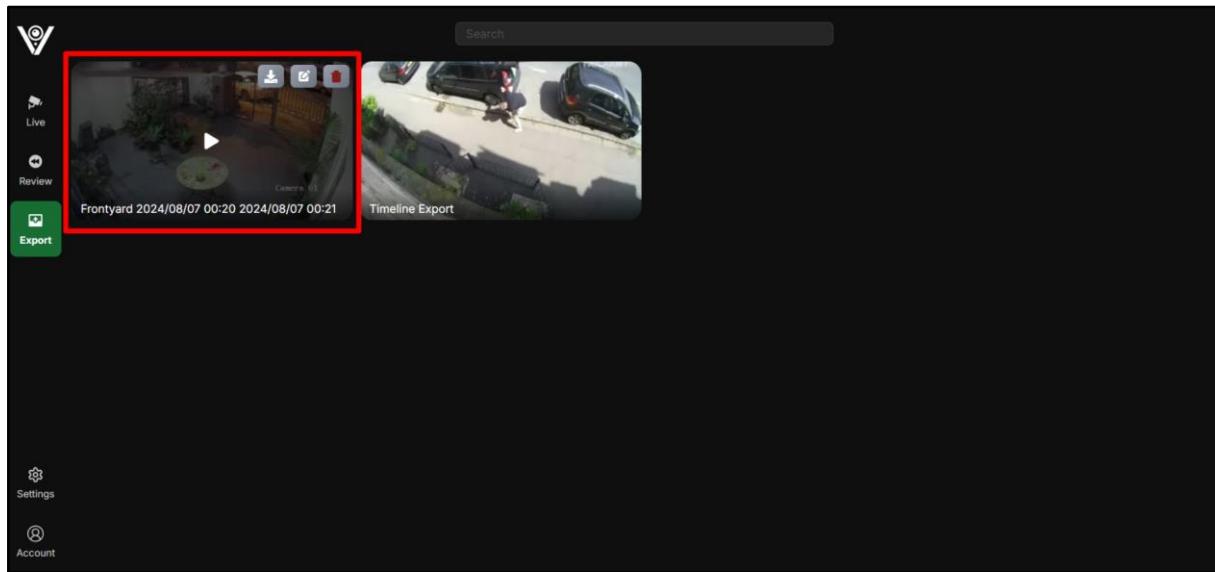


Figure 553. View export detail

Export is shown in detail (camera name, date, time, recording video, ...)

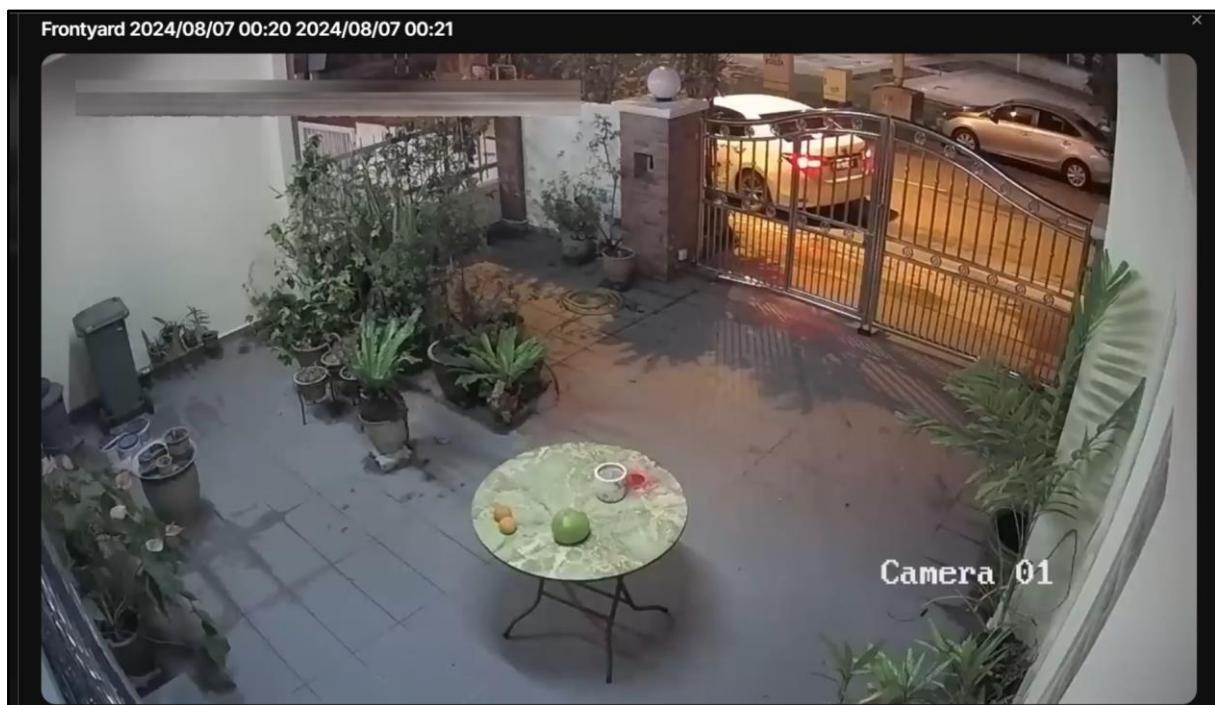


Figure 554. View export detail

3.3.2.27 Export event recording

To export event recording, go to “Review” page:

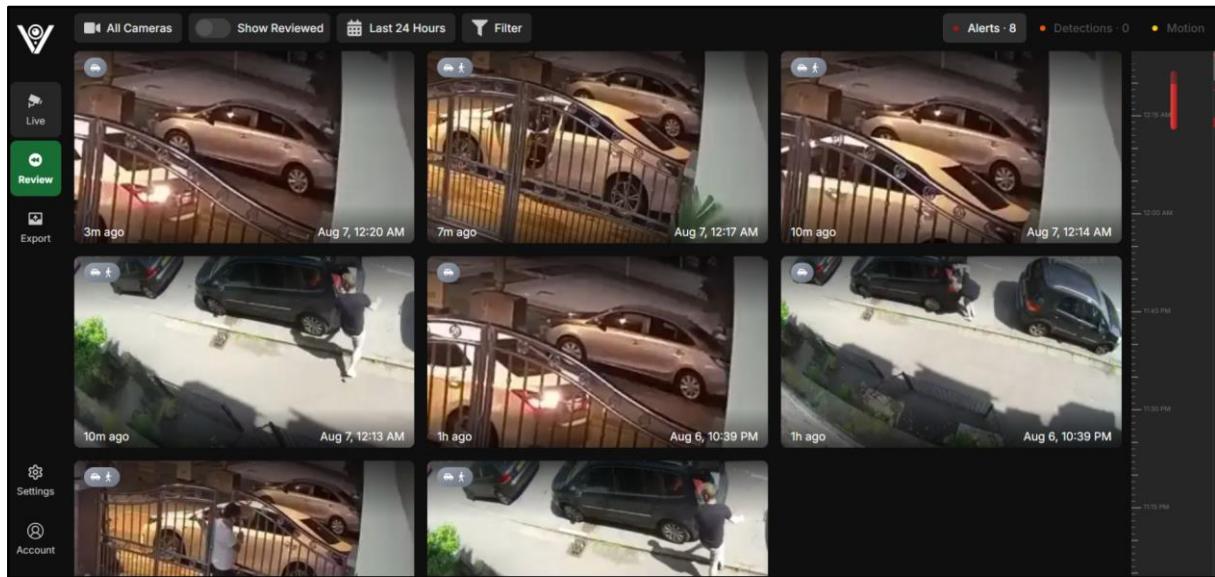


Figure 555. Export event recording

Right click on the event(s) you want to export. Then the “Export” button is shown. Please click on it to start exporting process

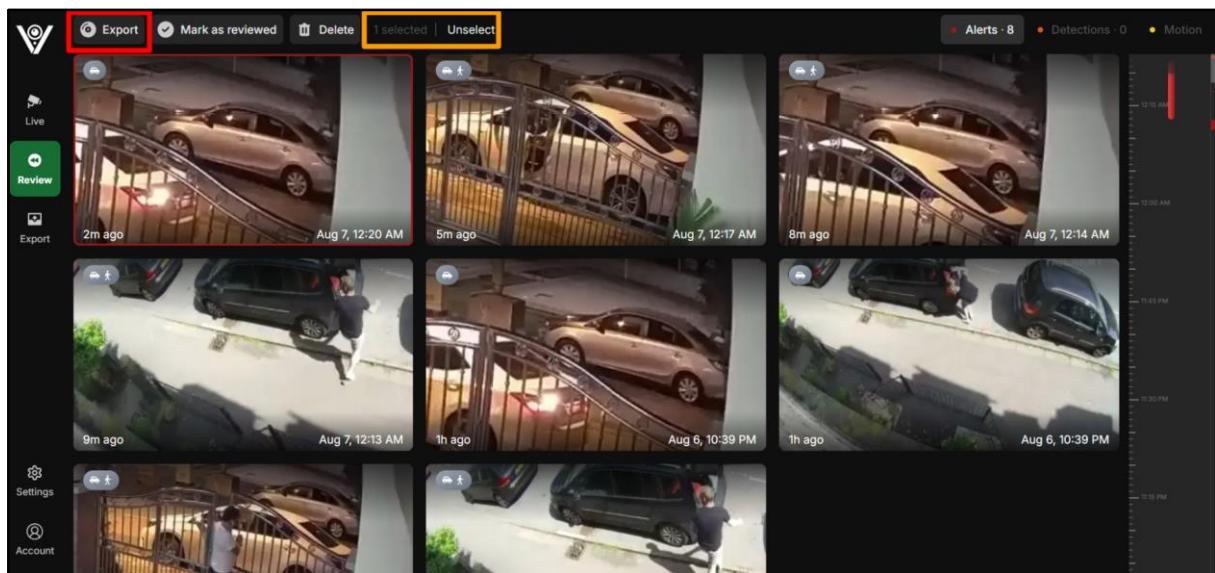


Figure 556. Export event recording

Then, a message from the system popped up “Successfully started export. View the file in the Export page”:

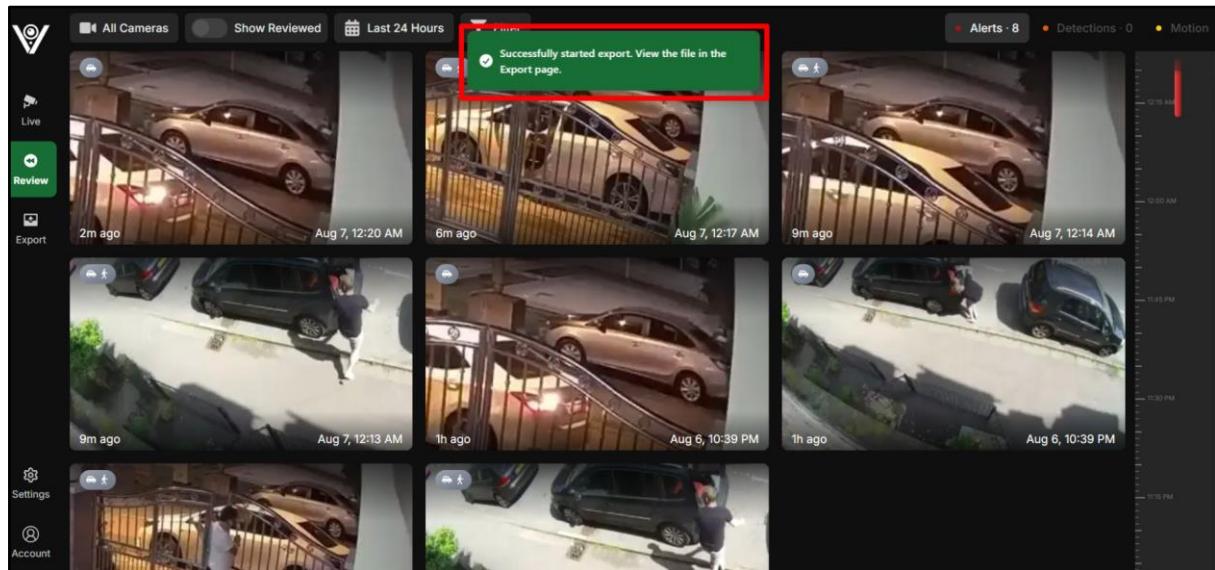


Figure 557. Export event recording

You can go to the “Export” page to check. A new export has been created:

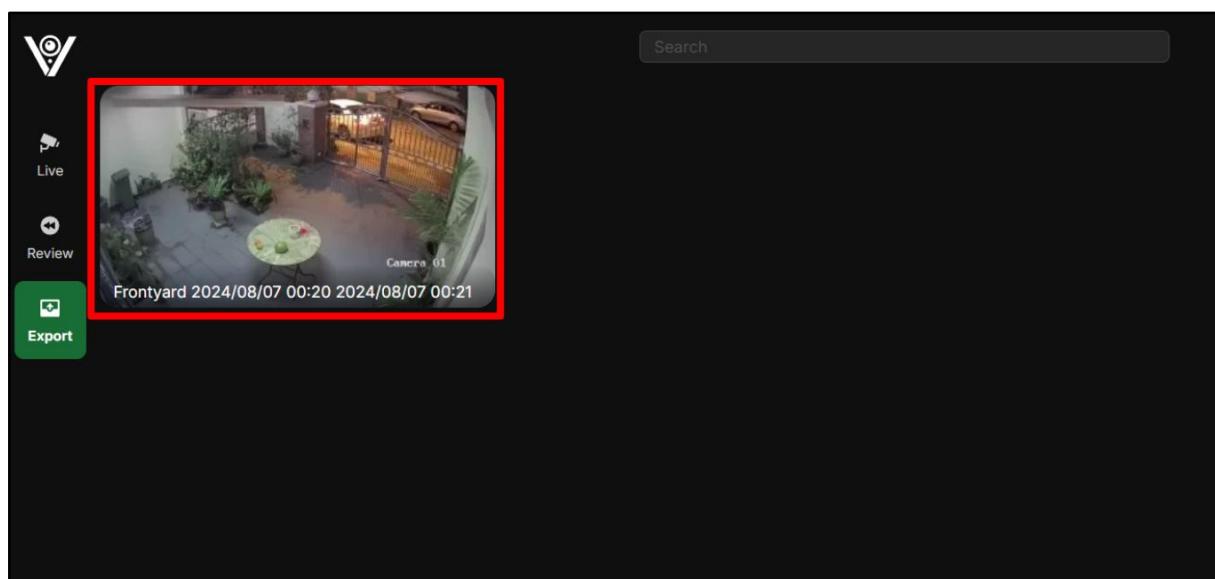


Figure 558. Export event recording

3.3.2.28 Export camera recording

Go to view camera in detail, click on button “Export”

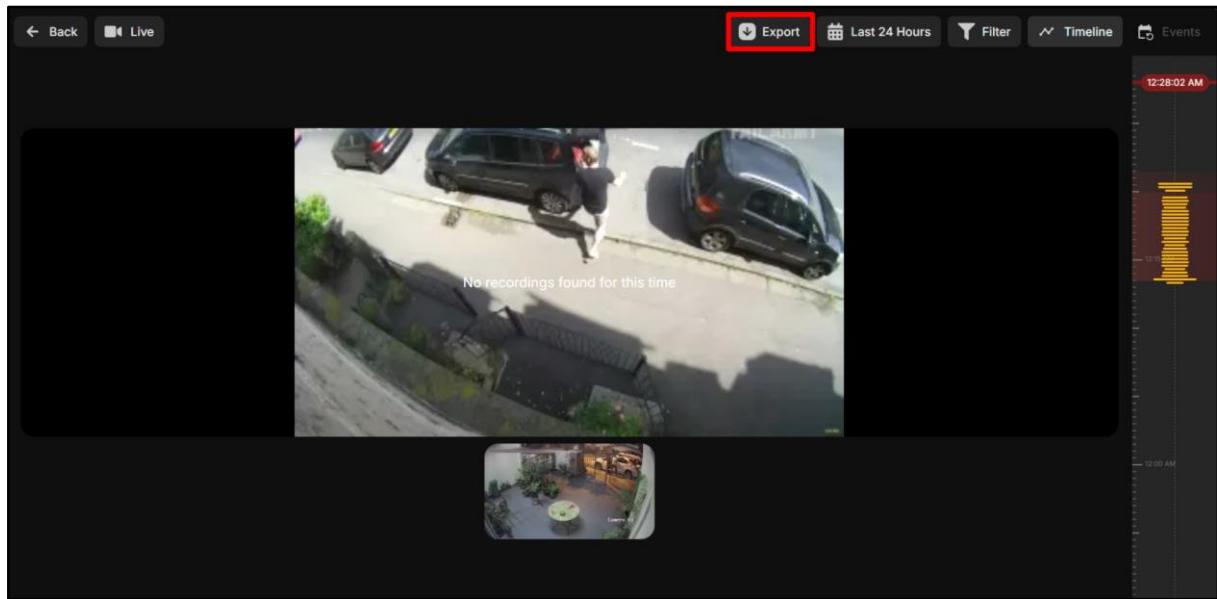


Figure 559. Export camera recording

Enter export name. If you did not enter any input name, the default name is “<camera id> - <date>”. After that click on the button “Export” to start exporting.

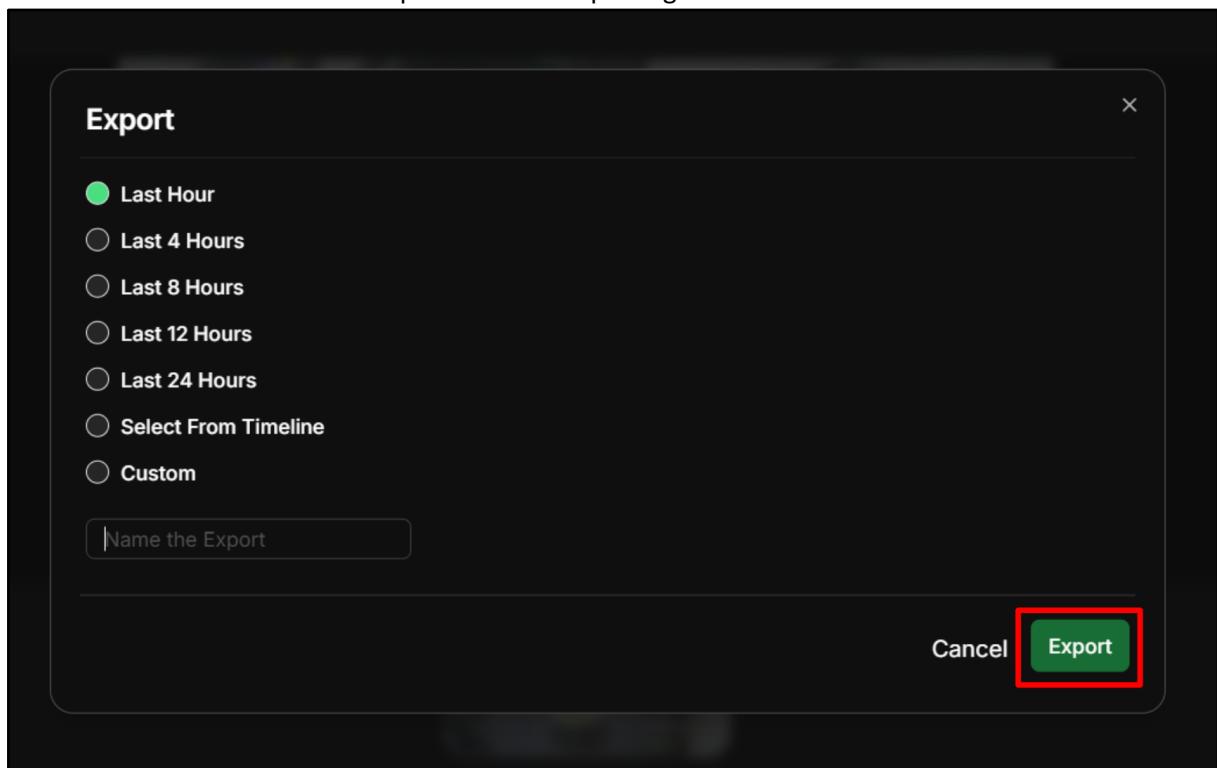


Figure 560. Export camera recording

Then, a message from system pop up “Successfully started export. View the file in the Export page”:

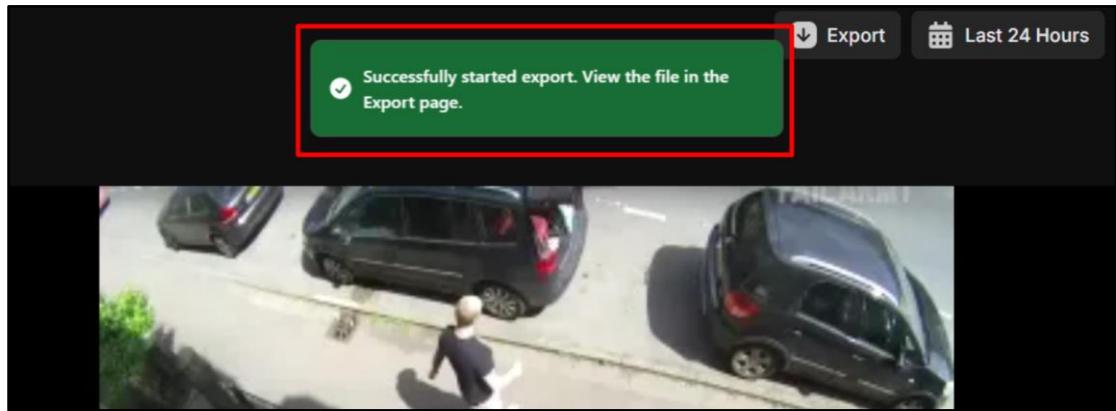


Figure 561. Export camera recording

go to “Export” paged, you can see a new export has been created with default name:

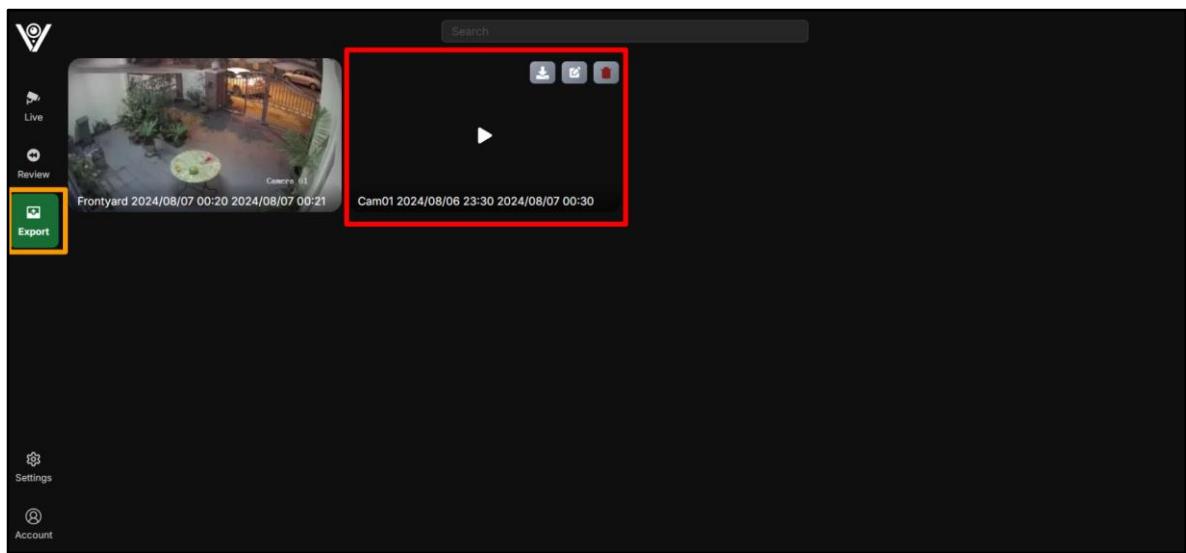


Figure 562. Export camera recording

Click on that export thumbnail to view in detail:

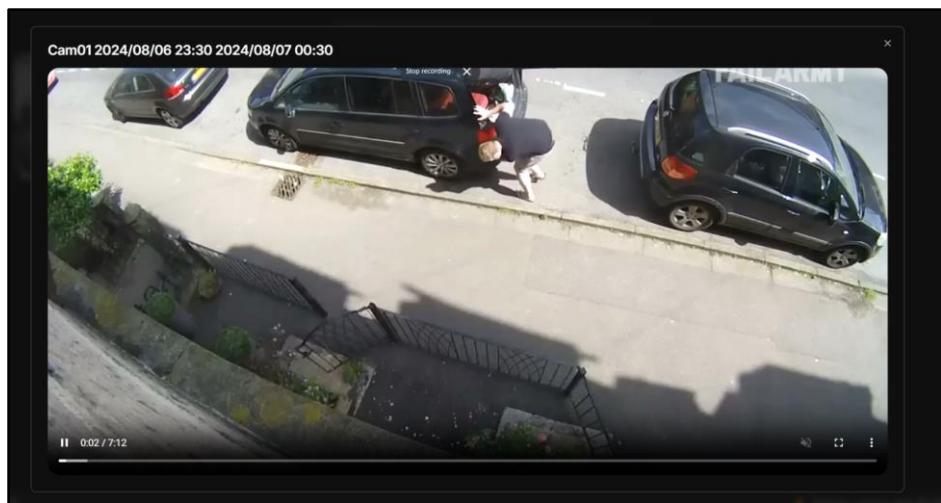


Figure 563. Export camera recording

Try another case: choose a time range for the export and click “Export”:

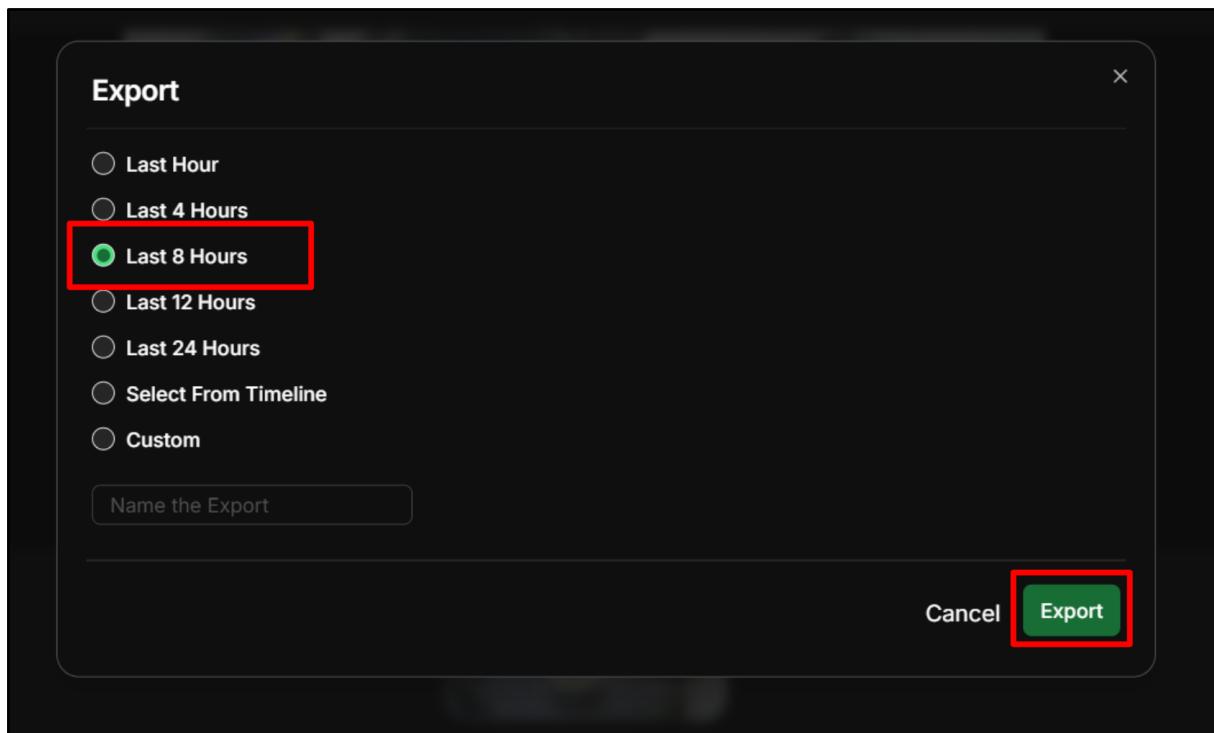


Figure 564. Export camera recording

Then, a message from the system popped up “Successfully started export. View the file in the Export page”:

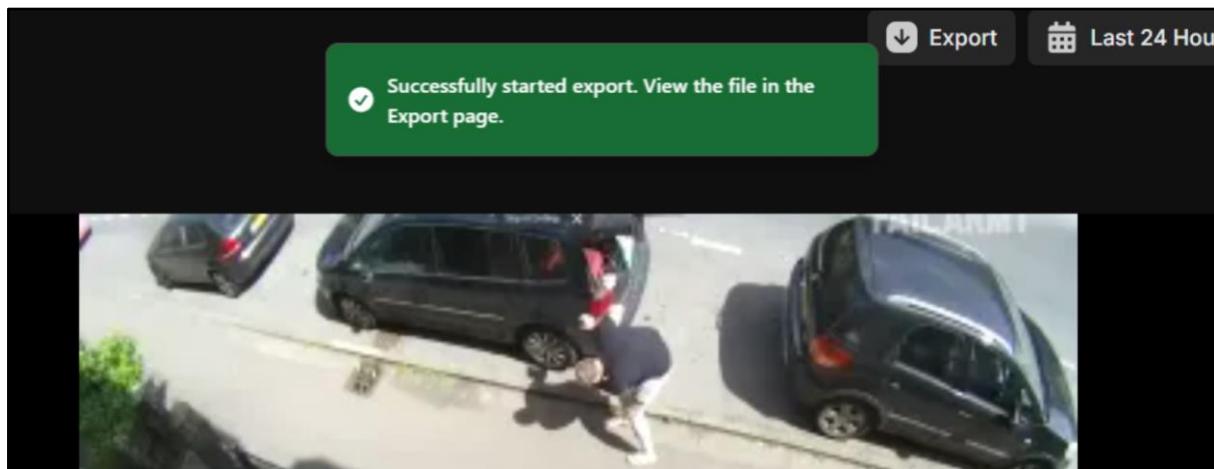


Figure 565. Export camera recording

Exporting process is ongoing:

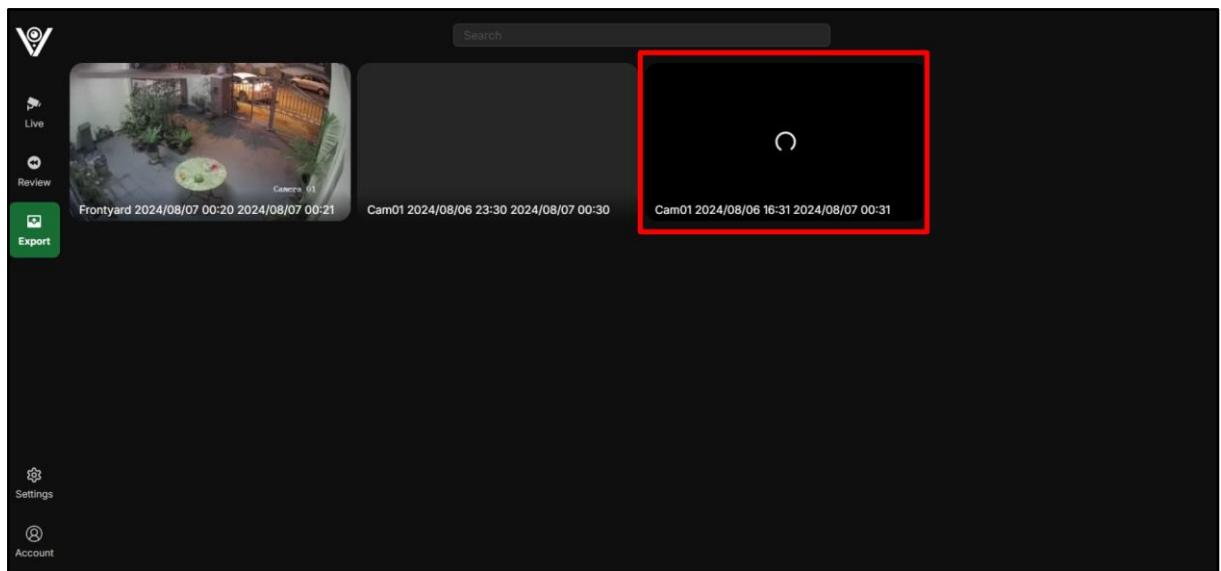


Figure 566. Export camera recording

A new export has been created:

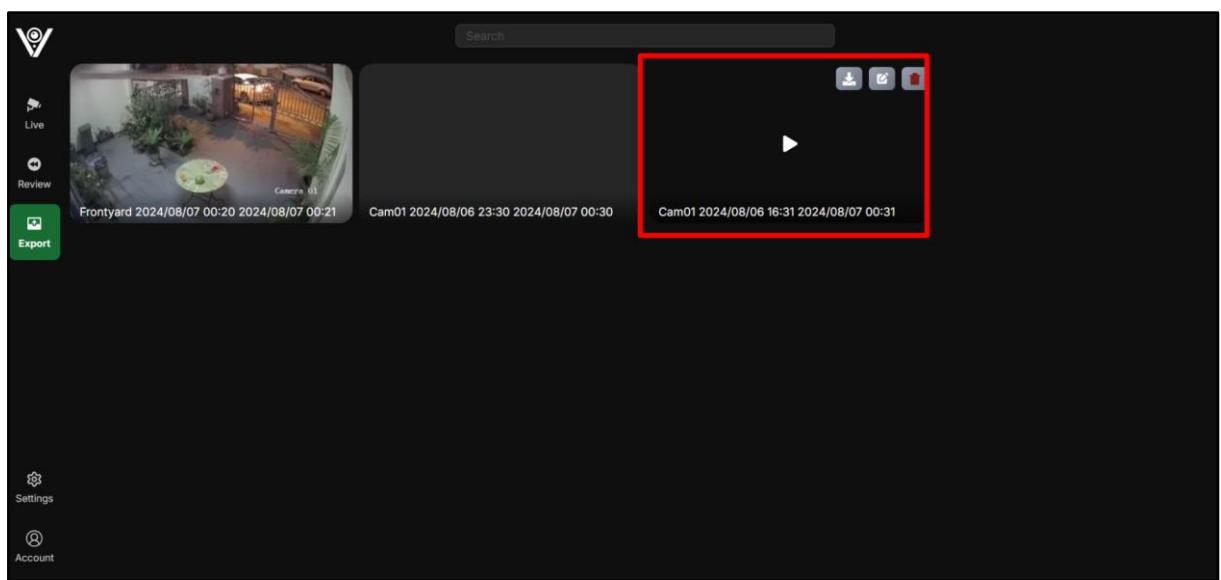


Figure 567. Export camera recording

View it in detail:

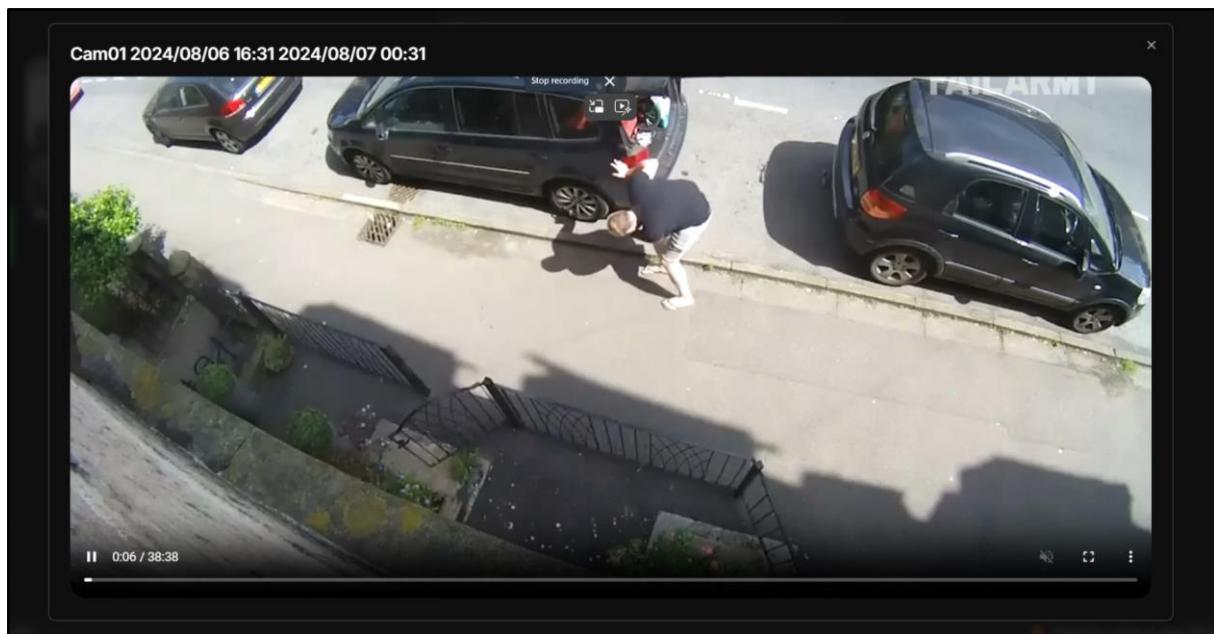


Figure 568. Export camera recording

Another case: **Select from timeline case**

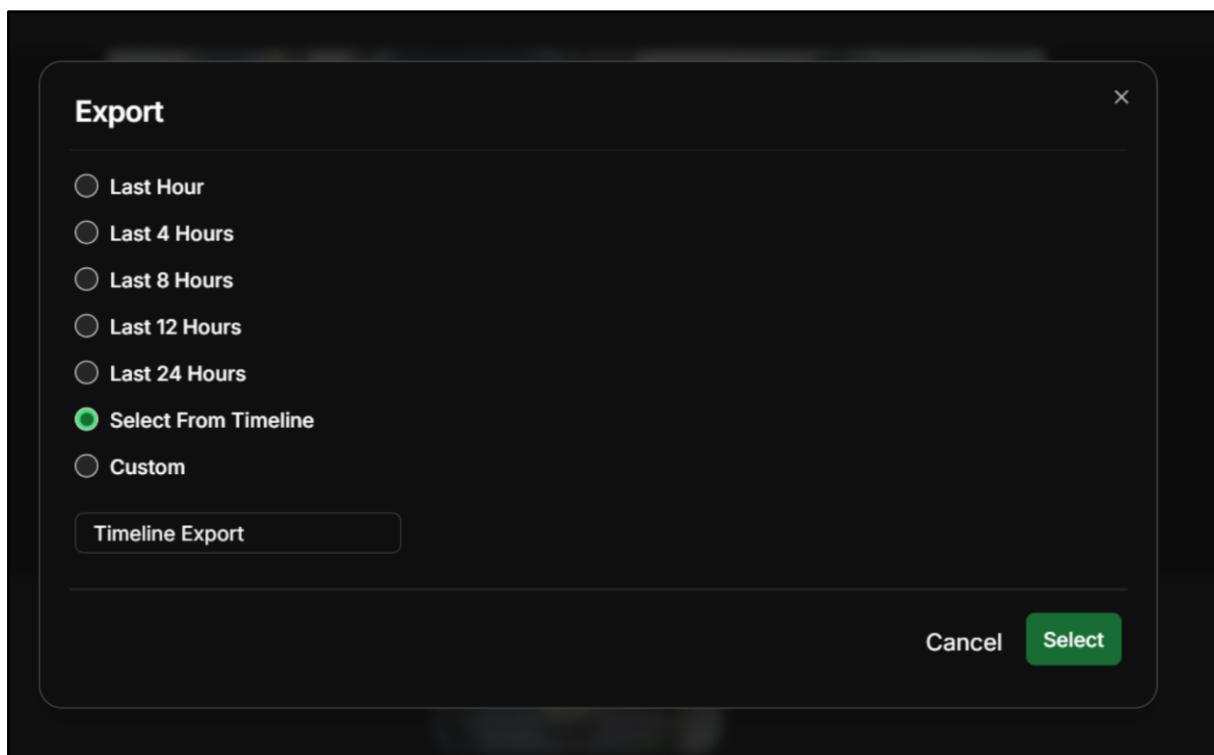


Figure 569. Export camera recording

Click on “Select From Timeline”. Then, click “Select”.

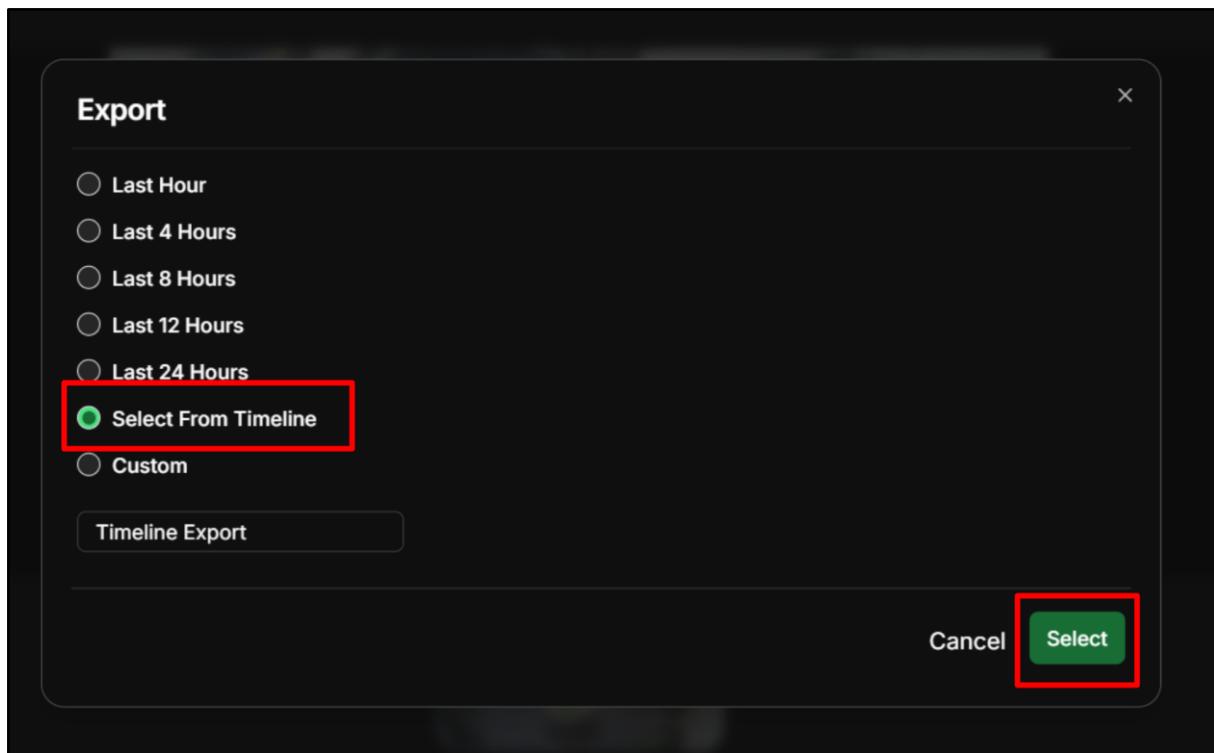


Figure 570. Export camera recording

A new pop-up appears: “Save Export” or “Cancel”:

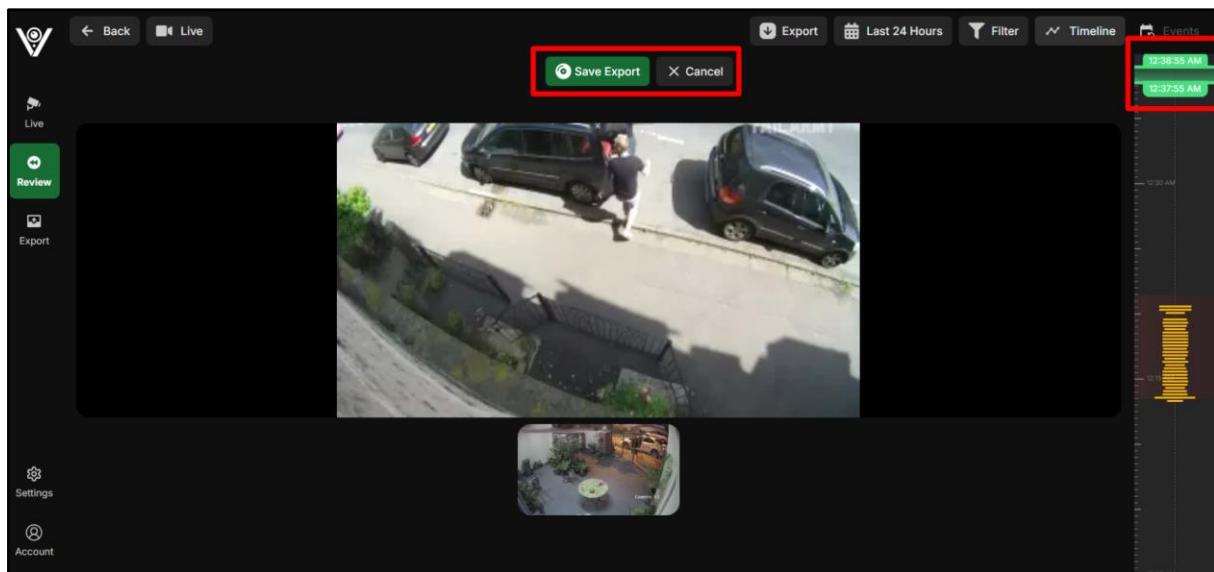


Figure 571. Export camera recording

Scrub the timeline as you want:

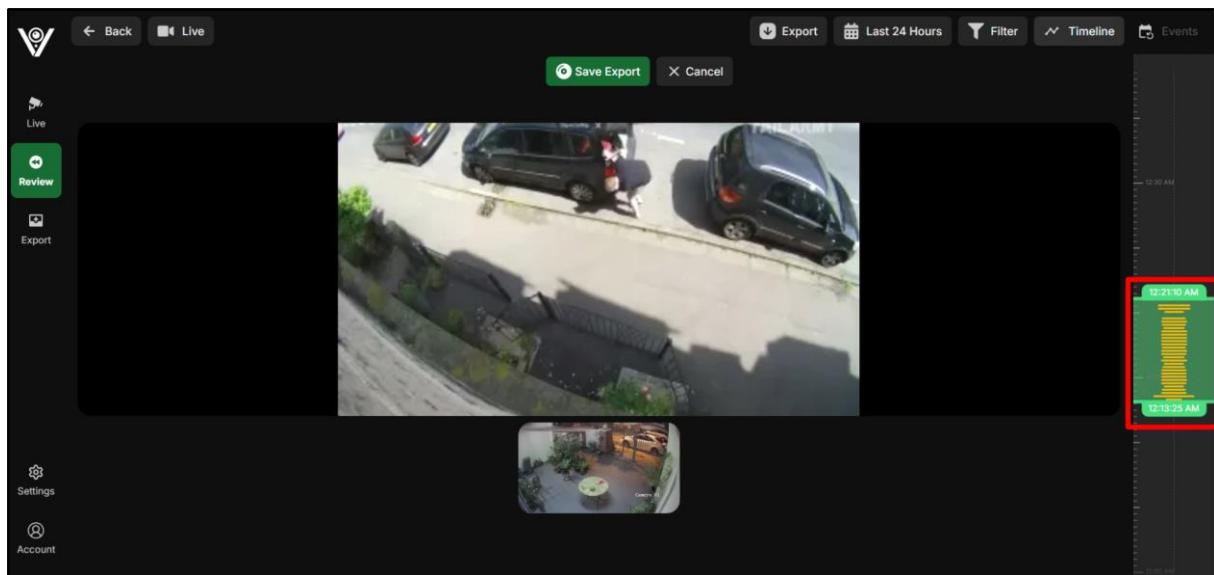


Figure 572. Export camera recording

Click on "Save Export":

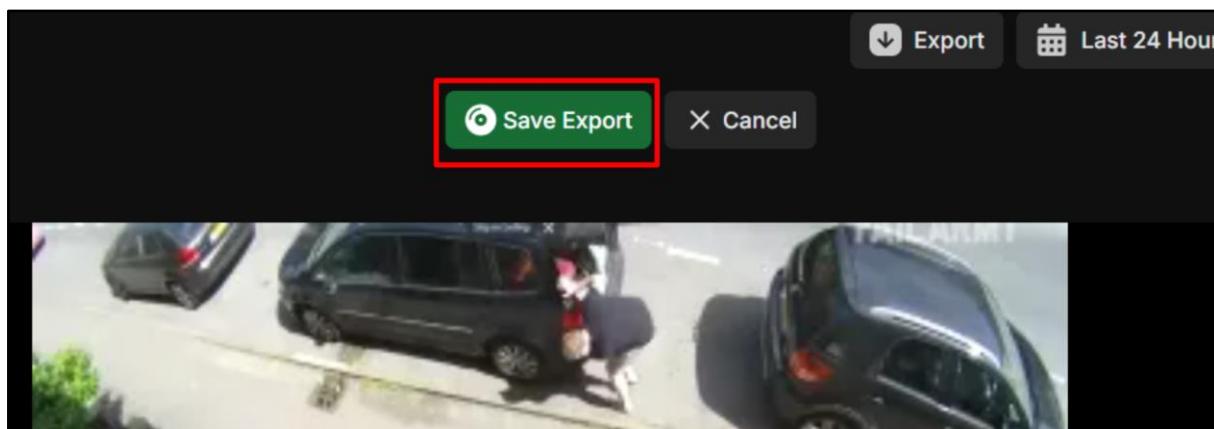


Figure 573. Export camera recording

Then, a message from system pop up "Sucessfully started export. View the file in the Export page":

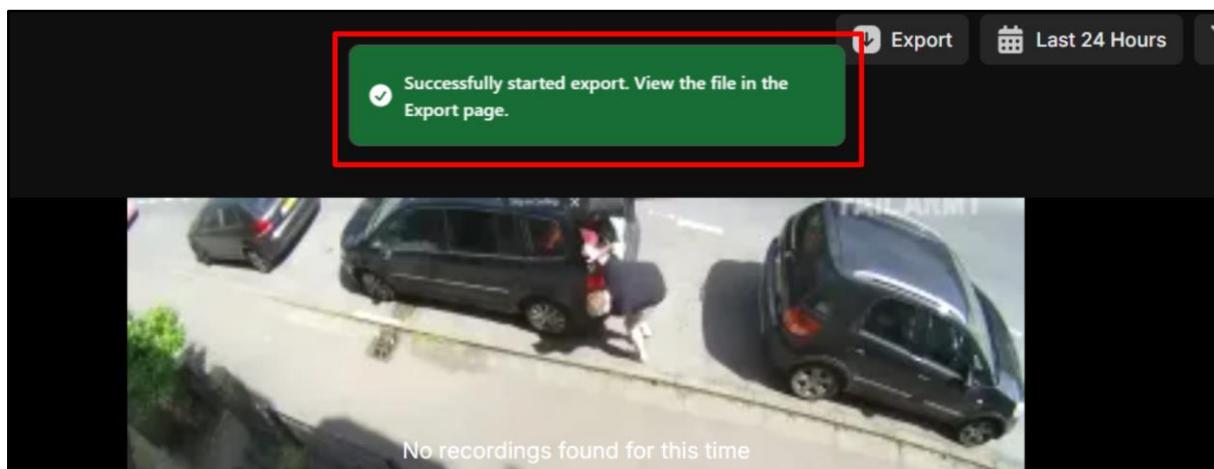


Figure 574. Export camera recording

Timeline export has been created:

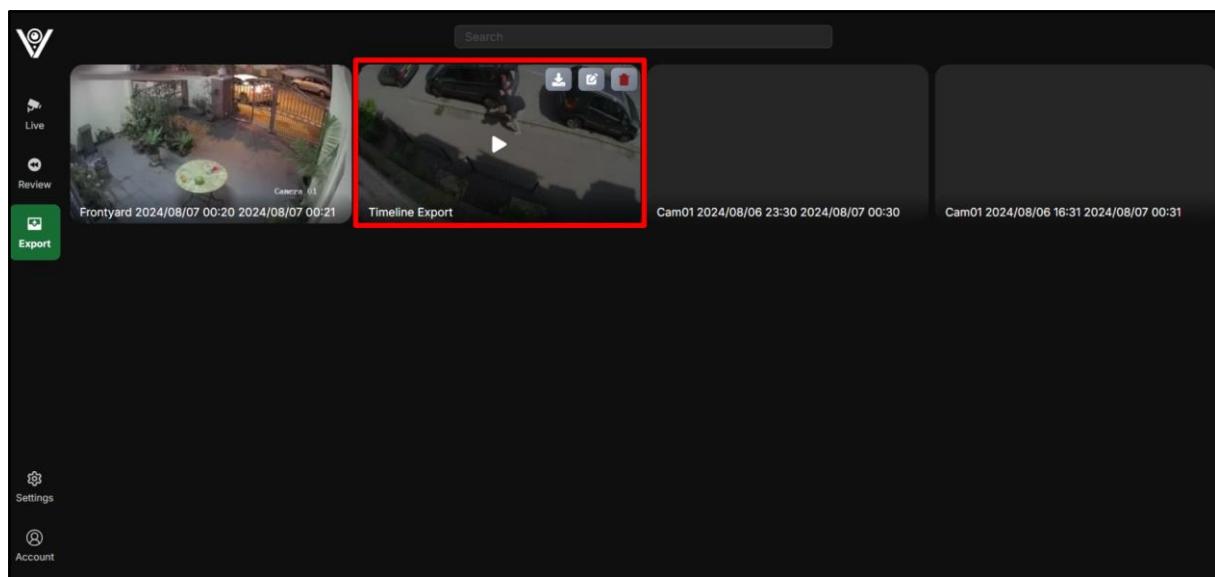


Figure 575. Export camera recording

View Timeline export in detail:

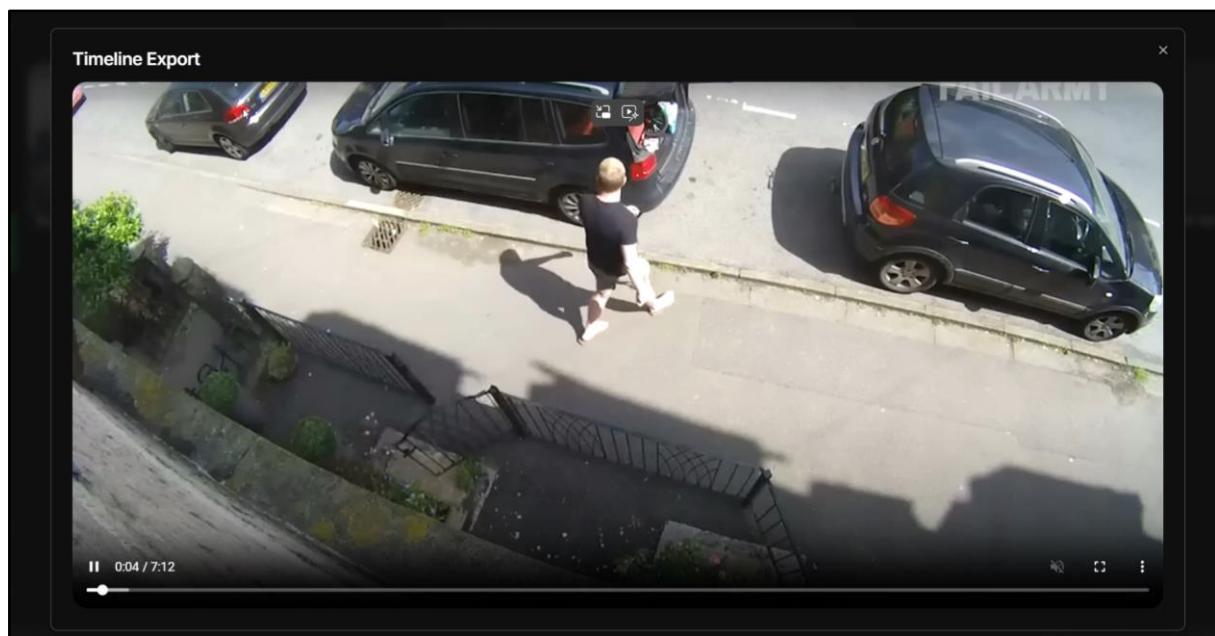


Figure 576. Export camera recording

3.3.2.29 Download export

To download export, go to “Export” page and hover the mouse an the export items you want to download. Then, the download button will appear on the top-right corner of the export items (as highlighted in the image below). Click on download button and downloading process starts.

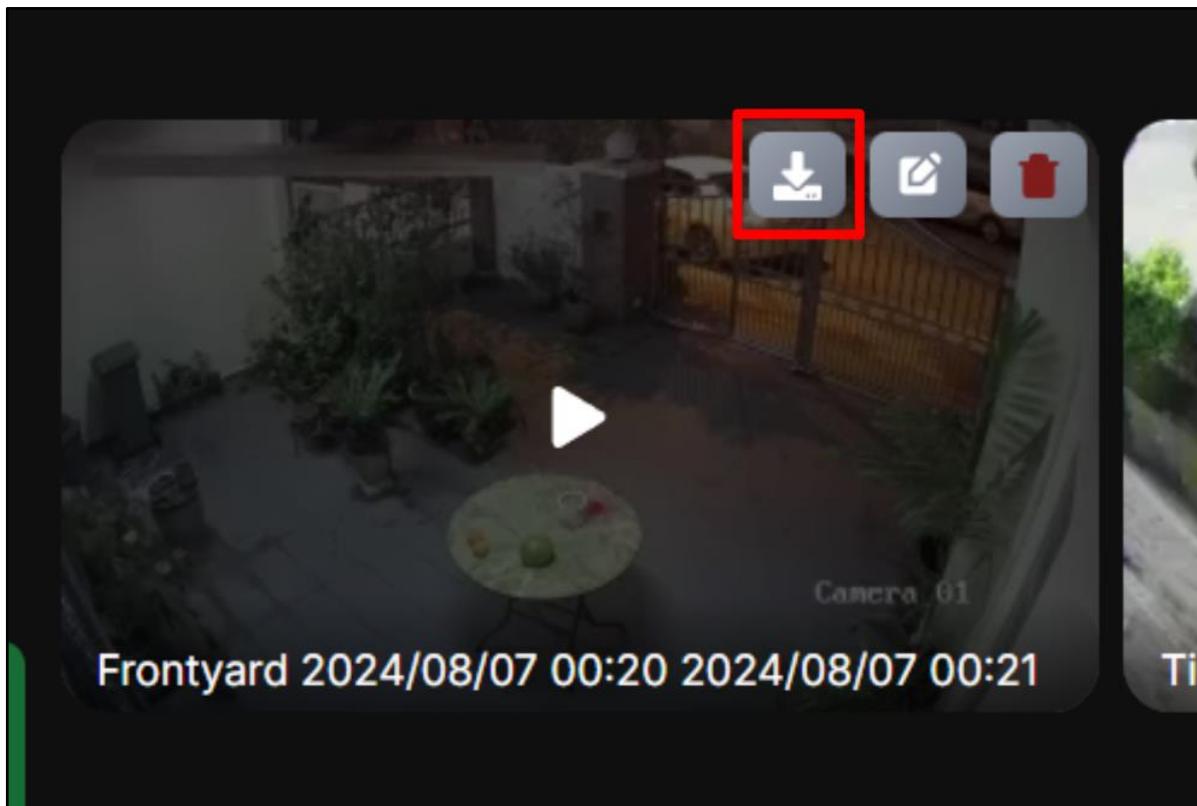


Figure 577. Download export

Wait for awhile, downloading process will finish as soon as possible.

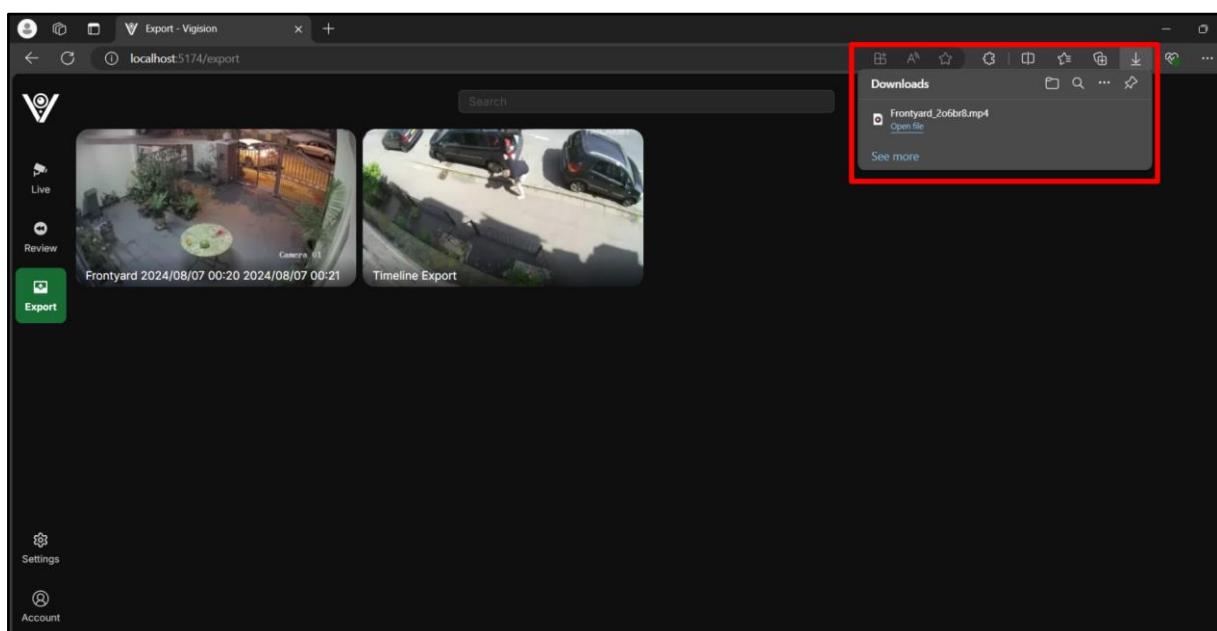


Figure 578. Download export

You can watch the export video using any video player that supports the .mp4 format.

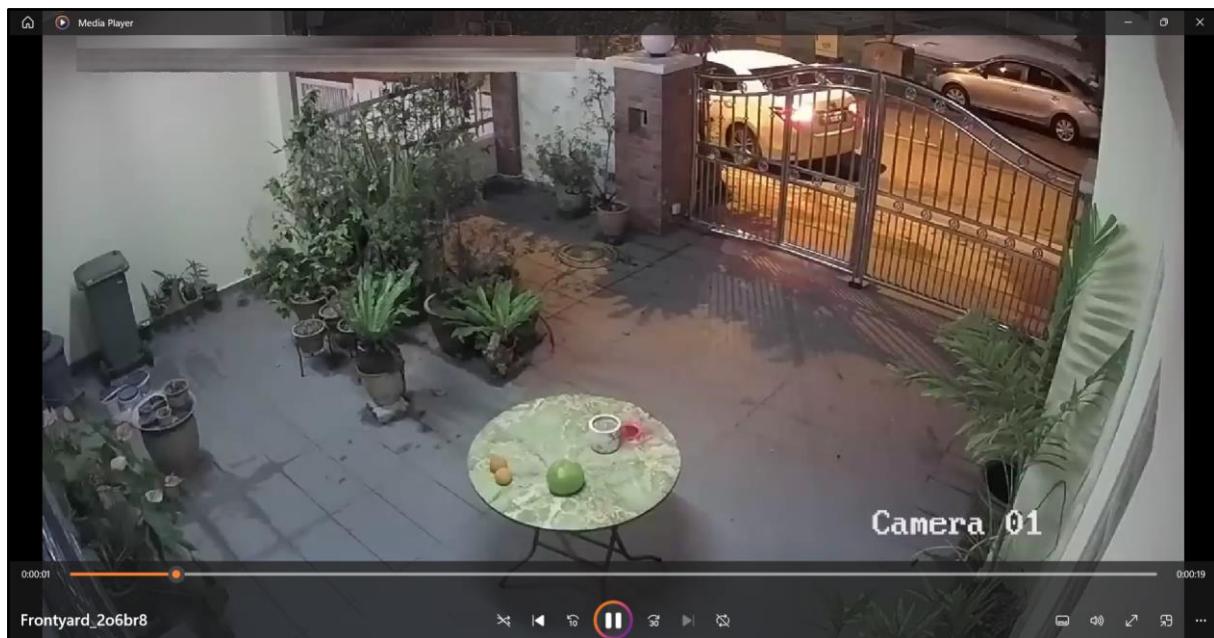


Figure 579. Download export

3.3.2.30 Rename export

Like downloading export above, but instead of clicking button download, you please click on button "Rename" as show in the image below:

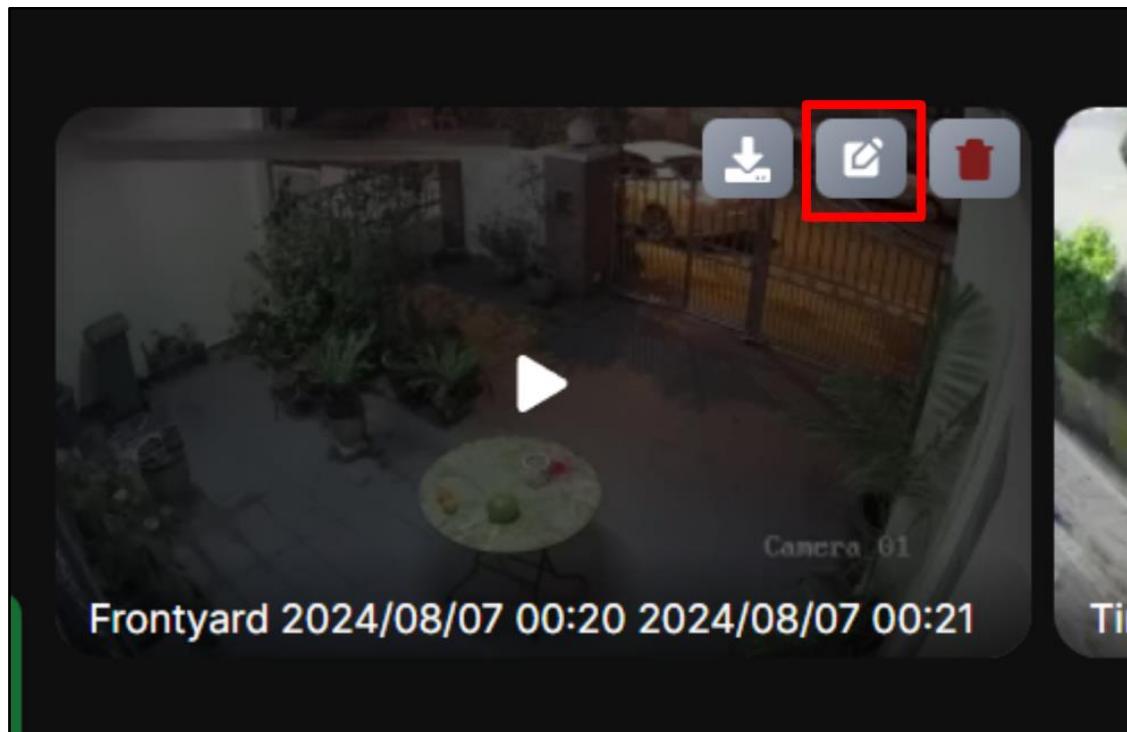


Figure 580. Rename export

Next, “Rename Export” pop-up appears and you please enter new name.

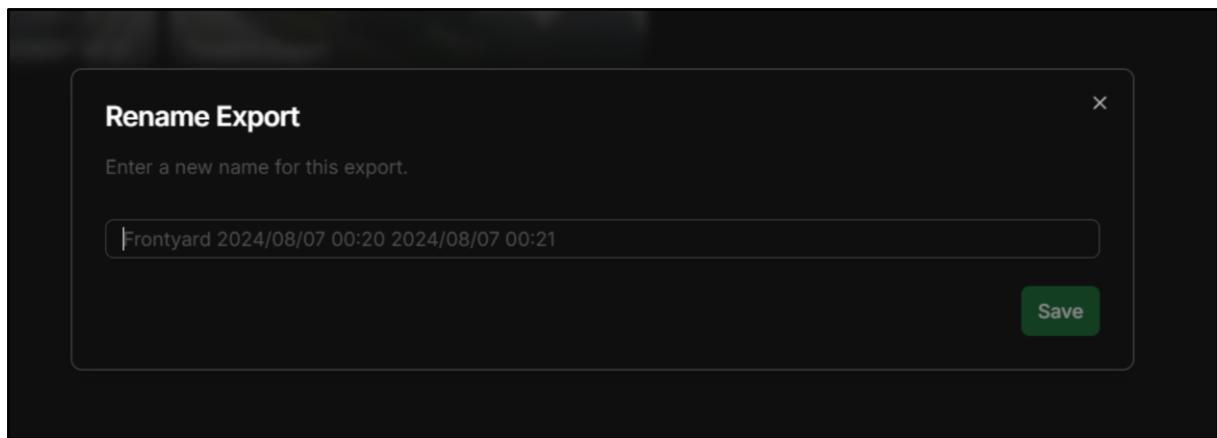


Figure 581. Rename export

Click save, and see the the export with its new name.

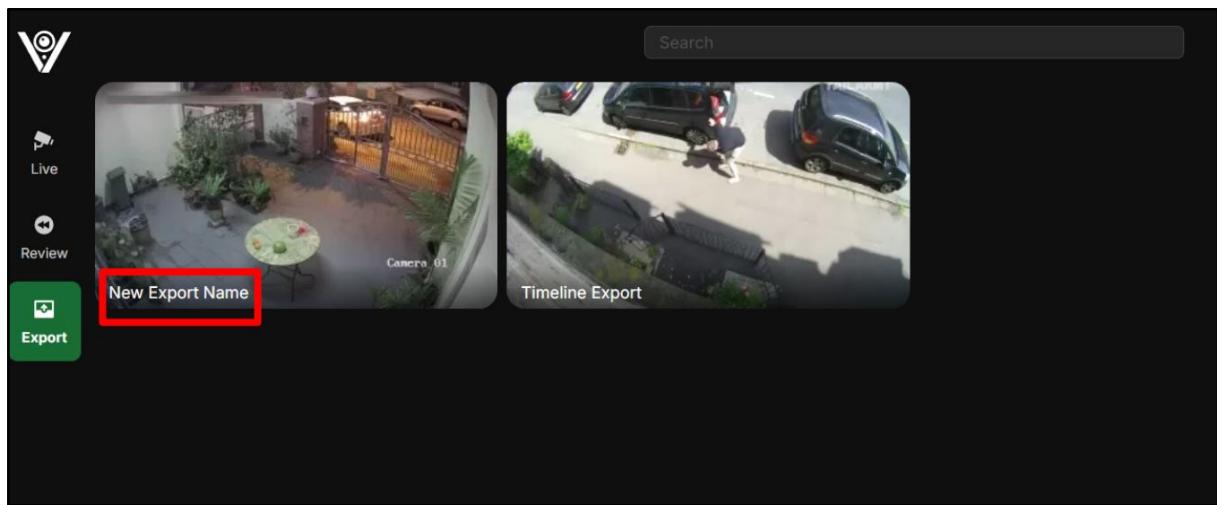


Figure 582. Rename export

3.3.2.31 Delete export

Similarly, click on the button “Delete” first, then confirm delete and the mission is done:

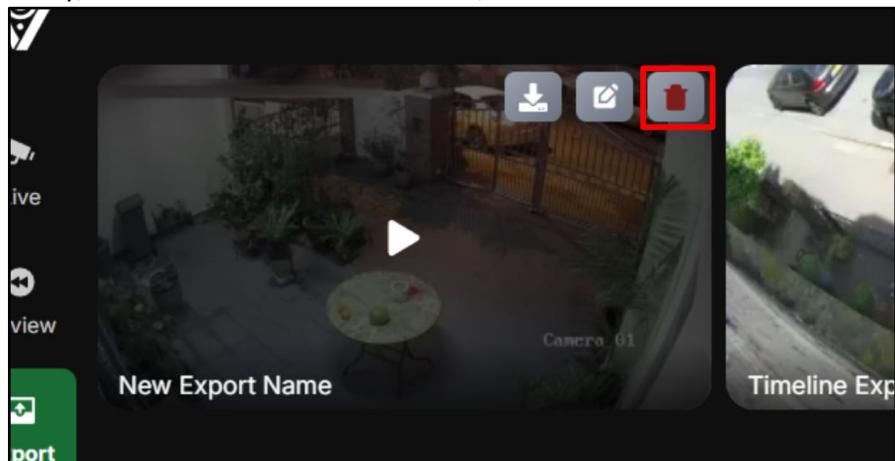


Figure 583. Delete export

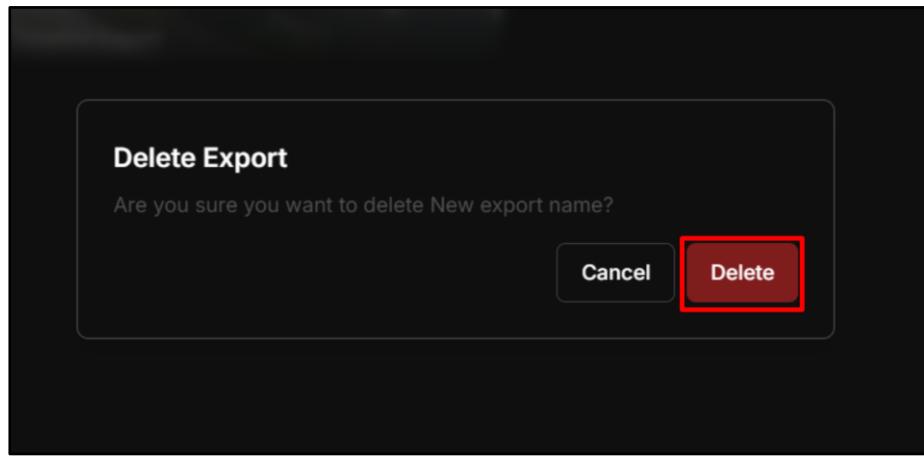


Figure 584. Delete export

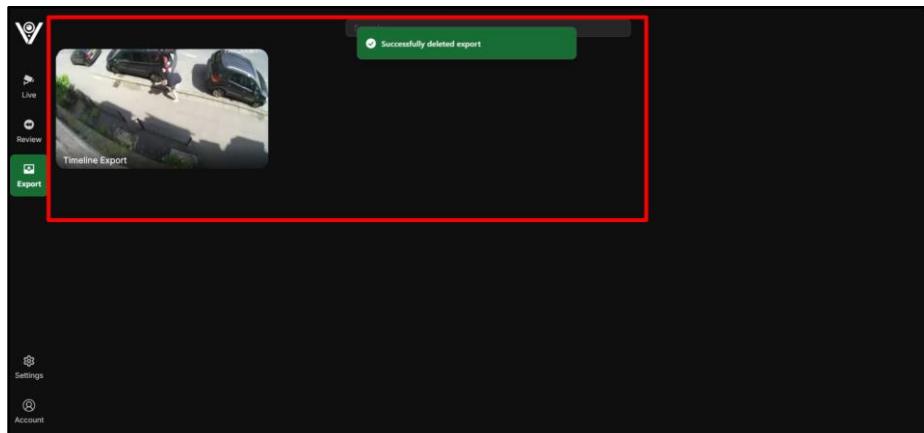


Figure 585. Delete export

3.3.2.32 Config

3.3.2.32.1 View configuration

To view configuration, click on button “Setting” in the sidebar

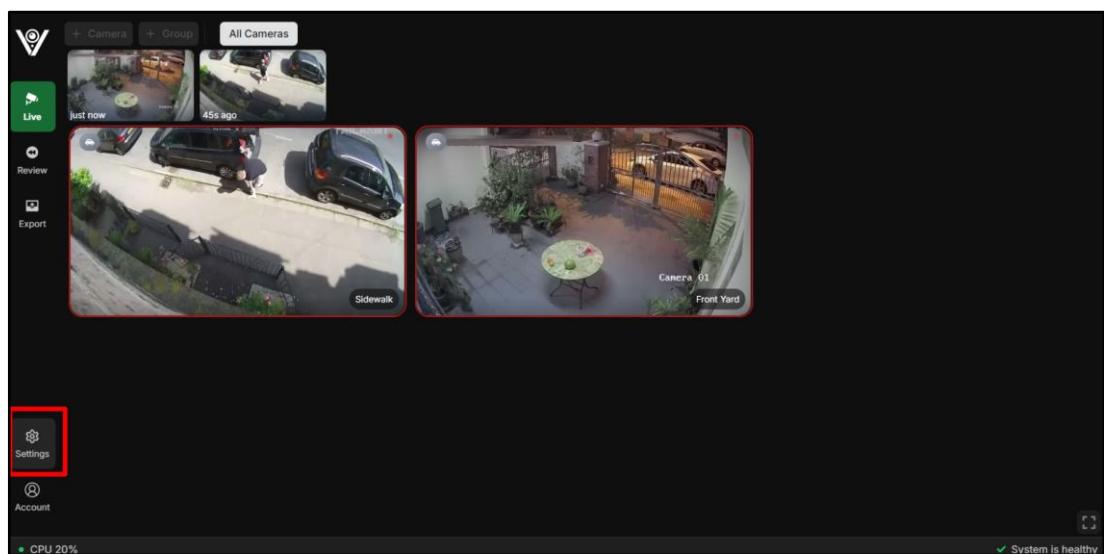


Figure 586. View configuration

Next, click on the button “Configuration”.

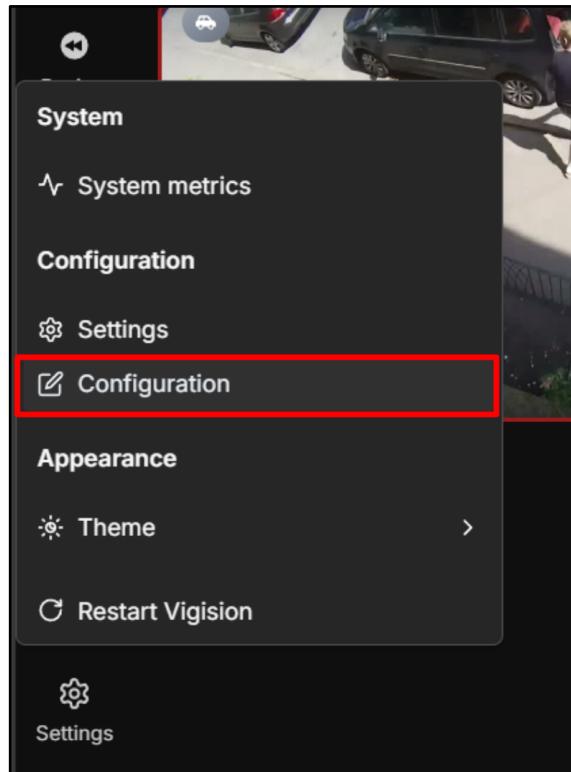


Figure 587. View configuration

After that, you will see the “Configuration” page as below:

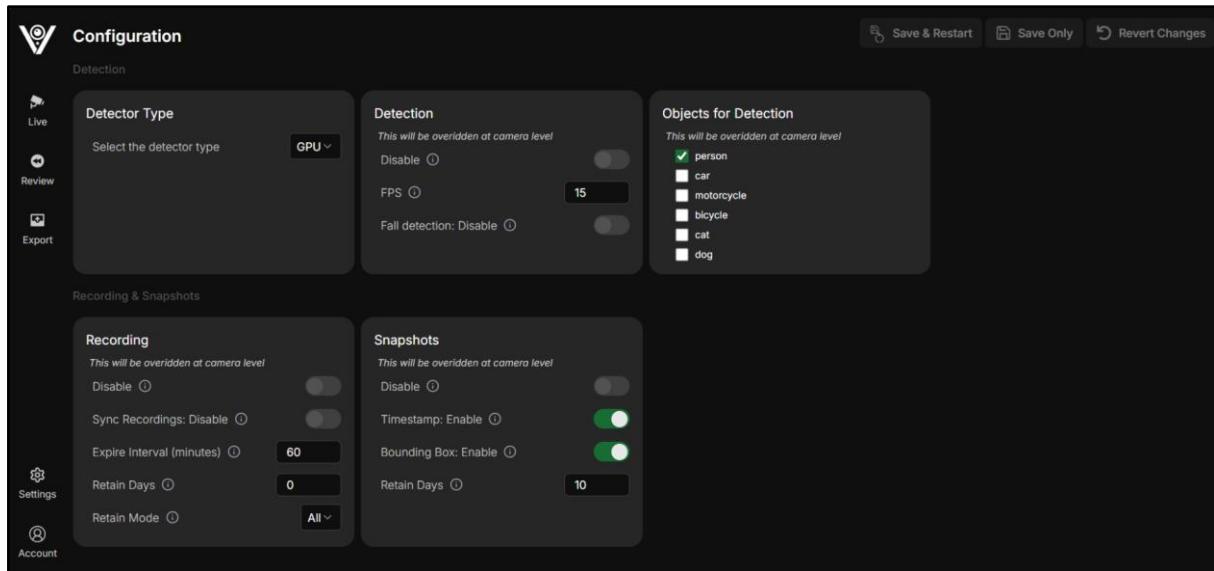


Figure 588. View configuration

There are 2 groups of configuration: 1. **Detection** and 2. **Recording & Snapshots**

In **Detection** group, you can configure **Detector Type**, **Detection’s options**, **Object for Detection**.

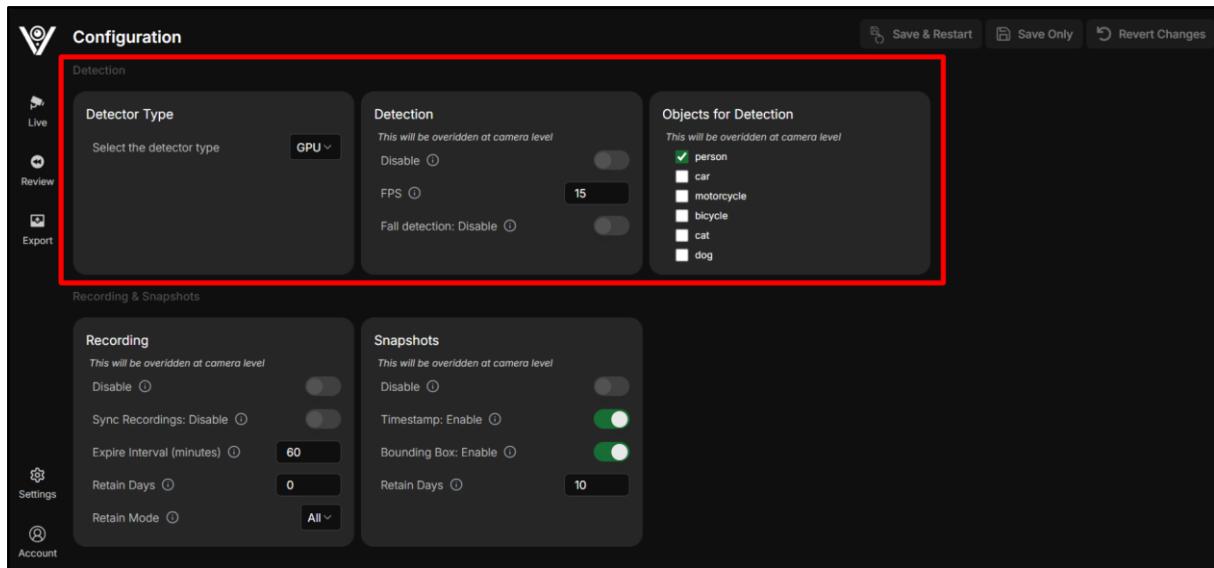


Figure 589. View configuration

In **Recording & Snapshots** group, you can configure options related to recording and snapshots.

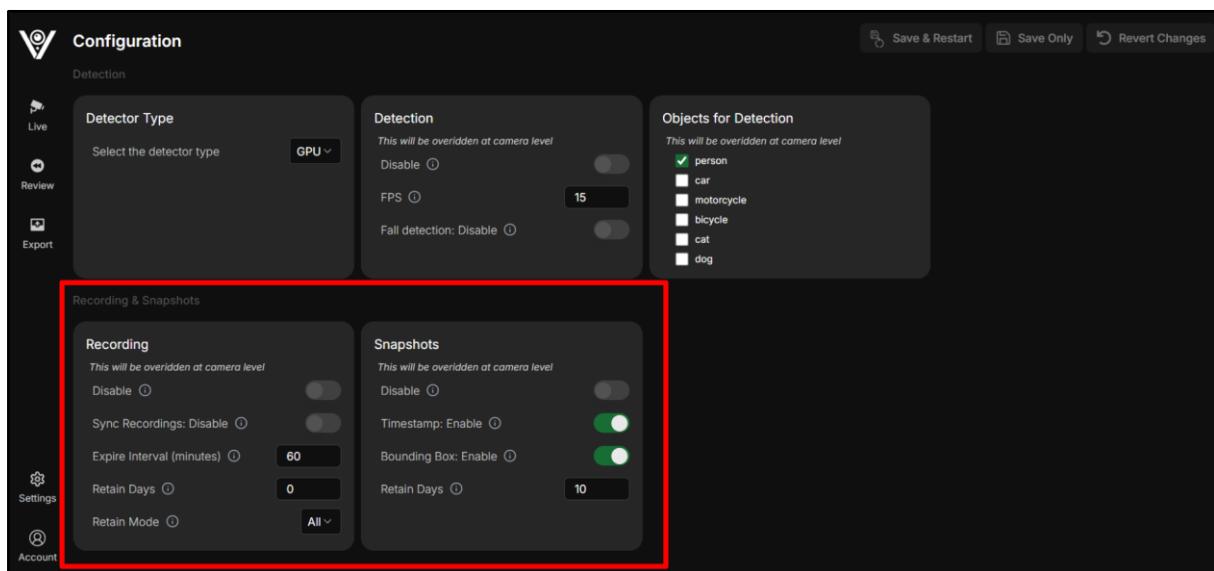


Figure 590. View configuration

The below image demonstrates options for detector type: GPU, CPU. Select a detector type that most suitable for you system.

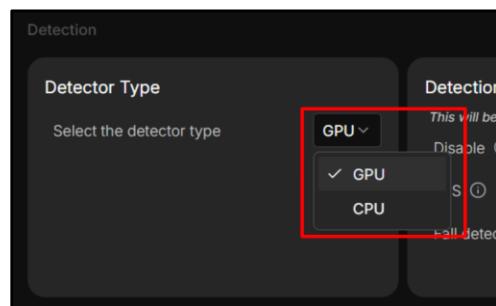


Figure 591. View configuration

Next, go to **Detection block**. Please note that these configurations will be overridden at camera level.

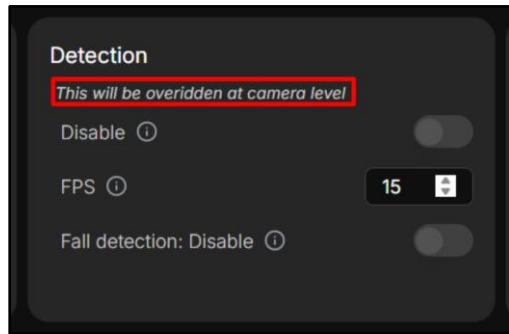


Figure 592. View configuration

You can enable/disable detection:

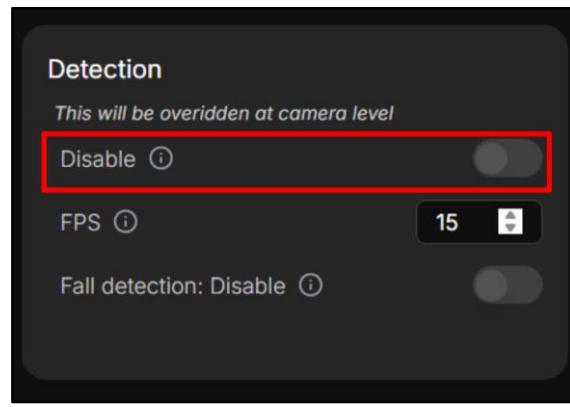


Figure 593. View configuration

Or you set an appropriate FPS for cameras:

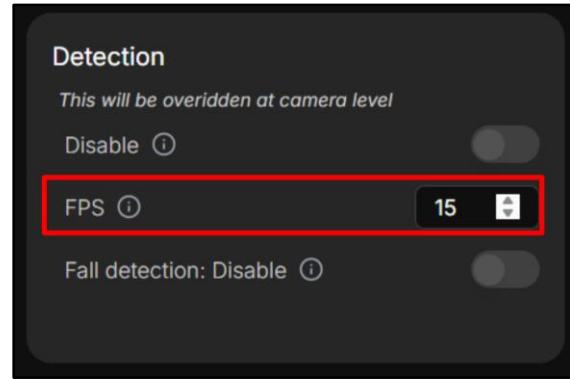


Figure 594. View configuration

Or you can enable/disable fall detection feature:

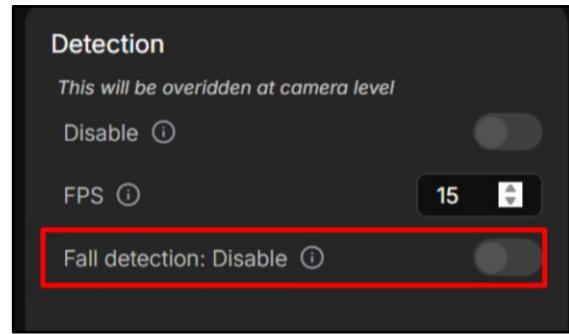


Figure 595. View configuration

You can select which objects you want to detect. The object list includes: person, car, motorcycle, bicycle, cat, dog. In the future there are more object will be added.

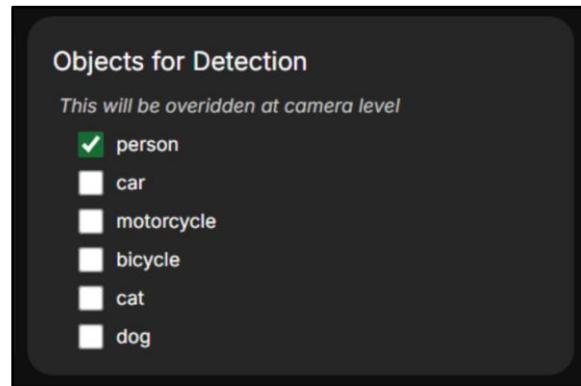


Figure 596. View configuration

Go to group **Recording & Snapshots**, you can enable/disable recording feature:

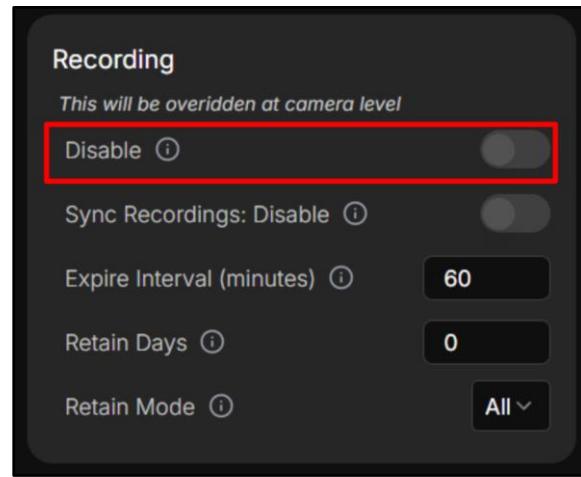


Figure 597. View configuration

Or you can enable/disable synchronize recordings:

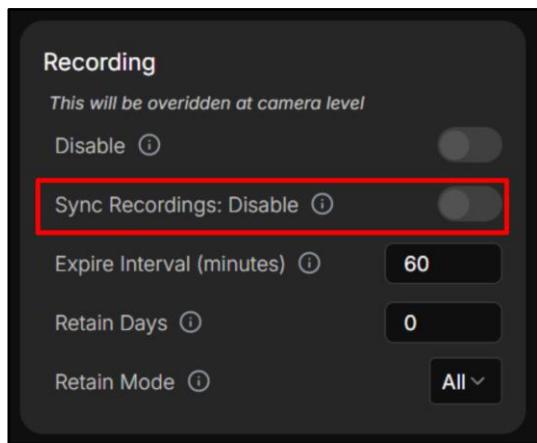


Figure 598. View configuration

Or you can set **Expire Interval** (in minutes):

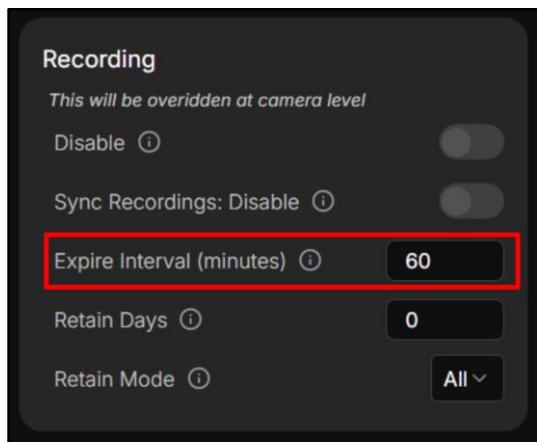


Figure 599. View configuration

Another option is to set up retain days for recordings:

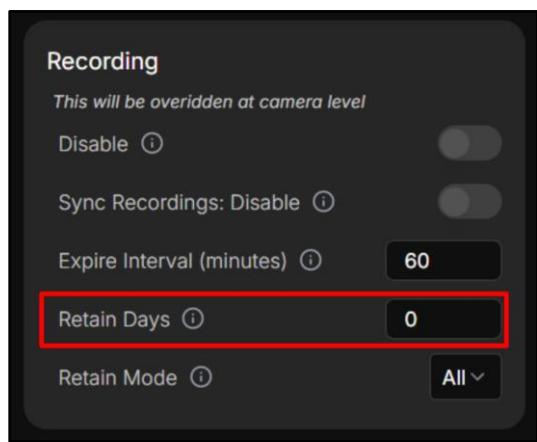


Figure 600. View configuration

Or setup retain mode for recording:

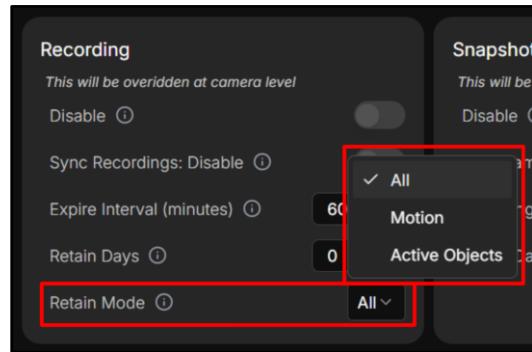


Figure 601. View configuration

3.3.2.32.2 Revert configuration change

If you want to revert configuration change, click on the button “Revert Changes” on the top-right corner of the Configuration page.

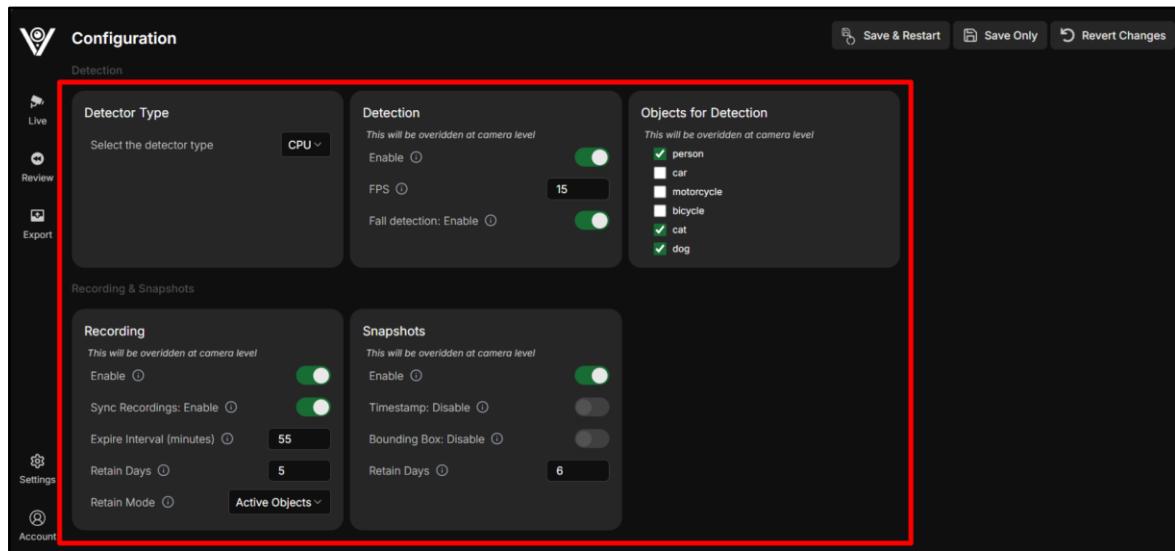


Figure 602. Revert configuration change

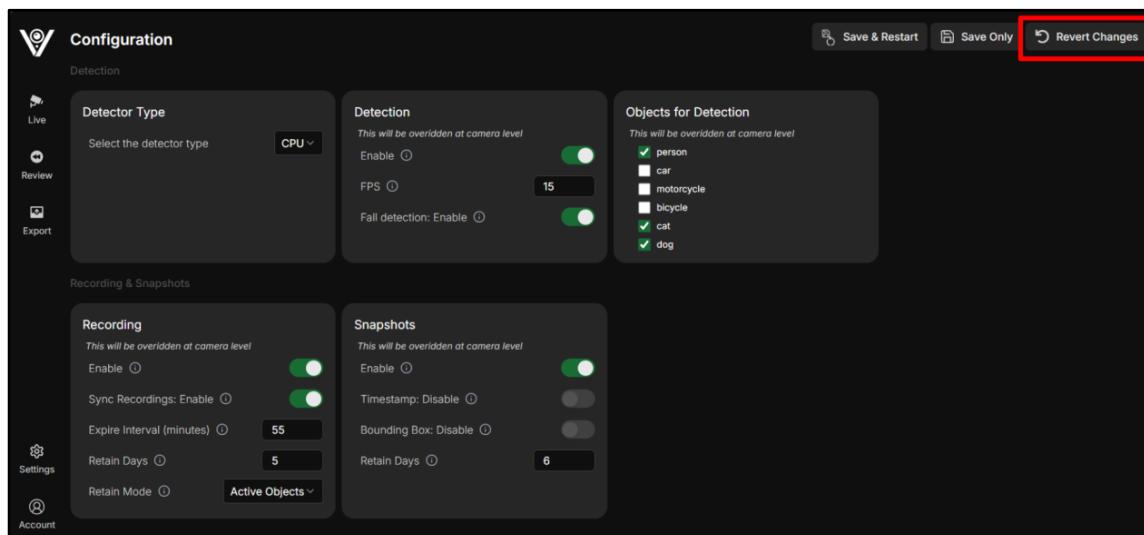


Figure 603. Revert configuration change

Next, click on the button “Revert”

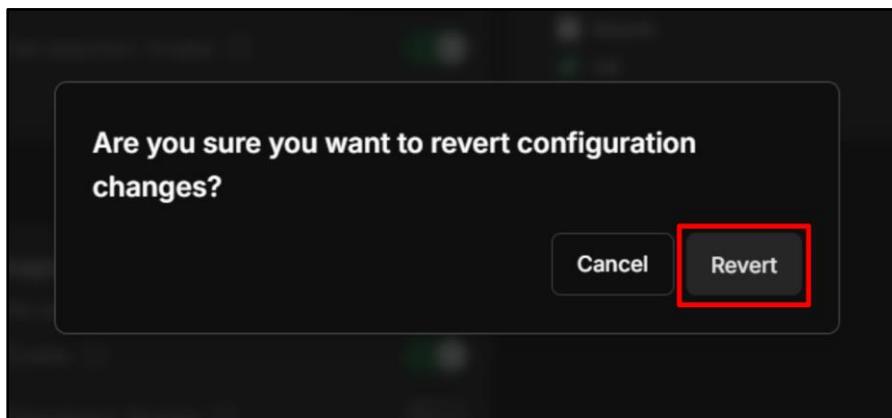


Figure 604. Revert configuration change

A message will pop up say congratulation has been reverted to the original values:

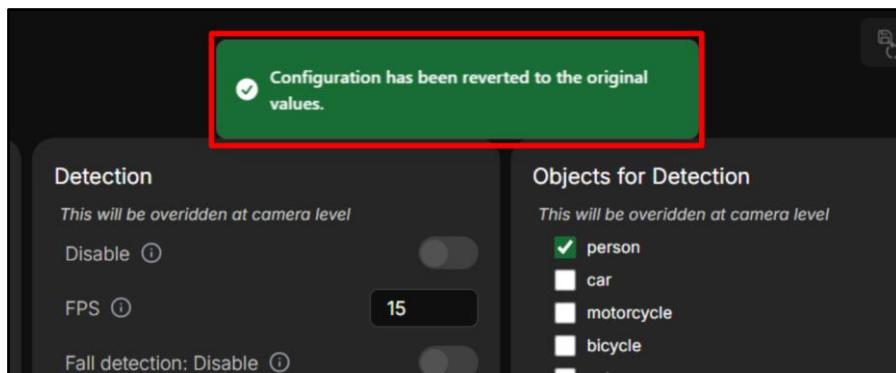


Figure 605. Revert configuration change

You can see that configuration has been reverted:

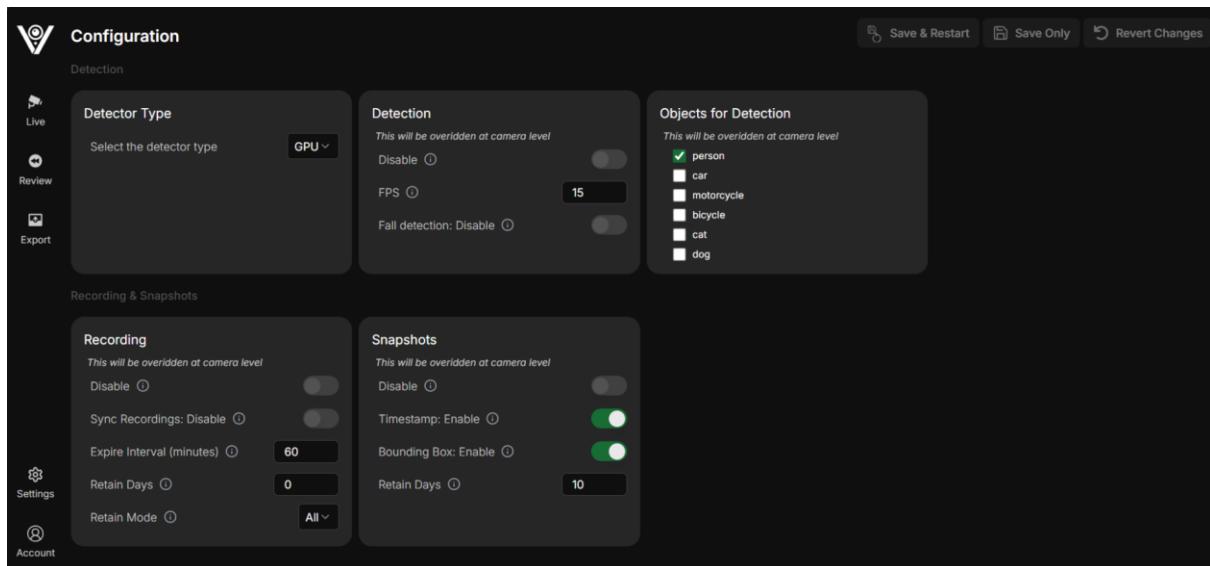


Figure 606. Revert configuration change

3.3.2.32.3 Save configuration

After update the configuration, you can save them or save then restart.

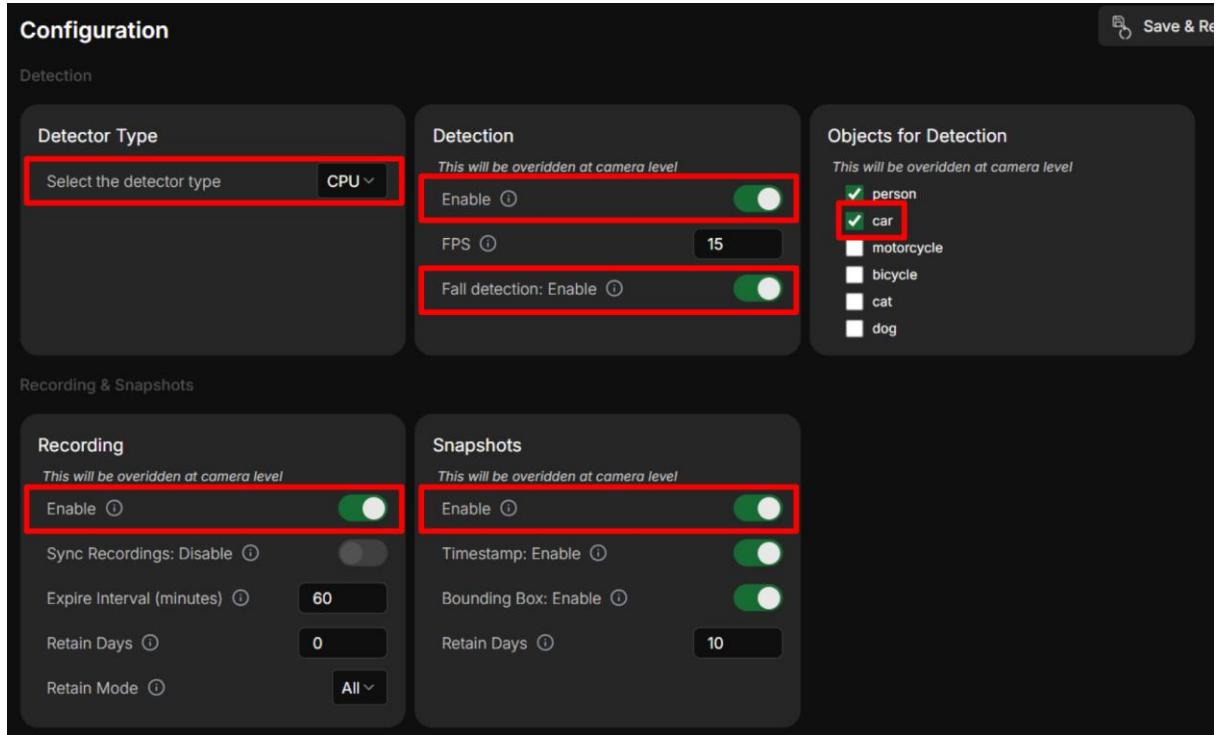


Figure 607. Save configuration

If you want to save, not restart the system now, click on the button “Save Only”.

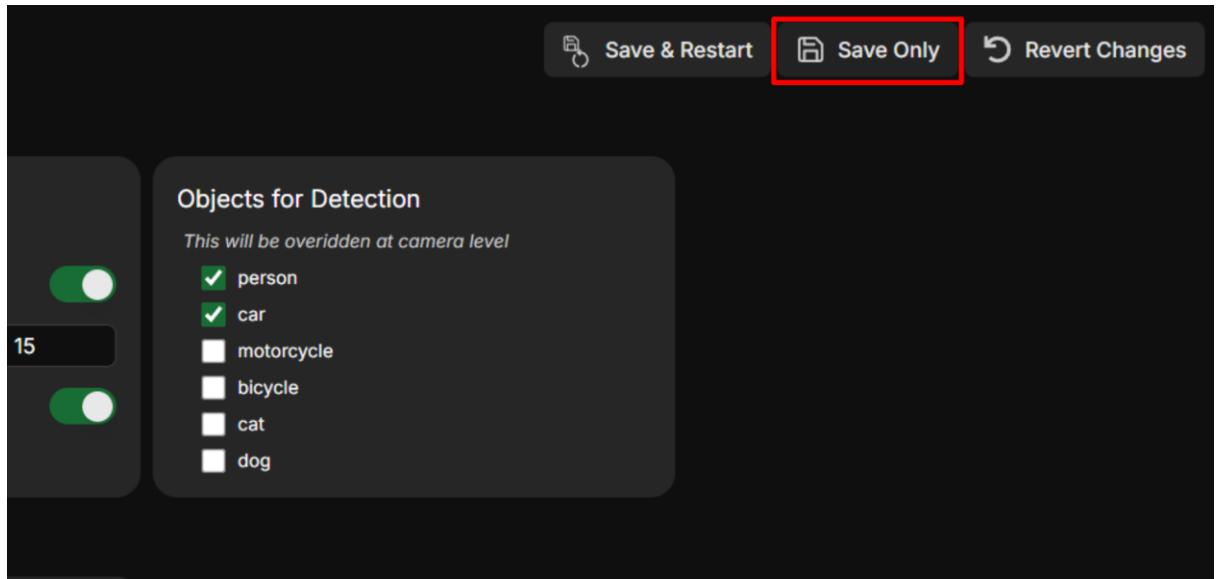


Figure 608. Save configuration

Subsequently, the system send a message to inform configuration has been saved.

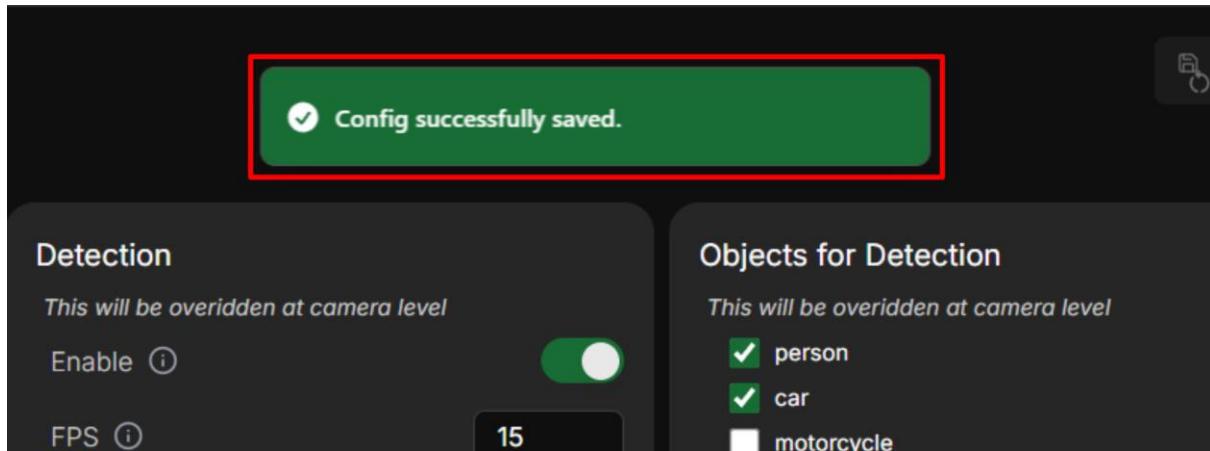


Figure 609. Save configuration

You can refresh the page to check if it has been updated or not

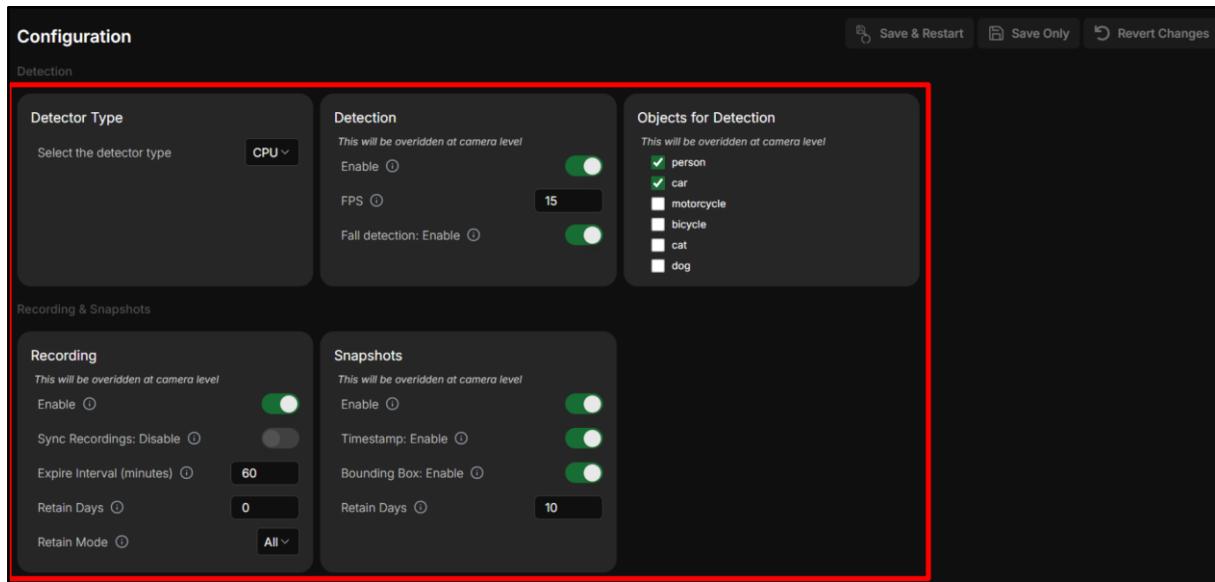


Figure 610. Save configuration

3.3.2.32.4 Save configuration and restart

After update the configuration, you can **save** them or **save then restart**.

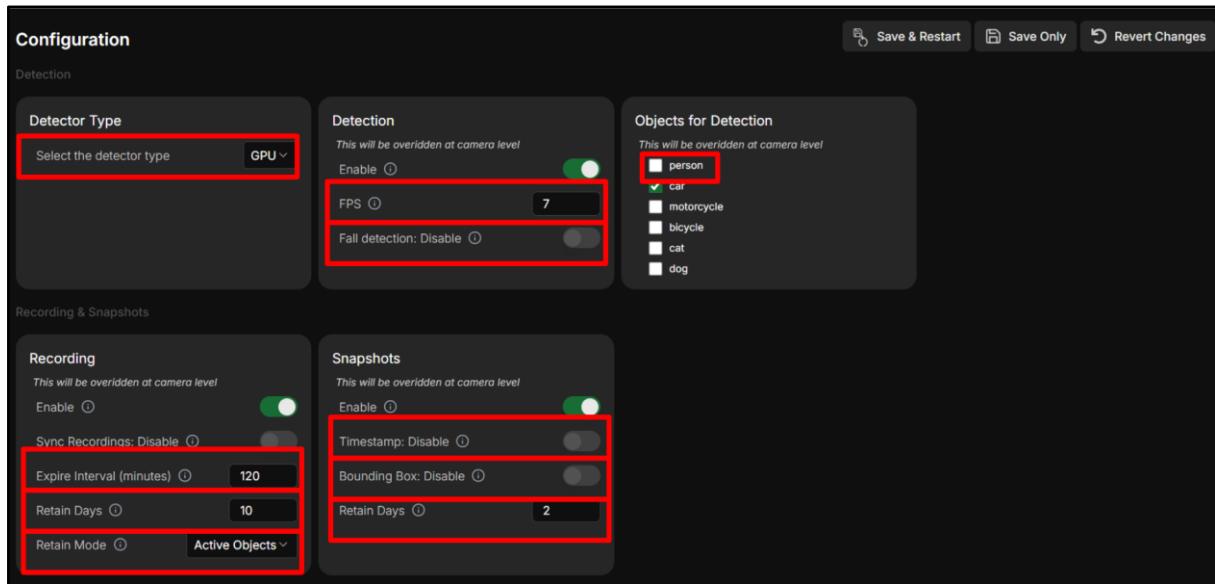


Figure 611. Save configuration and restart

If you want to save then restart the system to apply new configuration, please click on the button “Save & Restart”.

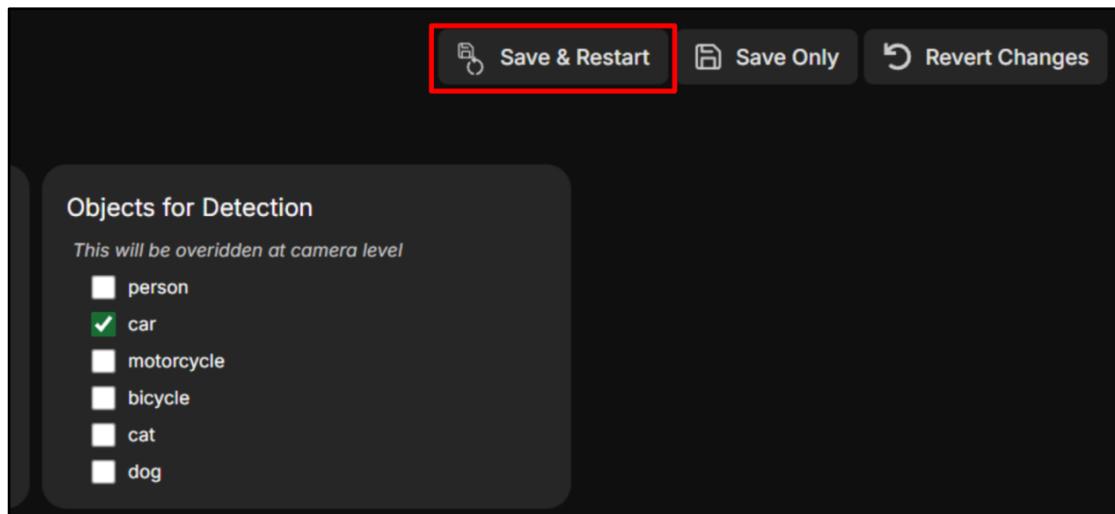


Figure 612. Save configuration and restart

After that, click Restart to finish.

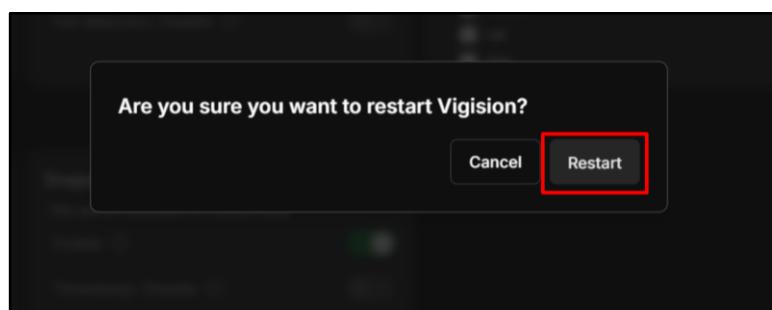


Figure 613. Save configuration and restart

Next, the system will start to restart itself.

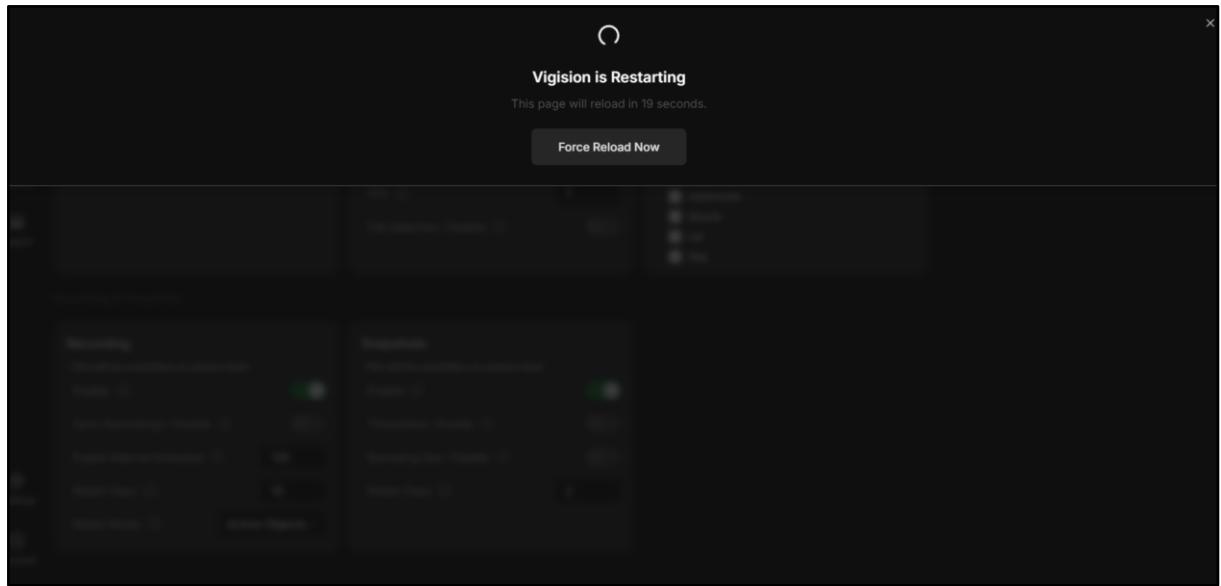


Figure 614.

You can go to “Configuration” page to check if the configuration. And of course the new configuration has been applied.

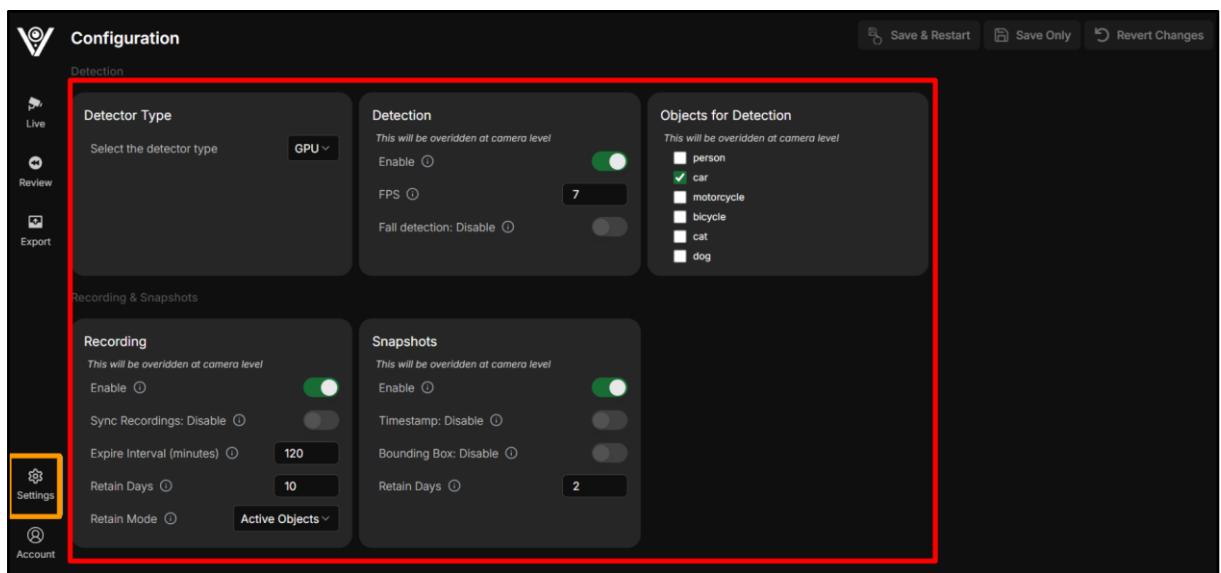


Figure 615. Save configuration and restart

One thing to take note is that the configuration in “Configuration” page is also called default configuration for a new camera. This means that when you click on button “Add” to add new camera,

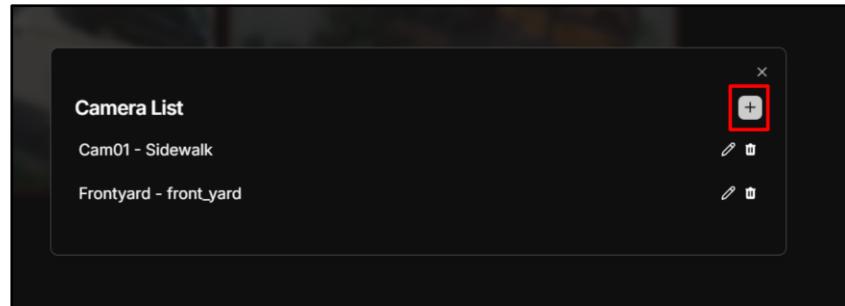


Figure 616. Save configuration and restart

the default configuration will apply to this form:

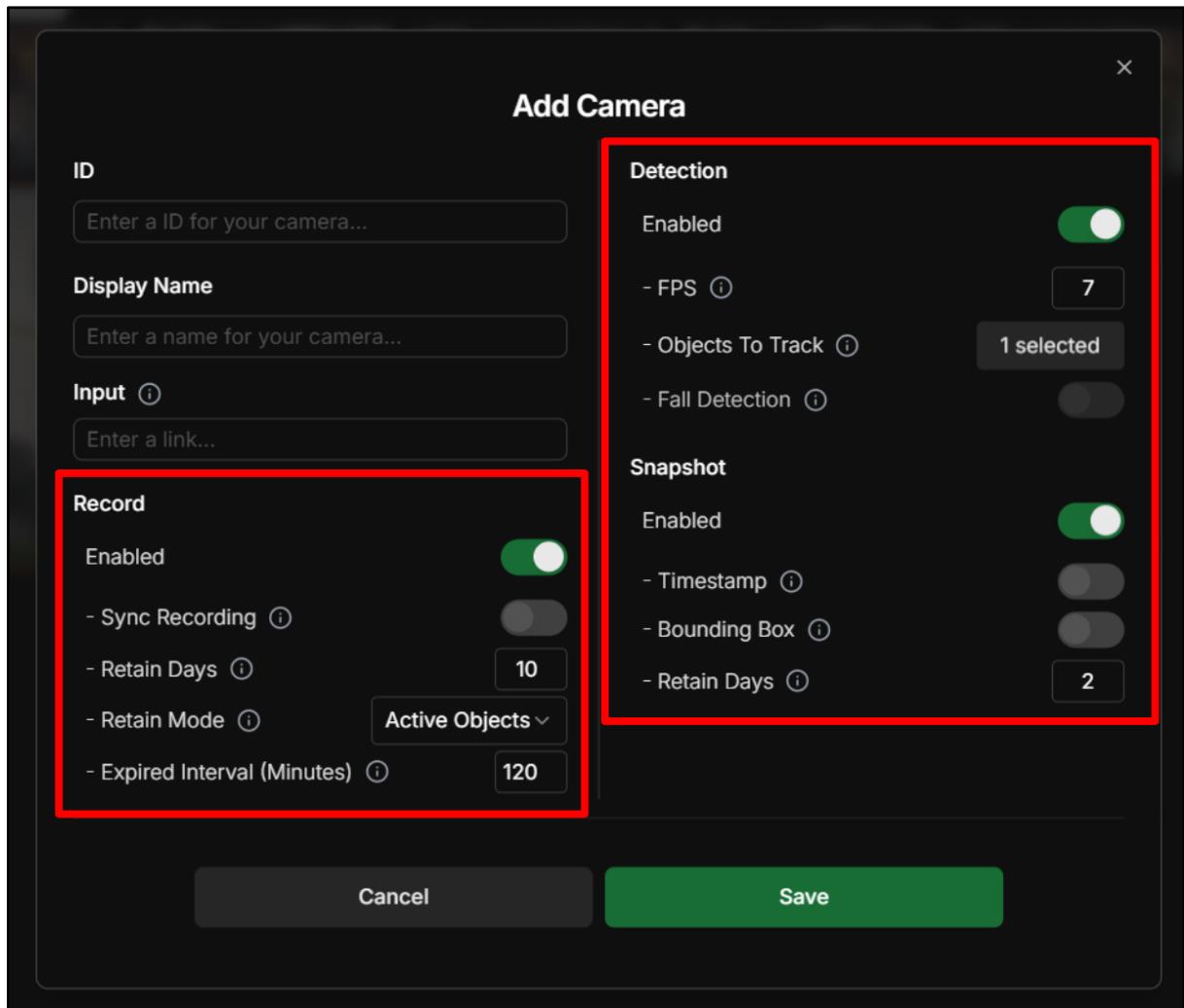


Figure 617. Save configuration and restart

Another thing to take note is that when you update default configuration in “Configuration” page, the saved cameras will not be affected.

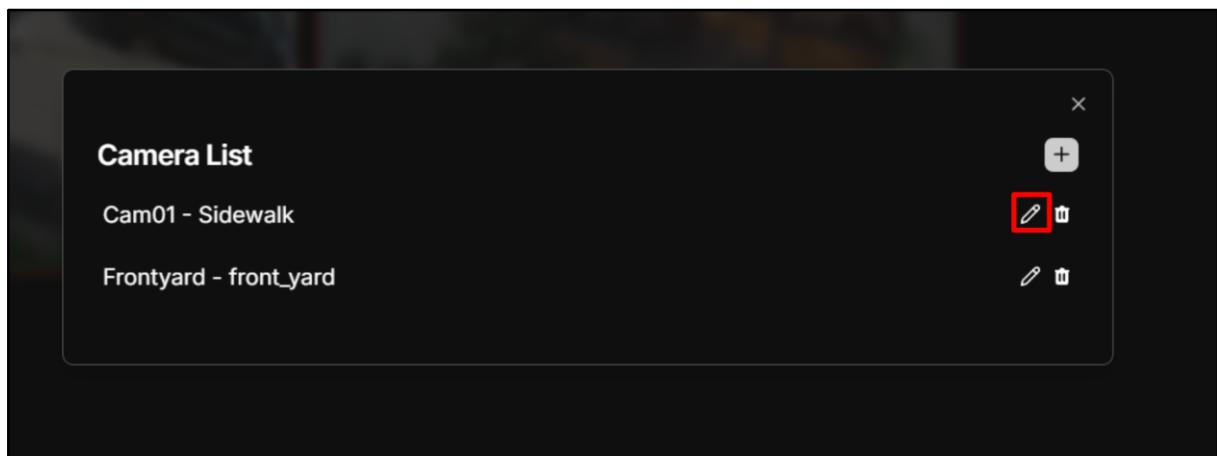


Figure 618. Save configuration and restart

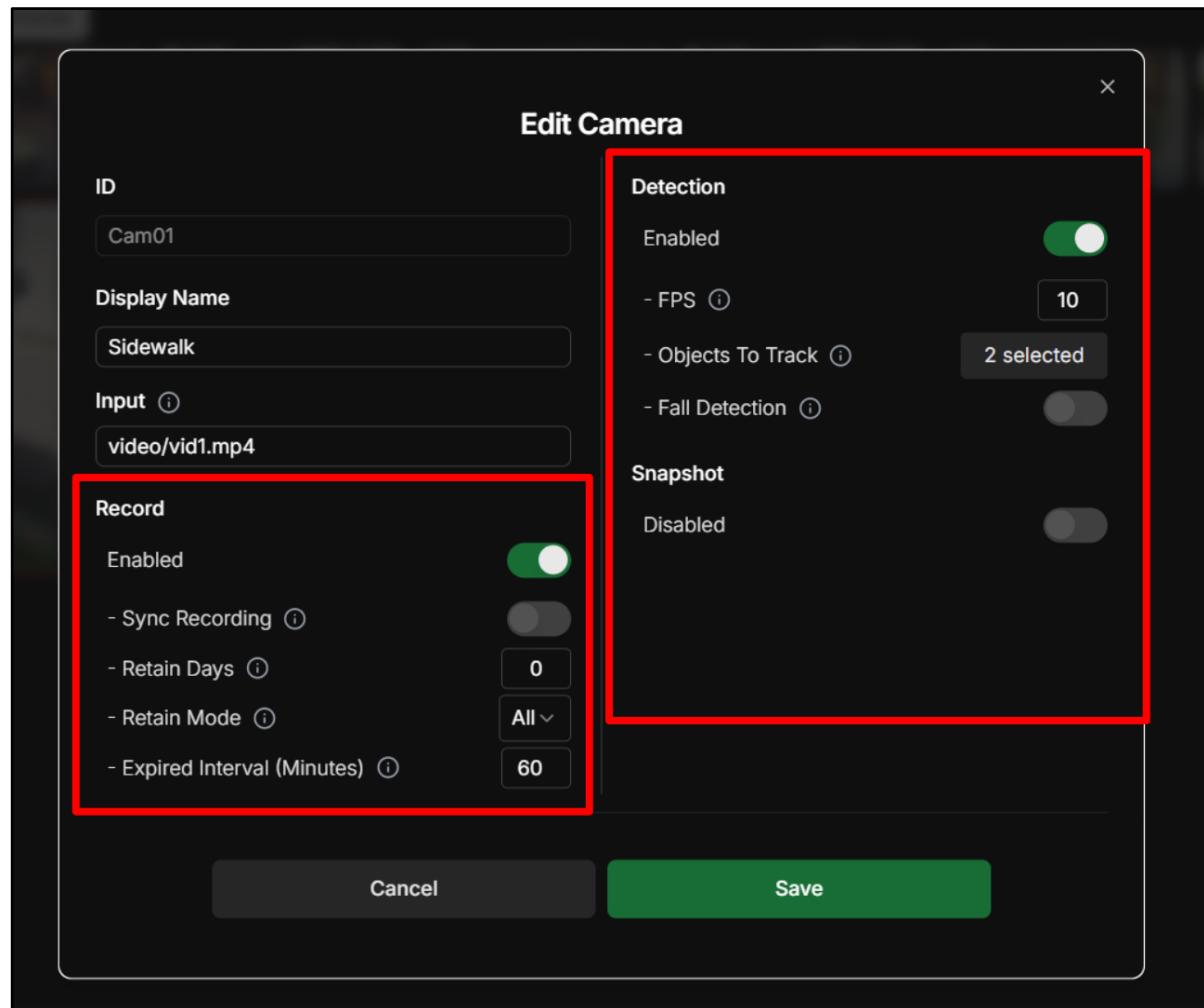


Figure 619. Save configuration and restart

3.3.2.33 View system metrics

To view system metrics, click on the button “**Settings**”, then, **System metrics**

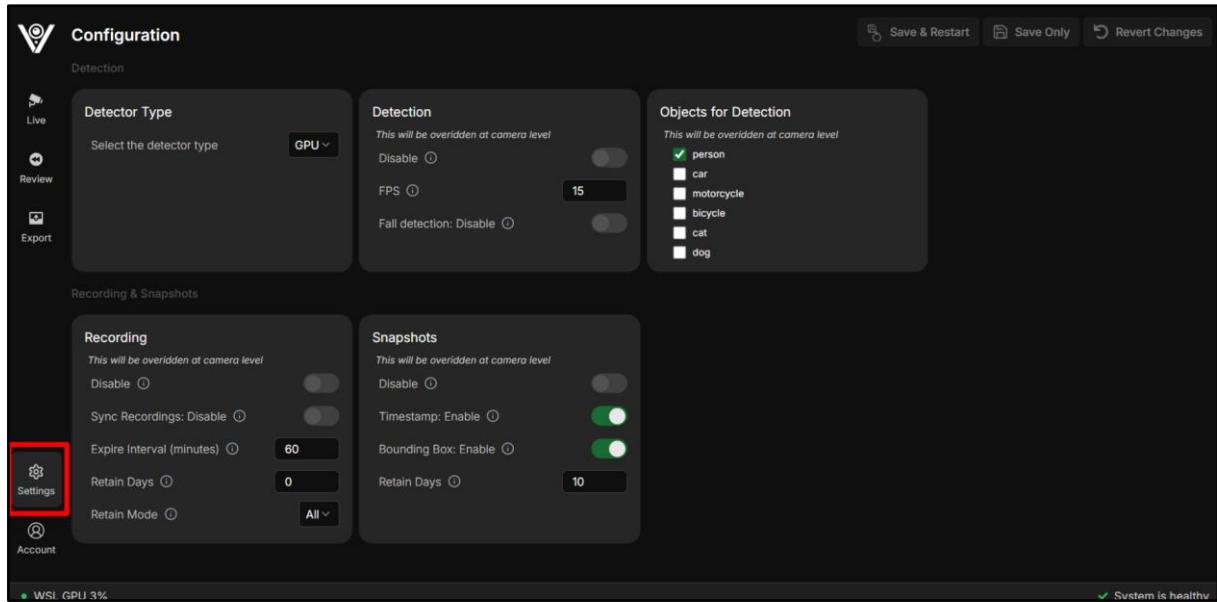


Figure 620. View system metrics

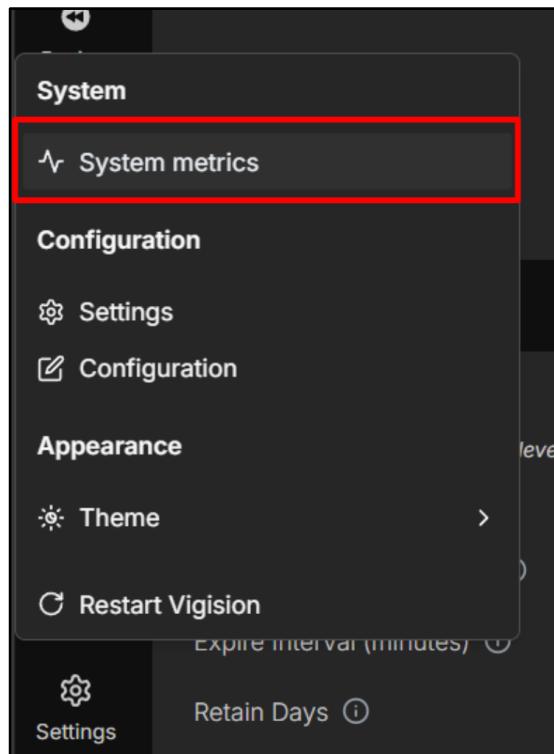


Figure 621. View system metrics

You can view **General** metrics, **Storage** metrics, and **Cameras** metrics. The **General metrics** is shown first by default.

Please have a look at **General** metrics

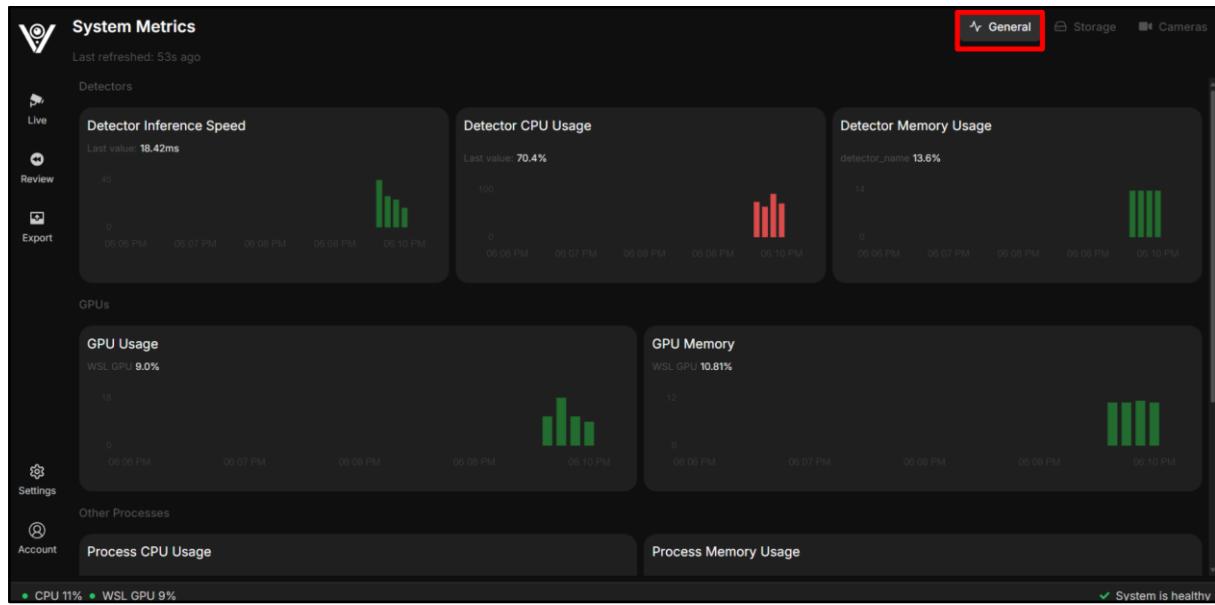


Figure 622. View system metrics

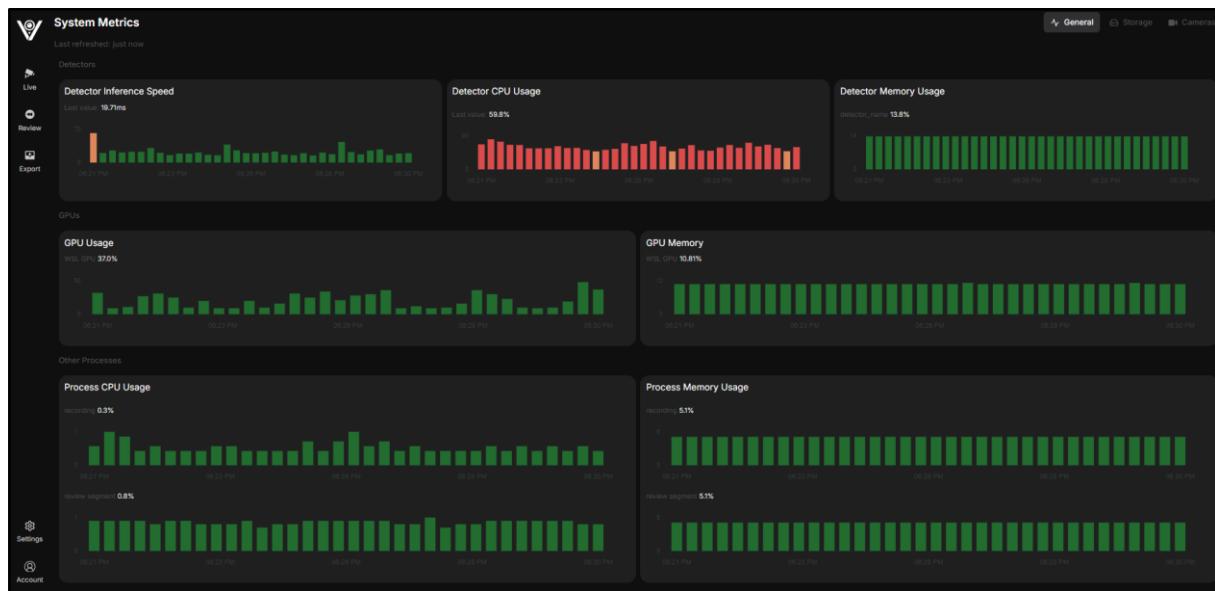


Figure 623. View system metrics

With **Detector Inference Speed**, you can view the history of inference speed (in milliseconds)

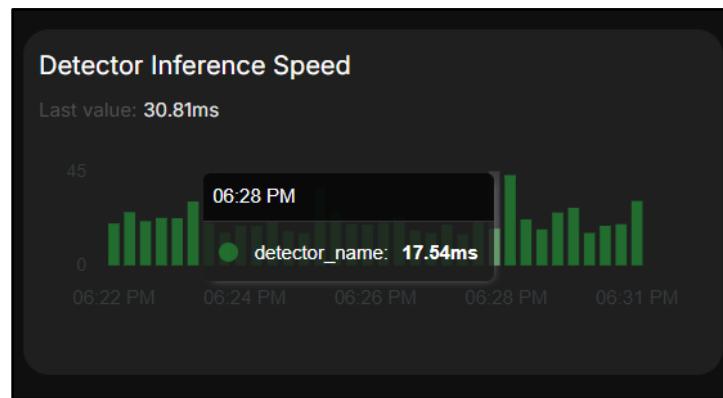


Figure 624. View system metrics

With **Detector CPU Usage**, you can view history of CPU usage (in percentage)

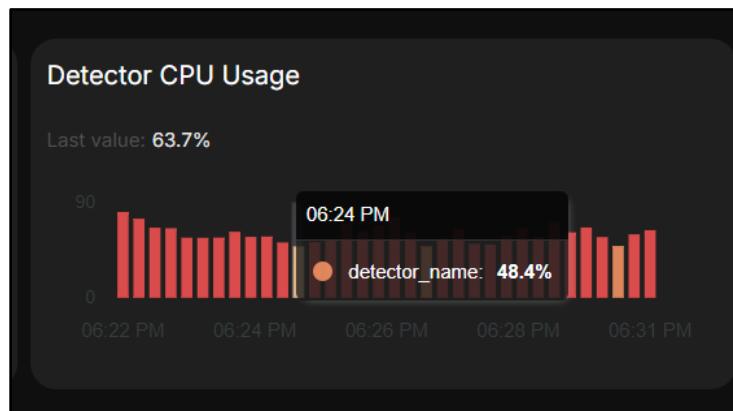


Figure 625. View system metrics

With **Detector Memory Usage**, you can view history of Memory usage (in percentage)

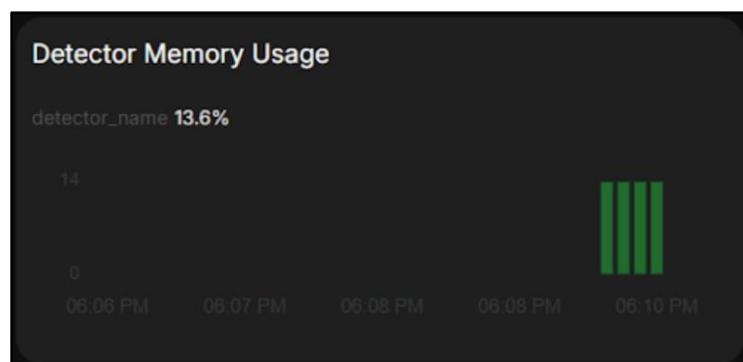


Figure 626. View system metrics

No, go to the next part - GPU groups. This provide a report of **GPU Usage** and **GPU Memory** in percentage, like below:

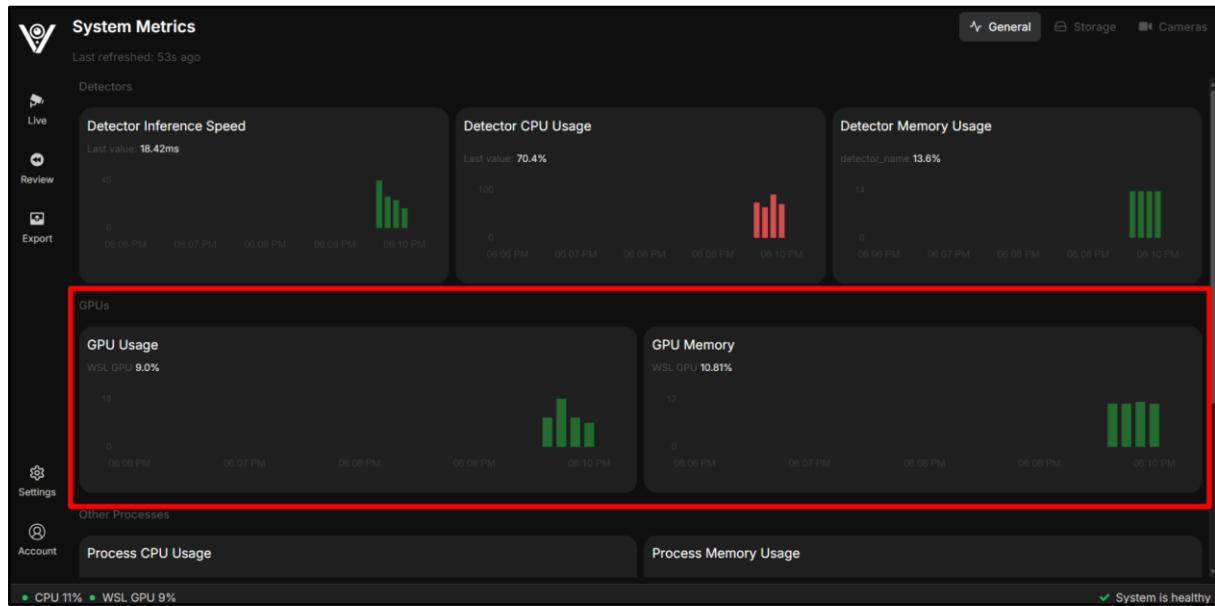


Figure 627. View system metrics

Next, go to other processes. This group give you a report of **Process CPU Usage** and **Process Memory Usage**.

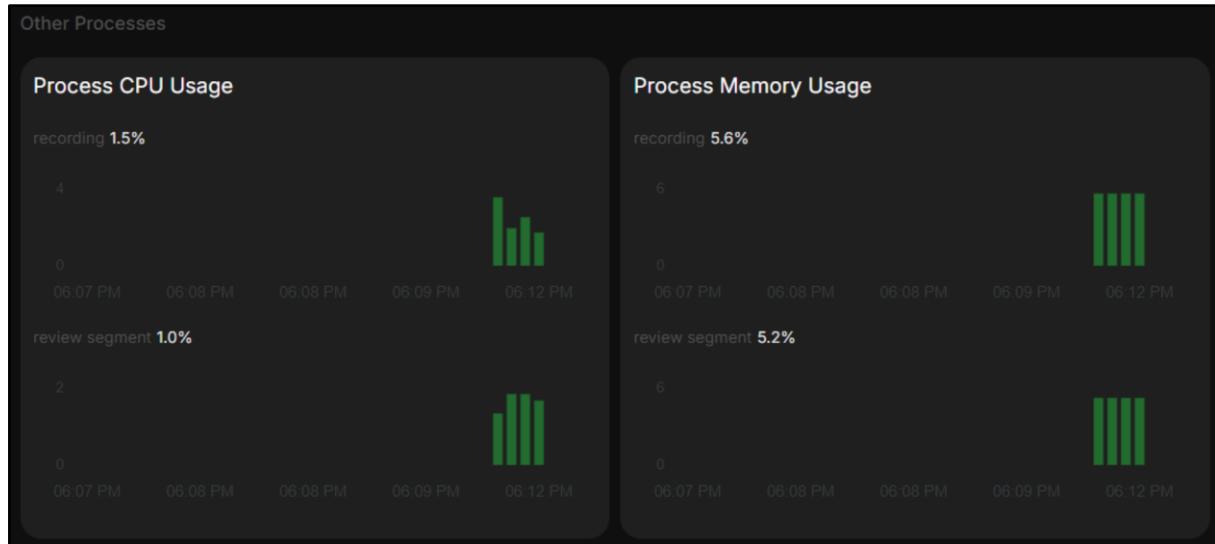


Figure 628. View system metrics

That is all for **General** metrics. Now we will have a look at **Storage metrics**.

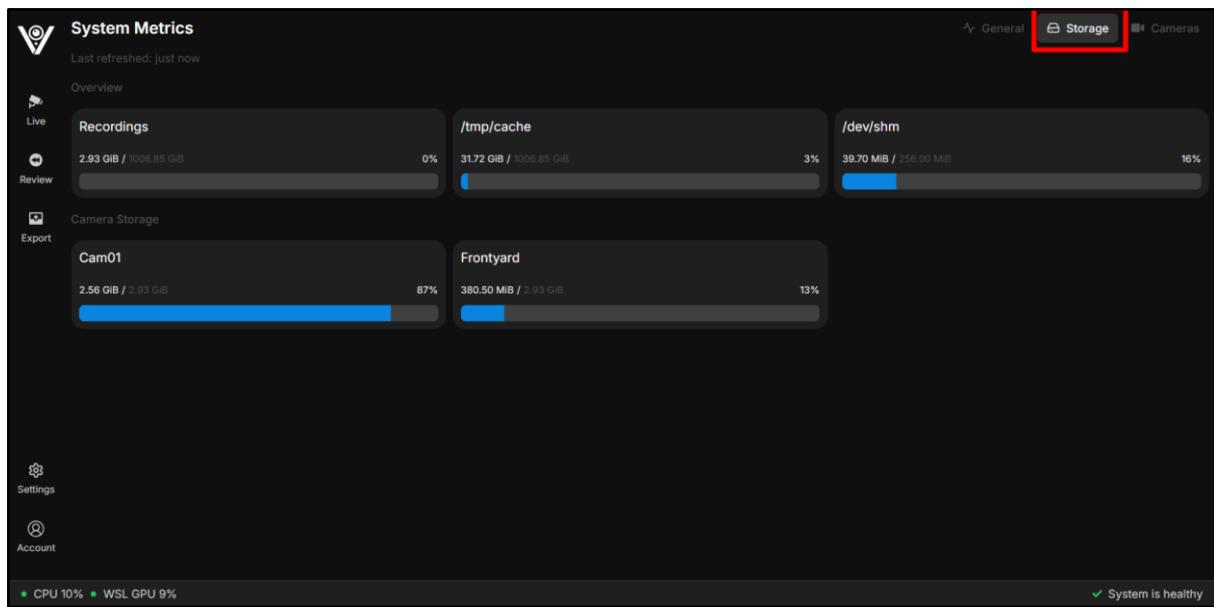


Figure 629. View system metrics

Storage metrics include size of **Recordings**, **/tmp/cache**, and **/dev/shm** in each one's total memory.

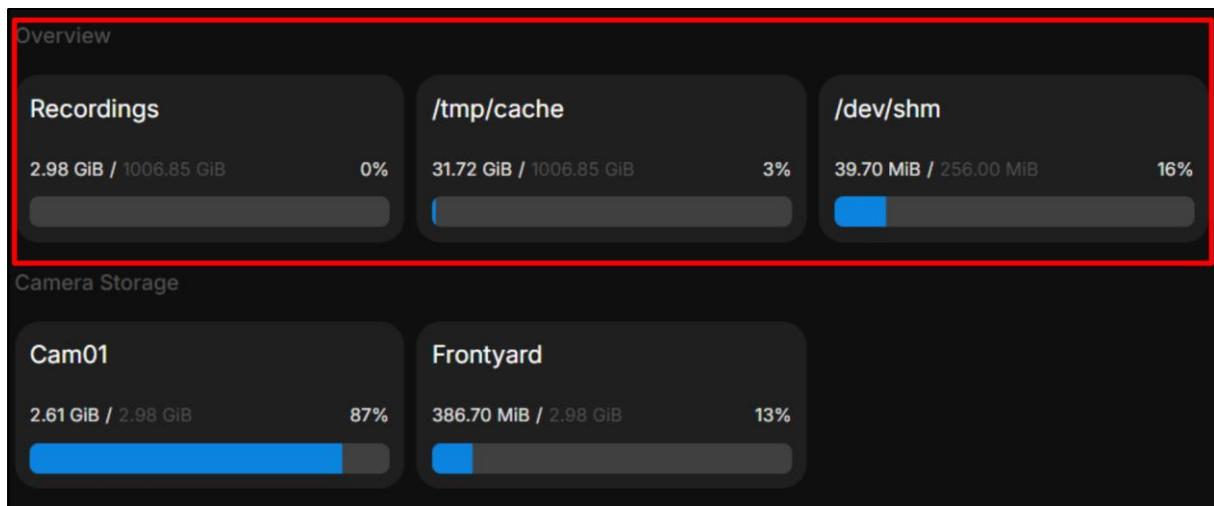


Figure 630. View system metrics

Storage metrics also include Camera Storage information - how much percent of storage each camera has used, as well as, real storage size for each camera.

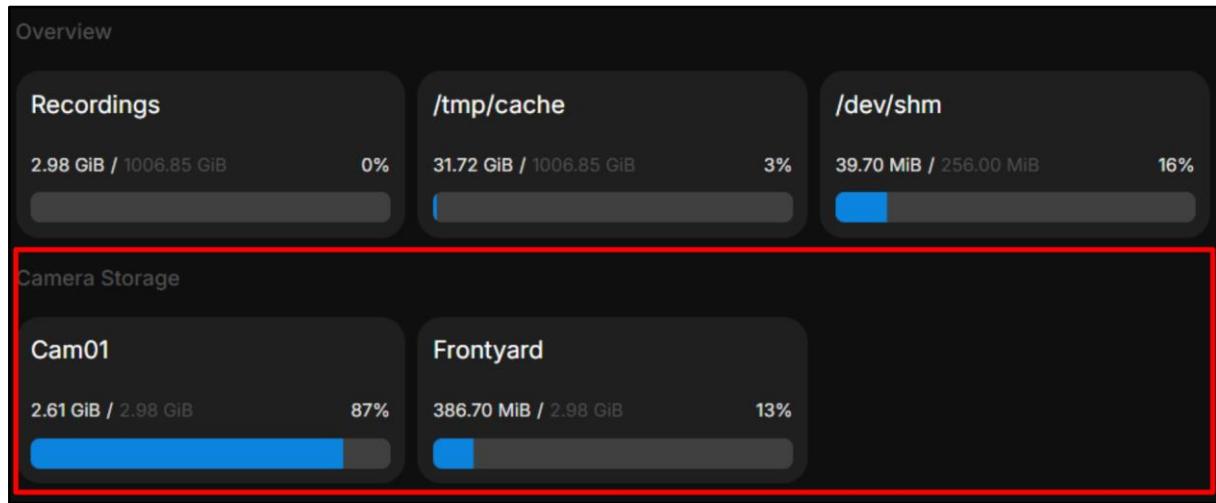


Figure 631. View system metrics

Now, we will go to **Cameras metrics**. This page includes Overview and metrics for each camera.

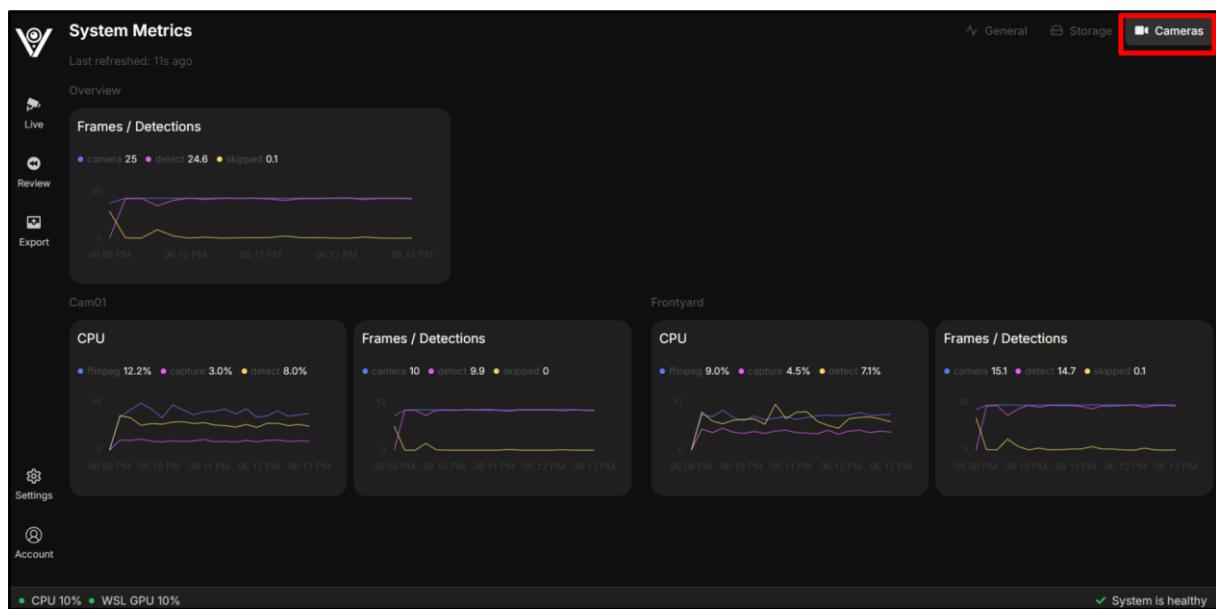


Figure 632. View system metrics

Overview is general information about **all cameras**. It provides a summary of **overall frames per second**, **overall detections per second**, and **overall skipped frames per second**, by time.

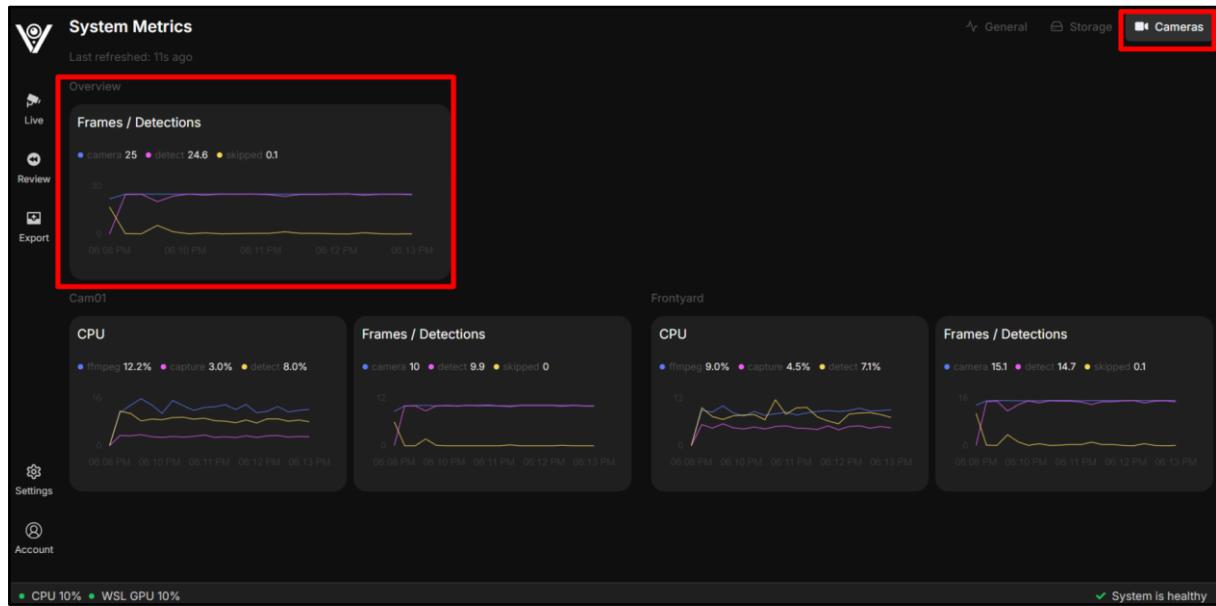


Figure 633. View system metrics

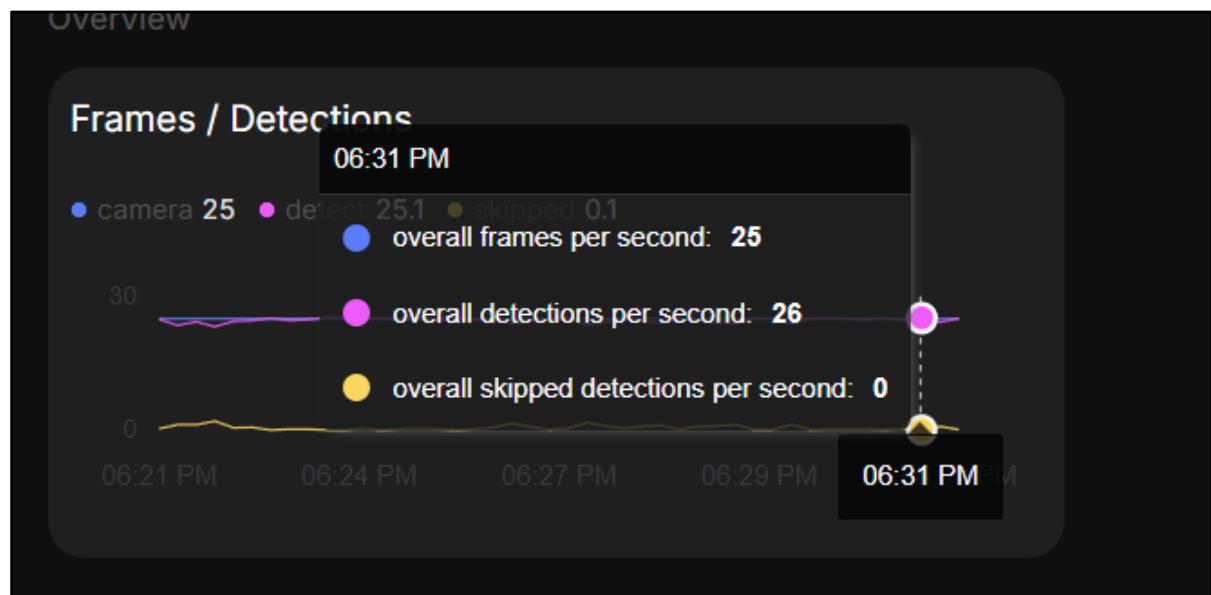


Figure 634. View system metrics

You can also view **overall frames per second**, **overall detections per second**, **overall skipped frames per second**, and **CPU usage** of each camera.

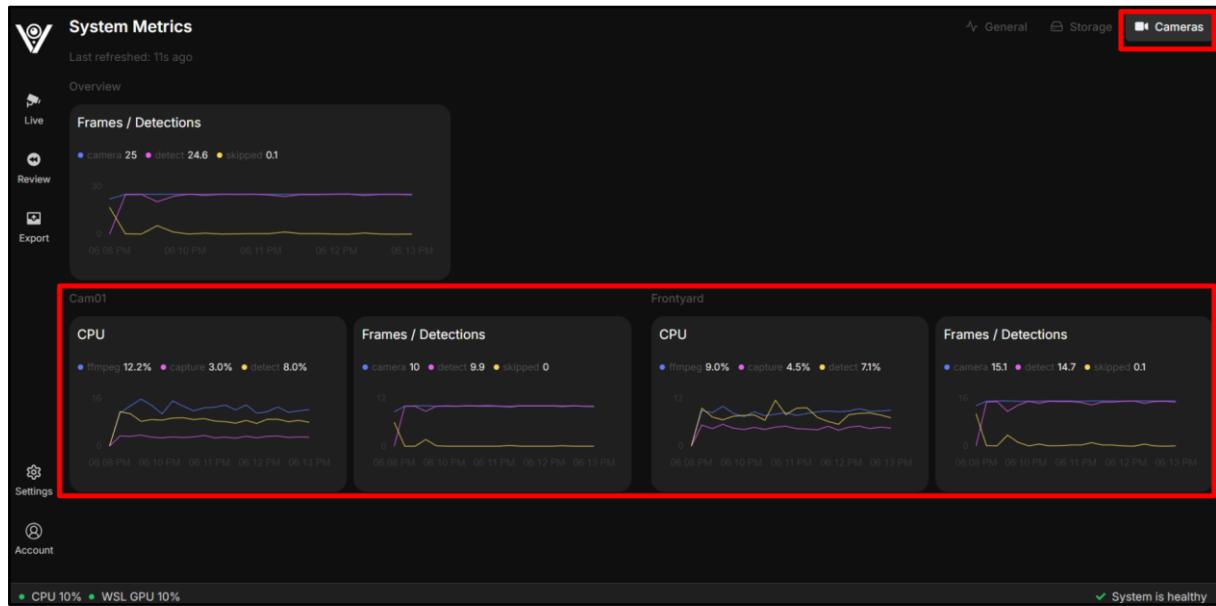


Figure 635. View system metrics

3.3.2.34 Restart application

To restart the application, there are many ways.

Way 1:

Click on button “Settings” on the sidebar.



Figure 636. Restart application

Next, click on “Restart Vigision”

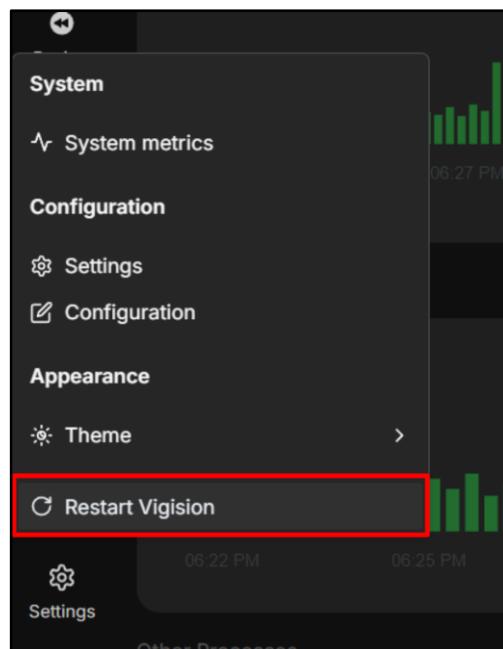


Figure 637. Restart application

Next, click on “Restart”

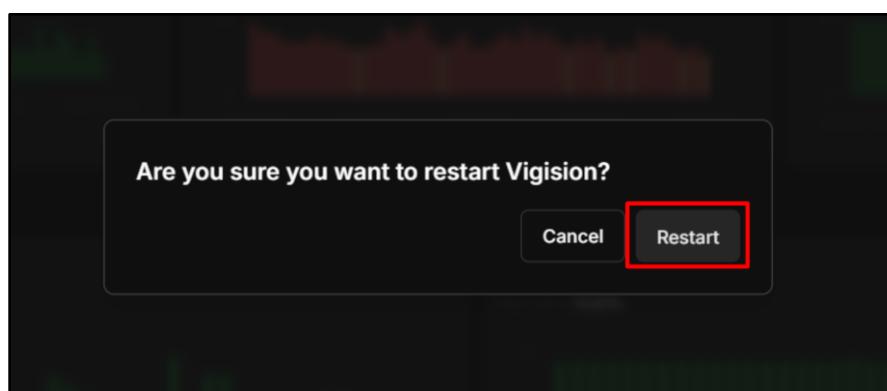


Figure 638. Restart application

You can optionally click on “Force Reload Now” or just wait for awhile.

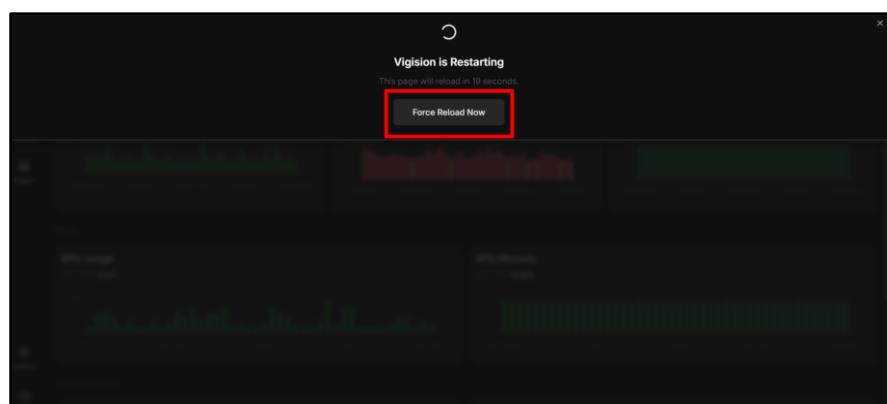


Figure 639. Restart application

Way 2:

If you have just edited camera configuration, you can click on button “**Restart Now**” on the message that pops up.

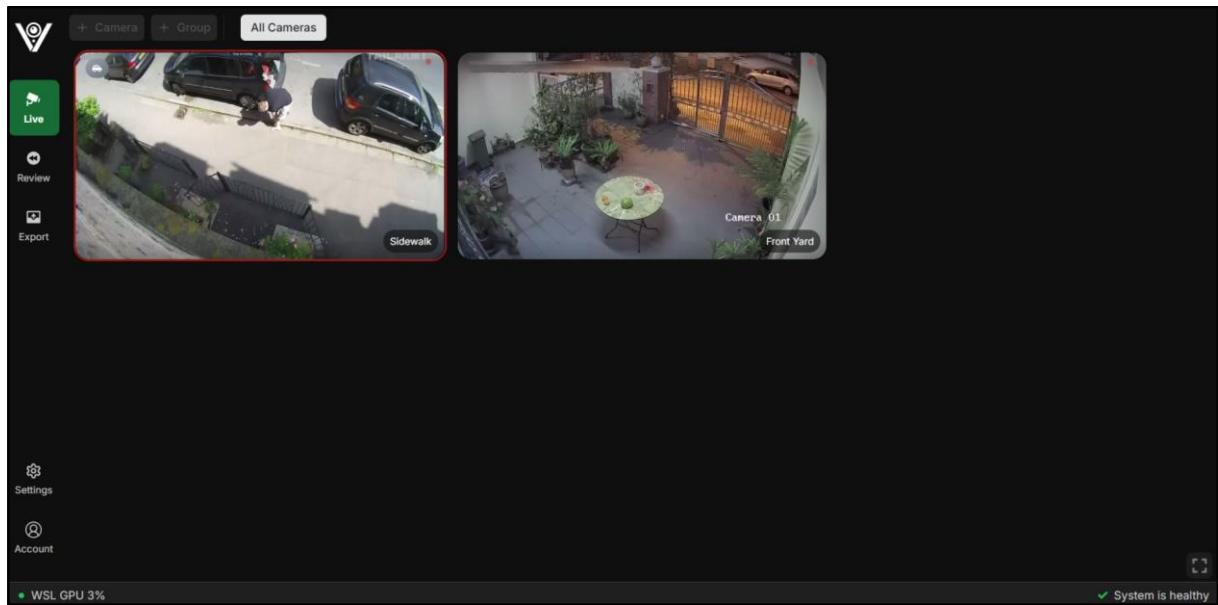


Figure 640. Restart application



Figure 641. Restart application

Then, wait for awhile for click “Force Reload Now”. This is up to your want .

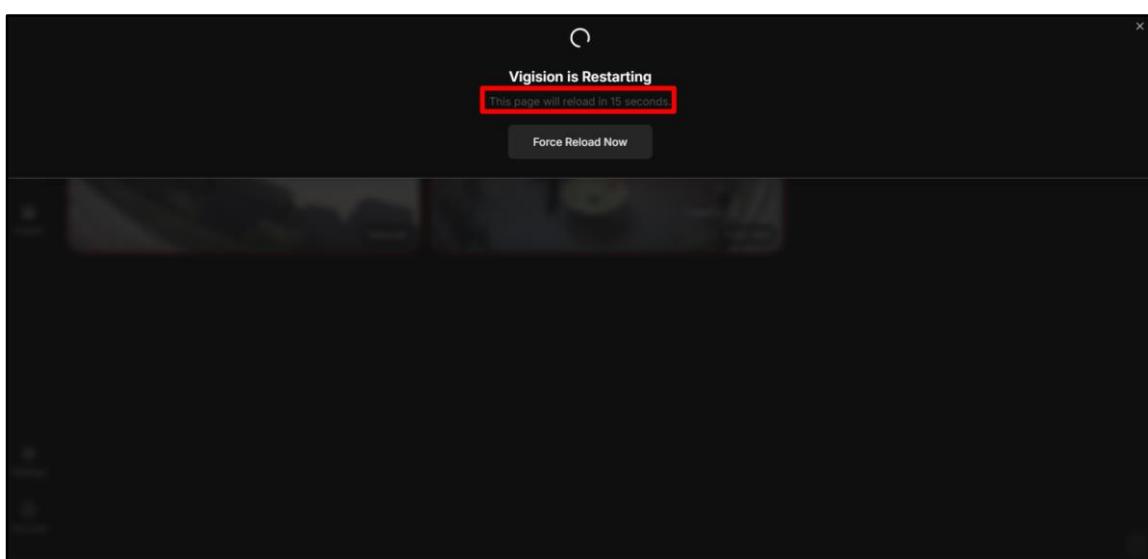


Figure 642. Restart application

After that, the system starts up again.

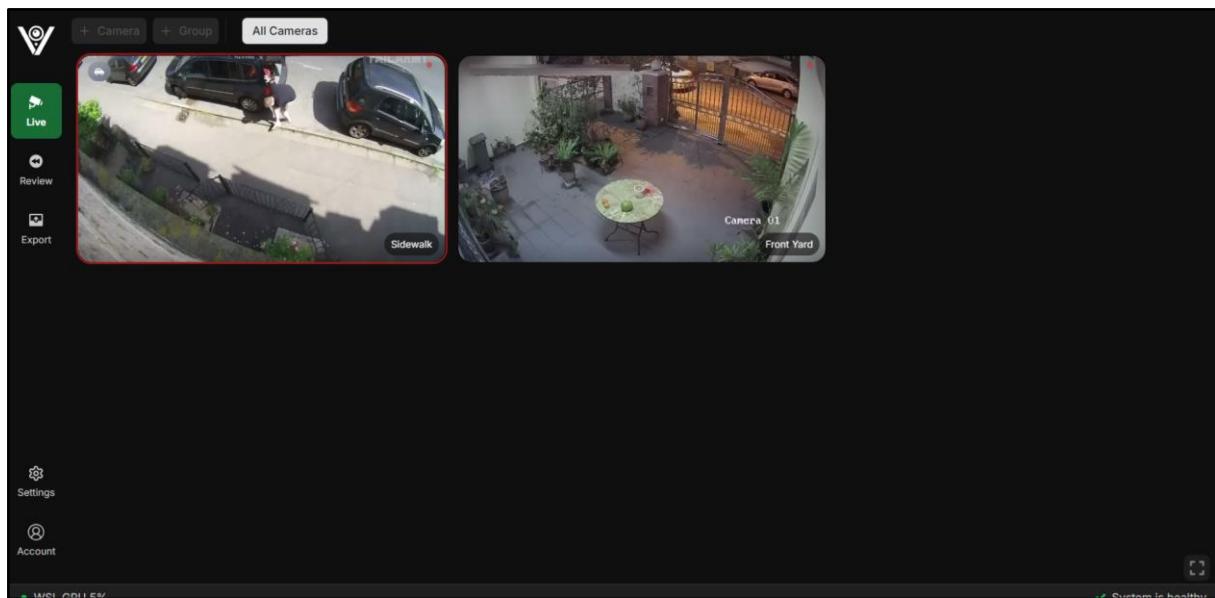


Figure 643. Restart application

3.3.2.35 Change application theme

To change the application theme, please click on the button “**Settings**” on the sidebar.

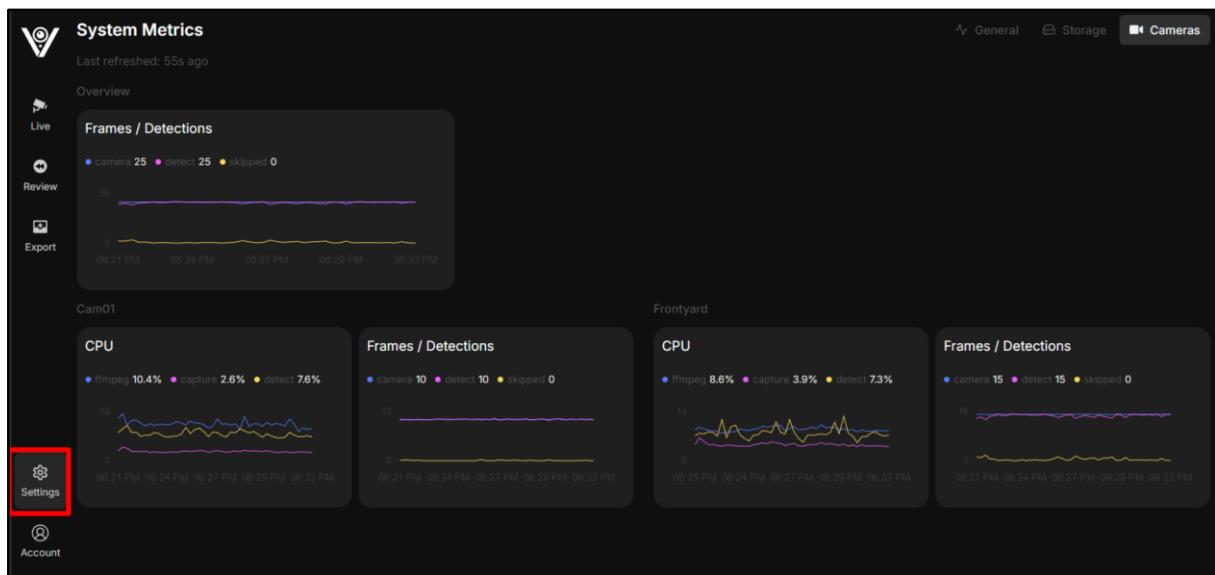


Figure 644. Screenshot of Change application theme

Next, click on the button “**Theme**” → “**Light**”

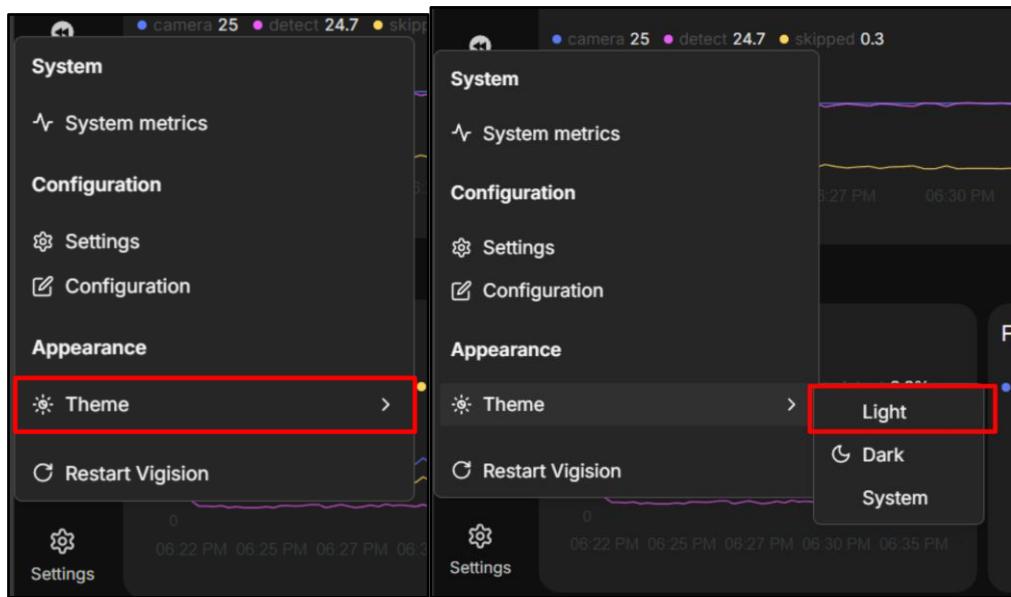


Figure 646. Screenshot of Change application theme

As the result, the application theme changed to the selected theme.

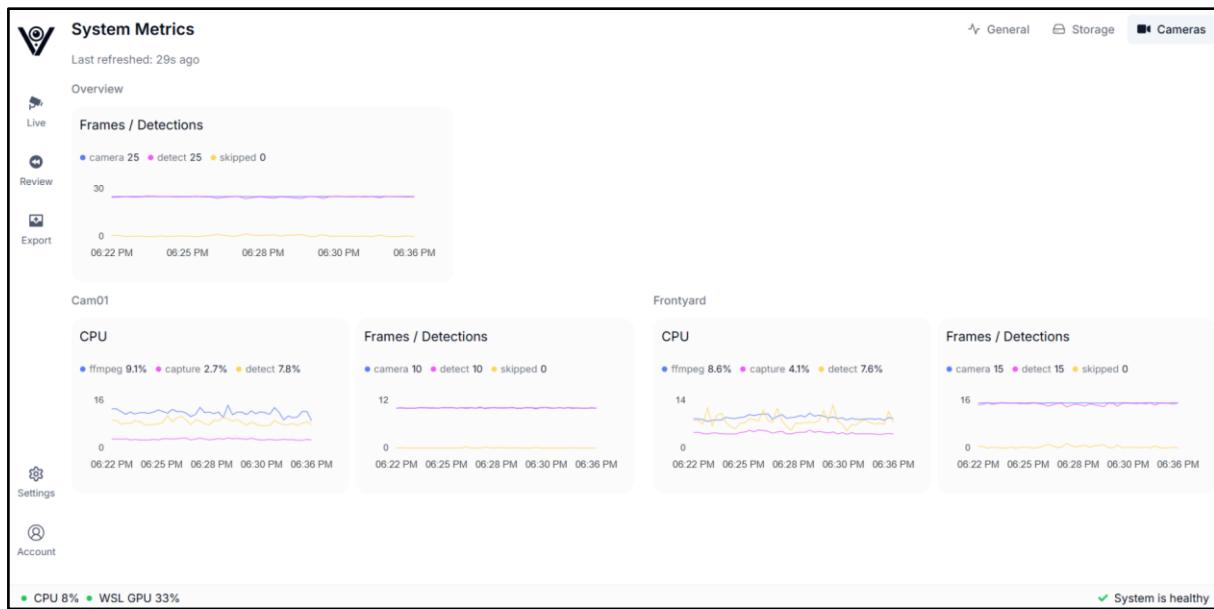


Figure 647. Screenshot of Change application theme

3.3.2.36 Forgot password

If you forgot password you can reset it by clicking on the button “Forgot password?”

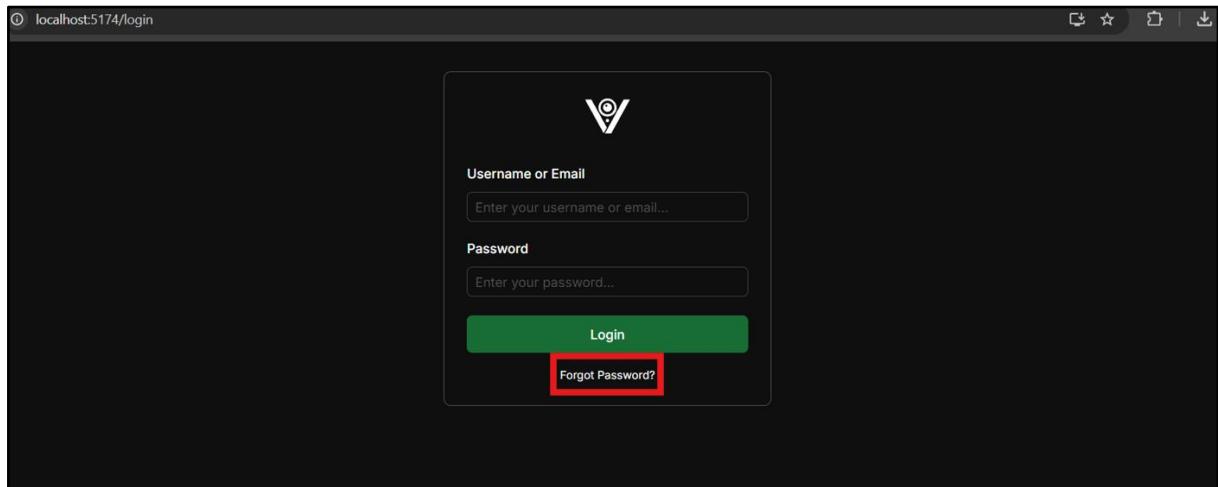


Figure 648. Button reset password

“Reset your password” pop up is shown. You could enter email address and OTP, thn, click “Verify OTP”

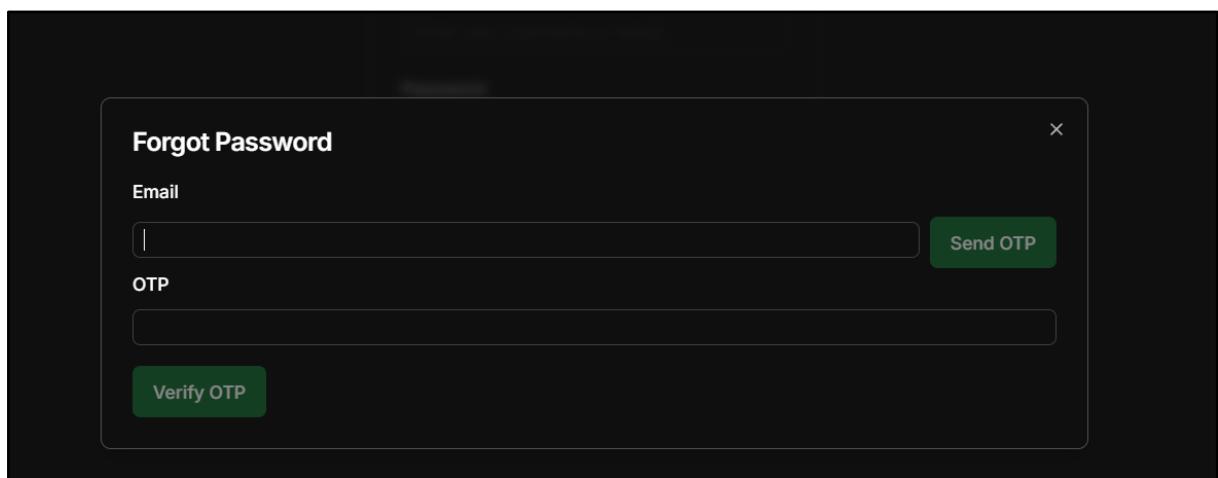


Figure 649. Screenshot of entering email and OTP

After verifying successfully, enter you new password and confirm new password:

The screenshot shows a 'Password reset' form. At the top, it says 'Set a new password'. Below that are two input fields: 'New password' and 'Confirm new password', each accompanied by a lock icon. A note below the fields states: 'At least 16 characters OR at least 8 characters including a number and a letter.' A large blue 'Set password' button is at the bottom.

Figure 650. Screenshots of entering new password

Below is an example of error case: email address and OTP are incorrect:

The screenshot shows a 'Forgot Password' form. It has fields for 'Email' (containing '03533213') and 'OTP' (containing '32'). Error messages are displayed: 'Invalid email address' next to the email field and 'OTP must be 6 characters long' next to the OTP field. A green 'Send OTP' button is to the right of the email field, and a green 'Verify OTP' button is at the bottom.

Figure 651. Screenshots of invalid email/OTP

Below is an example of error case: OTP is incorrect:

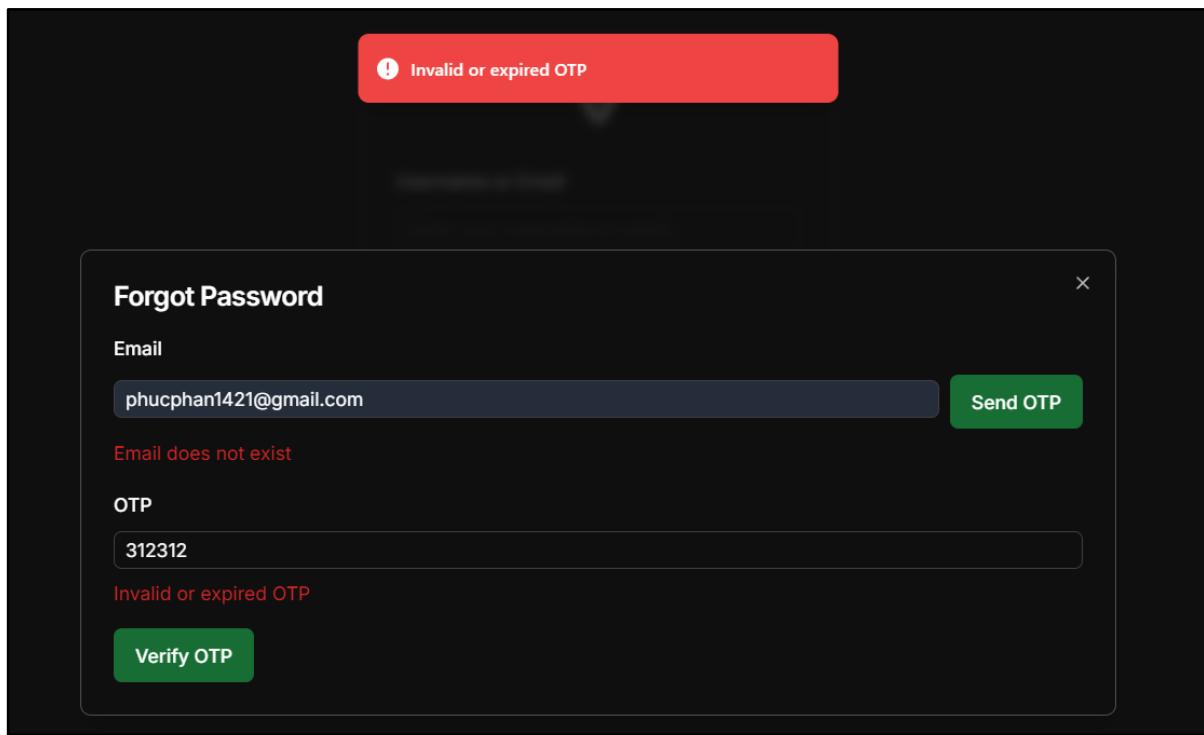


Figure 652. Another example of invalid email address/OTP

3.3.3 User guide for member

Management performs the same functions as admin but has limitations in some specific functions, described in section 3.3.1 Overview

VII. Appendix

1. Research Paper

1.1 Information

- **Title:** Advancing the Frontiers of Object Detection: A Comparative Analysis of YOLOv9, YOLOv8, and YOLOv7 on the VisDrone-DET2019 Dataset
- **Authors:** Dat Vo Minh, Phuc Phan Hong, Anh Dinh The, Thinh Nguyen Le Quang and Hoang Ngoc Tran
- **Conference:** The 39th International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC 2024)
- **Date of Acceptance:** May 2024

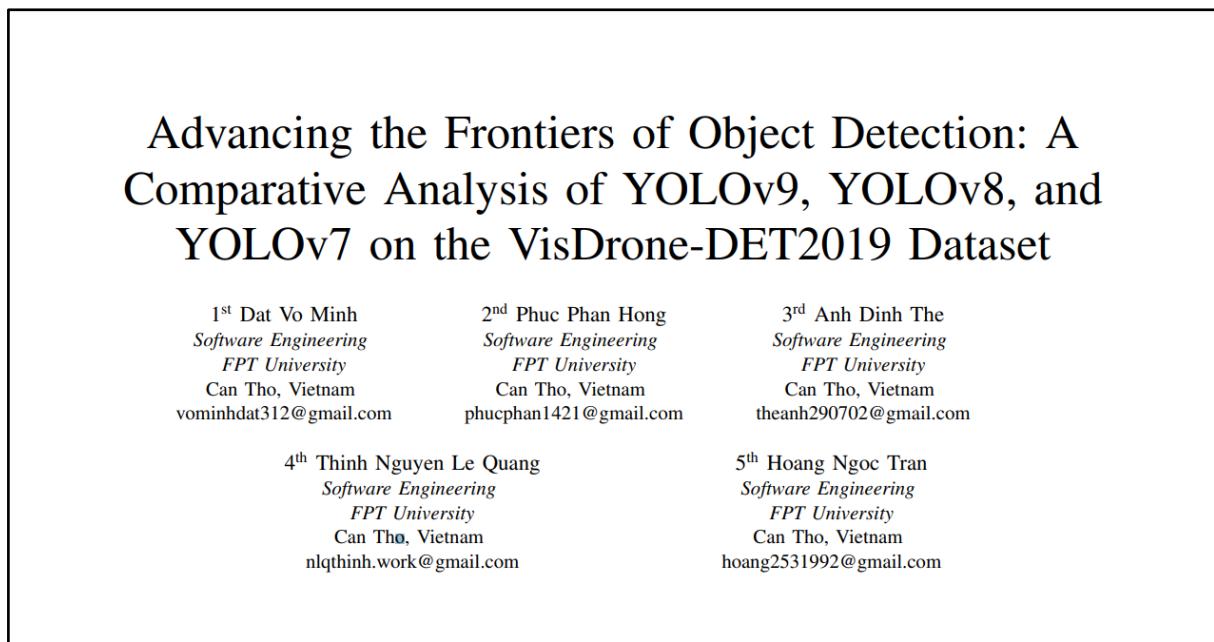


Figure 653. Paper Title with Authors

May 14, 2023

Dear Vo Minh Dat,

I am delighted to inform you that your paper, titled "Advancing the Frontiers of Object Detection: A Comparative Analysis of YOLOv9, YOLOv8, and YOLOv7 on the VisDrone-DET2019 Dataset" (Paper ID: 4633), has been accepted for presentation at the International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC) 2024.

The reviewers' scores and comments have been provided to you on May 14, 2024. These comments are intended to help you improve the quality of your paper for the final submission.

On behalf of the ITC-CSCC 2024 organizing committee, I would like to extend my sincere congratulations on your paper's acceptance. We highly appreciate your contribution to the conference and look forward to your participation in Okinawa, Japan.

Please be sure to follow the guidelines for the final manuscript submission, which will be provided to you in due course. If you have any questions or concerns, please do not hesitate to contact me or the conference secretariat.

Once again, congratulations on your achievement, and we look forward to seeing you at ITC-CSCC 2024.

Best regards,

Toshihisa Tanaka
Technical Program Committee Chair, ITC-CSCC 2024

Takayuki Nakachi
Organizing Committee Chair, ITC-CSCC 2024

Figure 654. Acceptance Letter

1.2 Abstract

This research delves into the evolving landscape of object detection algorithms, with a spotlight on the latest iterations of the You Only Look Once (YOLO) models: YOLOv9, YOLOv8, and YOLOv7. Object detection, which is critical in various real-world applications, such as surveillance and autonomous driving, requires a continuous refinement of algorithms to enhance accuracy and efficiency. Using the VisDrone-DET2019 dataset, known for its challenging aerial images, this study performs a comparative analysis focusing on performance metrics such as mean average precision (mAP50, mAP50-95), precision and recall. The findings reveal that YOLOv9 surpasses its predecessors in detection precision and mean average precision scores, setting a new benchmark in object detection capabilities. This comparison underscores the critical advancements in YOLO models, paving the way for future research and applications in complex detection environments.

2. References

- [1] "Flask," Pallets Projects, 2023. [Online]. Available: <https://flask.palletsprojects.com/en/3.0.x/#user-s-guide>
- [2] "React," Meta, 2023. [Online]. Available: <https://react.dev/>
- [3] "TypeScript," Microsoft, 2023. [Online]. Available: <https://www.typescriptlang.org/docs/>
- [4] "Axios," Matt Zabriskie, 2023. [Online]. Available: <https://axios-http.com/docs/intro>
- [5] "Tailwind CSS," Tailwind Labs, 2023. [Online]. Available: <https://tailwindcss.com/docs/>
- [6] "NGINX," NGINX, Inc., 2023. [Online]. Available: <https://docs.nginx.com/>
- [7] "Vite," Evan You and Contributors, 2023. [Online]. Available: <https://vitejs.dev/>
- [8] "ngrok," ngrok, 2023. [Online]. Available: <https://ngrok.com/docs>
- [9] "SQLite," D. Richard Hipp, 2023. [Online]. Available: <https://www.sqlite.org/docs.html>
- [10] "Peewee," Charles Leifer, 2023. [Online]. Available: <https://docs.peewee-orm.com/en/latest/>
- [11] "PyTorch," Meta AI and The PyTorch Community, 2023. [Online]. Available: <https://pytorch.org/docs/stable/index.html>
- [12] "Peewee," Charles Leifer, 2023. [Online]. Available: <https://docs.peewee-orm.com/en/latest/>
- [13] "ZeroMQ," iMatix Corporation, 2023. [Online]. Available: <https://zeromq.org/>
- [14] "OpenCV," OpenCV.org, 2023. [Online]. Available: <https://opencv.org/>
- [15] J. Redmon and A. Farhadi, "YOLO," 2015. [Online]. Available: https://pjreddie.com/darknet/yolo/#google_vignette
- [16] J. Xu et al., "AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time," arXiv preprint arXiv:2211.03375, 2022. Available: <https://arxiv.org/pdf/2211.03375.pdf>
- [17] "NumPy," NumPy Developers, 2023. [Online]. Available: <https://numpy.org/doc/stable/>
- [18] "pandas," The pandas development team, 2023. [Online]. Available: <https://pandas.pydata.org/>
- [19] "Docker," Docker, Inc., 2023. [Online]. Available: <https://docs.docker.com/manuals/>
- [20] "Python," Python Software Foundation, 2023. [Online]. Available: <https://www.python.org/doc/>
- [21] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," arXiv preprint arXiv:1611.05431, 2016. Available: <https://doi.org/10.48550/arXiv.1611.05431>.
- [22] Z. Zhu, Z.-P. Bian, J. Hou, Y. Wang, and L.-P. Chau, "When Residual Learning Meets Dense Aggregation: Rethinking the Aggregation of Deep Neural Networks," arXiv preprint arXiv:2004.08796, 2020. Available: <https://arxiv.org/abs/2004.08796>.
- [23] C. Menacho and J. Ordonez, "Fall detection based on CNN models implemented on a mobile robot," in 2020 17th International Conference on Ubiquitous Robots (UR), 2020, doi: 10.1109/UR49135.2020.9144836.

- [24] X. Cai, X. Liu, S. Li and G. Han, "Fall Detection Based on Colorization Coded MHI Combining with Convolutional Neural Network," 2019 IEEE 19th International Conference on Communication Technology (ICCT), Xi'an, China, 2019, pp. 1694-1698, doi: 10.1109/ICCT46805.2019.8947223.
- [25] C.-B. Lin, Z. Dong, W.-K. Kuan, and Y.-F. Huang, "A Framework for Fall Detection Based on OpenPose Skeleton and LSTM/GRU Models," *Appl. Sci.*, vol. 11, no. 1, Art. no. 329, 2021. Available: <https://doi.org/10.3390/app11010329>.
- [26] H. Zheng and Y. Liu, "Lightweight Fall Detection Algorithm Based on AlphaPose Optimization Model and ST-GCN," *Mathematical Problems in Engineering*, vol. 2022, Article ID 9962666, pp. 1-15, 2022. [Online]. Available: <https://doi.org/10.1155/2022/9962666>.
- [27] H. Duan, J. Wang, K. Chen, and D. Lin, "PYSKL: Towards Good Practices for Skeleton Action Recognition," in Proceedings of the 30th ACM International Conference on Multimedia, Lisboa, Portugal, 2022, pp. 7351–7354. [Online]. Available: <https://doi.org/10.1145/3503161.3548546>.
- [28] R. Egawa, A.S.M. Miah, K. Hirooka, Y. Tomioka, and J. Shin, "Dynamic Fall Detection Using Graph-Based Spatial Temporal Convolution and Attention Network," *Electronics*, vol. 12, no. 15, p. 3234, 2023. [Online]. Available: <https://doi.org/10.3390/electronics12153234>.
- [29] GlobeNewswire, Global Smart Home Security Market Report 2023-2028: Guarding Homes in the Digital Age - Smart Home Security Market Set to Double with Technological Surge. [Online]. Available: <https://www.globenewswire.com/news-release/2023/09/12/2741386/0/en/Global-Smart-Home-Security-Market-Report-2023-2028-Guarding-Homes-in-the-Digital-Age-Smart-Home-Security-Market-Set-to-Double-with-Technological-Surge.html>
- [30] JNA ORG, Home Security Systems: 6 Trends to Know in 2023. [Online]. Available: <https://jna.org/home-security-systems-trends-know-2023>
- [31] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," arXiv preprint arXiv:2305.09972, 2024.
- [32] R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ADICS58448.2024.10533619.
- [33] Zahedian-Nasab, N., Jaber, A., Shirazi, F., Kavousipor, S.: Effect of virtual reality exercises on balance and fall in elderly people with fall risk: a randomized controlled trial. *BMC Geriatrics* 21 (2021) <https://doi.org/10.1186/s12877-021-02462-w>
- [34] Yan, S., Xiong, Y., Lin, D.: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition (2018)
- [35] Keskes, O., Noumeir, R.: Vision-based fall detection using st-gcn. *IEEE Access* 9, 28224–28236 (2021) <https://doi.org/10.1109/ACCESS.2021.3058219>
- [36] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks, 2261–2269 (2017) <https://doi.org/10.1109/CVPR.2017.243>
- [37] Kwolek, B., Kepski, M.: Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine* 117, 489–501 (2014) <https://doi.org/10.1016/j.cmpb.2014.09.005>

[38] Núñez-Marcos, A., Azkune, G., Arganda-Carreras, I.: Vision-based fall detection with convolutional neural networks. *Wireless Communications and Mobile Computing* 2017, 1–16 (2017) <https://doi.org/10.1155/2017/9474806>

[39] Romaissa, B., Oussalah, M., Nini, B., Bounab, Y.: Vision-based fall detection using body geometry, 170–185 (2021) https://doi.org/10.1007/978-3-030-68799-1_13

[40] Abdo, H., Amin, K., Hamad, A.: Human fall detection using spatial temporal graph convolutional networks. *IJCI. International Journal of Computers and Information* 10, 80–98 (2023) <https://doi.org/10.21608/ijci.2023.204529.1105>