PlatePatrol

Authors: Christine Li, Vicky Liu, Andy Zhao
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—Traditional license plate recognition systems rely on fixed cameras, limiting real-time tracking for law enforcement. PlatePatrol bridges this gap by transforming everyday dash cams into a crowdsourced automatic license plate recognition (ALPR) network, enabling broader coverage at faster speed. Each dash cam processes ALPR locally and transmits detections to a secure cloud server, where license plates are matched against law enforcement watchlists. In road tests, PlatePatrol identified 468% more license plates than a human driver and achieved 8.06s capture-to-alert latency. Designed for speed, scale, privacy, and security, PlatePatrol enhances public safety with community-driven intelligence.**

***Index Terms*—Automatic License Plate Recognition, Dash Cam, Distributed System, Law Enforcement, Public Safety**

1 INTRODUCTION

Automatic License Plate Recognition (ALPR) is widely used in law enforcement to track vehicles involved in criminal activities such as stolen cars, hit-and-run incidents, and AMBER alerts [1]. However, traditional ALPR systems rely on fixed-location cameras—such as those installed on police vehicles, toll booths, and traffic lights—which offer limited spatial coverage and often introduce delays in incident reporting [2]. This lack of real-time visibility hampers law enforcement’s ability to respond swiftly in high-stakes situations [3].

To address these limitations, we developed PlatePatrol, a crowdsourced ALPR system designed for both everyday drivers and law enforcement agencies. PlatePatrol enables real-time license plate detection through a distributed network of mobile dash cams. Each dash cam processes frames locally using on-device machine learning models and securely transmits recognized license plate data to a centralized cloud server. The backend compares the detected plate against law enforcement watchlists and, if a match is found, immediately notifies the appropriate personnel through API-driven webhooks.

Our system emphasizes four core design principles: end-to-end integration, near real-time processing, distributed cloud infrastructure, and strong privacy and security safeguards. PlatePatrol dash cams operate with dual functionality—recording standard video while simultaneously running ALPR models on-device—requiring no manual intervention from drivers. By performing inference on the edge, the system minimizes latency and reduces dependence on continuous server communication. Matched plate data is routed through a scalable, serverless cloud backend built on AWS Lambda and DynamoDB, with RESTful APIs enabling integration with third-party services. Throughout the pipeline, user privacy is preserved with an instant opt-in/opt-out mechanism, API key-based authentication, and data encryption using AWS Key Management Service (KMS) and PrivateLink.

With its scalable architecture and emphasis on low-latency detection, PlatePatrol fills a critical gap in ALPR coverage and provides actionable, real-time insights that support public safety and accelerate law enforcement response.

2 USE-CASE REQUIREMENTS

2.1 Dash Cam

To ensure compatibility with typical vehicle power sources [4], the dash cam must operate on a 12V DC input. In addition, the device must weigh less than 1.5lbs. A lightweight design minimizes potential injury or damage during sudden braking or collisions. The dash cam is also required to support driving footage recording and store it on an SD card. Finally, the dash cam must provide a general hands-free experience to avoid distracting the driver during operation. Specifically, the initial setup must be completed in less than 10 minutes per device over its lifetime. Once initialized, the system should not require any additional driver interaction.

2.2 ALPR

For the ALPR system to be effective, it must achieve a minimum end-to-end accuracy of 80% in recognizing U.S. license plate numbers. This metric is calculated by dividing the number of correctly identified vehicles by the total number of vehicles with driver-legible license plates observed. To ensure reliability, the combined false positive and false negative rates must remain below 20%. The false positive rate refers to the percentage of vehicles that are incorrectly matched to a license plate due to misidentification or false detection, while the false negative rate refers to the percentage of vehicles for which the system fails to produce a prediction.

2.3 End-To-End

The end-to-end system must ensure privacy by allowing drivers to opt in or out of the ALPR system within 1s. The critical path latency—from license plate capture to law enforcement notification over the network—must not exceed 1.1s to ensure timely alerts and a faster response to incidents. The system must capture at least 2 frames of each vehicle with a legible plate, minimizing ALPR errors from motion blur or partial visibility. Lastly, ALPR data access must be restricted to authorized law enforcement personnel, complying with PA law [5] to protect sensitive information and prevent misuse.

3 ARCHITECTURE

PlatePatrol consists of a distributed edge-cloud architecture comprising two main components: dash cams deployed in vehicles and a centralized server backend. These components support two primary user roles: drivers, who operate the dash cams passively during everyday travel, and administrators from watchlist and tip line services, who manage plates of interest and receive notifications.

Figure 1 presents a high-level overview of the system and the interactions between its components.

3.1 Dash Cam

Each dash cam captures license plate images along with associated metadata such as GPS coordinates, date, and time. The device includes a single-board computer to process captured frames, a camera module for real-time recording, a hardware switch to control ALPR activation, a GPS module for geolocation tagging, local storage for buffering video and detection events, and a network module to transmit detection results to the central server via HTTP.

3.2 Central Server

The central server coordinates communication across the system and provides RESTful APIs for both dash cam devices and external service integration. It receives inbound watchlist query requests from dash cams, manages plate subscriptions from third-party systems, and issues alerts when matches are detected. The server also maintains persistent storage for active watchlists and logged match records. Through a secure and scalable interface, external organizations can register for alerts, receive real-time notifications, and integrate PlatePatrol seamlessly into their existing workflows.

4 DESIGN REQUIREMENTS

4.1 Dash Cam

The dash cam should operate on a 12V DC power source through a car cigarette lighter. To ensure stable power despite fluctuations, our design incorporates a voltage regulator that converts 12V to a steady 5V supply (up to 5A) for the single-board computer and its peripherals (camera, GPS, and cellular modules). The total unit must weigh less than 1.5lbs, with approximately 1lb allocated for the single-board computer, peripherals, and batteries, and a 0.5lb margin for additional circuitry. The dash cam continuously records in 1-minute segments to support SD card compatibility. This 1-minute duration simplifies data management by allowing the system to periodically delete older clips and facilitate easier retrieval of specific footage by drivers if needed. Finally, to maintain a hands-free experience, the system must automatically begin recording within 30s of ignition, cease within 30s after engine shutdown, and be fully operational within 10 minutes of installation.

4.2 ALPR

The ALPR system must accurately recognize at least 80% of driver-readable license plates on U.S. roads while maintaining false positive and false negative rates below 20%. Achieving this requires the license plate detection subsystem to maintain an mAP50 of 90% or higher. mAP50, or mean average precision at 50 percent overlap, reflects the model's ability to accurately detect objects (precision) and identify all relevant objects (recall). Additionally, the optical character recognition (OCR) subsystem must achieve at least 90% accuracy in correctly identifying entire license plates.

4.3 End-To-End

The system ensures driver privacy by allowing drivers to opt in or out of the ALPR system via a single switch operation, with the new privacy setting taking effect within 1s. The system's critical path latency must be less than 1.1s. This overall latency budget includes approximately 200ms per frame for license plate detection, 50ms per frame for OCR, 500ms for the network handshake, 300ms for network transfer, and an additional 50ms allowance for overhead. The system must capture at least 2 frames of each legible vehicle. Assuming that the relative driving speed difference on a road is less than 10mph and license plate visibility is limited to 18ft, the system must achieve an ALPR processing rate of approximately 2fps (1). Additionally, to ensure that only authorized law enforcement personnel can access the ALPR system, the design enforces strict data security measures. Data stored on dash cams is erased upon shutdown, while server-stored data is managed based on match status: nonmatching records are deleted immediately, and matching records are encrypted and retained for 21 days. Data transmission is secured through a dedicated network architecture, and system access is controlled via authentication and audit logs.

$$2\,\text{frames} \div (18\text{ft} \div 10\text{mph}) \approx 2\text{fps} \tag{1}$$

5 DESIGN TRADE STUDIES

5.1 Dash Cam

5.1.1 Single-Board Computer

We selected the RPi 5 as our final platform, as it offers the best overall trade-off between cost, performance,

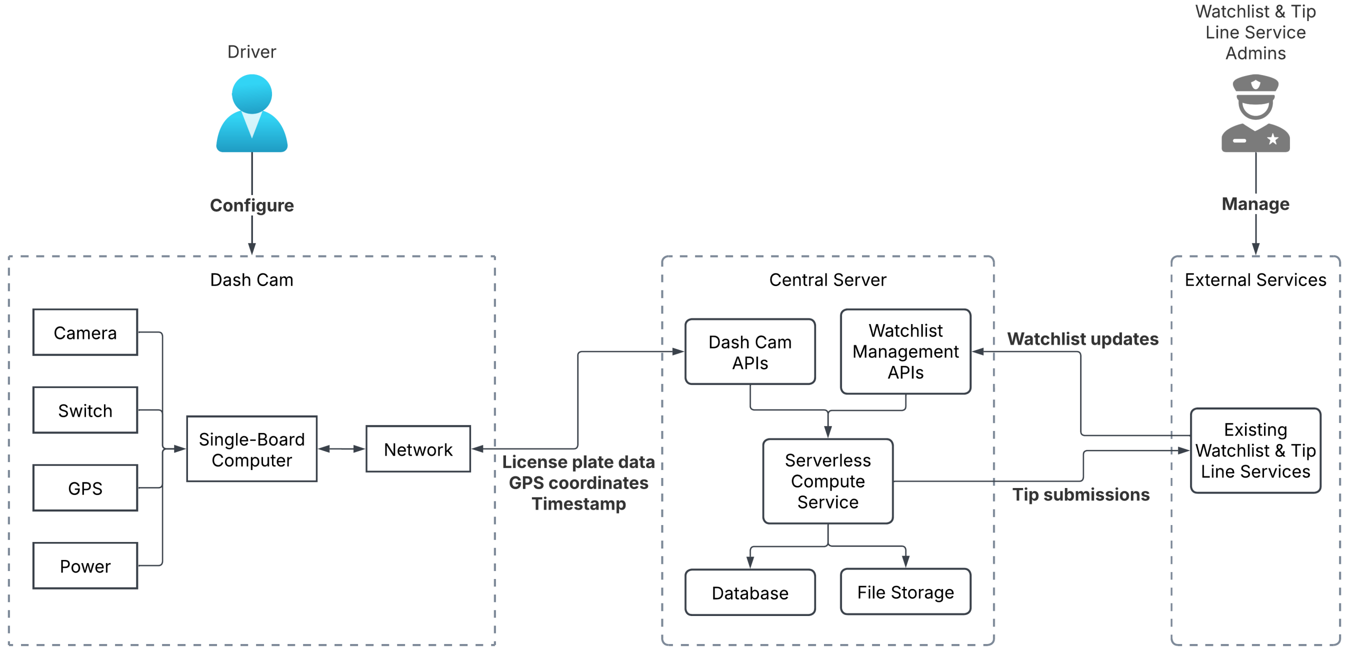


Figure 1: System Architecture Block Diagram

Table 1: Comparison of Single-Board Computers for Dash Cam

Board	Power (W)	YOLOv11n Latency (ms)	Cost (USD)	Notes
RPi 4	4.8	206	$75	Low power, slowest inference
RPi 5	11.6	168	$80	Best balance of cost and performance
Jetson Nano	10	100	$129	Faster inference, but deprecated OS
Jetson Orin Nano	15	10	$259	Fastest inference, highest cost

and integration simplicity for our edge-based ALPR system. Alternative options are summarized in Table 1.

5.1.2 Camera Module

In selecting the camera module for our dash cam, we prioritized image clarity and cost to meet our ALPR accuracy and crowdsourcing requirements. We evaluated several options, including RPi Camera Module 3, RPi High Quality Camera, RPi AI Camera, and RPi Global Shutter Camera. All candidates support 1080p recording. However, the RPi Global Shutter Camera uniquely minimizes motion blur and distortion, whereas the others rely on rolling shutter technology. The RPi Camera Module 3 is the most affordable at $25, followed by the RPi High Quality Camera and the RPi Global Shutter Camera at $50 each, with the RPi AI Camera at $70. Although the RPi AI Camera offers enhanced ALPR capabilities, our application does not require its additional processing power at increased cost. Considering image quality and affordability, the RPi Camera Module 3 is our final choice.

5.1.3 Network Communication

For network communication, we evaluated Wi-Fi, cellular, and LoRa options considering data transfer speed, coverage, and power consumption. Wi-Fi offers high data rates but is limited by range and infrastructure availability. Cellular communication provides extensive coverage and reliable connectivity suitable for real-time ALPR data transmission, despite higher power consumption. LoRa offers low-power, long-range capabilities but limited bandwidth unsuitable for image transfers and scalability concerns due to dedicated gateway requirements. Cellular communication was selected due to its optimal balance of coverage, reliability, and operational feasibility.

5.2 ALPR

5.2.1 License Plate Detection Model

For our license plate detection model, we considered YOLOv11n and YOLOv11s, since larger models exceed the computational capabilities of the RPi. YOLOv11n, with a size of 10.2MB, achieves a mAP50-95 of 60.82% and an inference latency of 168.08ms per image, whereas YOLOv11s, at 36.3MB, offers a higher mAP50-95 of 74.16% but with a significantly increased latency of 324.17ms per image. We selected YOLOv11n because it provides a lightweight solution with acceptable accuracy while ensuring low latency.

5.2.2 License Plate OCR Model

For our license plate OCR system, we experimented with three approaches: Convolutional Recurrent Neural Network (CRNN), Scene Text Recognition with a Single Visual Model (SVTR) with a Connectionist Temporal Classification (CTC) head, and SVTR augmented with both CTC and Show, Attend and Read (SAR) heads. Although the CRNN model is simple and effective for text recognition tasks, integrating it with PaddleOCR's Python API proved challenging, complicating its deployment within our pipeline. The SVTR model with the combined CTC and SAR head, while theoretically promising in capturing detailed character information, demonstrated excessive complexity and a tendency to overfit. Ultimately, we chose STR with CTC, as it provided performance comparable to CRNN while being significantly easier to integrate into our system, thereby streamlining development and ensuring reliable, low-latency ALPR operation.

5.2.3 Architectural Trade-Off

In designing our ALPR system architecture, we evaluated different configurations for performing detection, OCR, and watchlist queries between cloud and edge computing environments. Table 2 summarizes our trade-off analysis. Ultimately, we selected an approach with detection and OCR performed at the edge and watchlist queries conducted in the cloud. This configuration offers a practical balance, ensuring moderate network bandwidth usage and adequate computational performance while maintaining security.

5.3 Central Server

5.3.1 Cloud Platform

For the cloud platform, we compared Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform based on reliability, ease of integration, and flexibility. AWS excels with the widest range of services, a strong global infrastructure, and seamless support for serverless computing, storage, and databases. Although Microsoft Azure and Google Cloud Platform offer robust enterprise solutions, their serverless computing and database capabilities are less mature. We ultimately selected AWS for its comprehensive ecosystem and our team's familiarity with the platform.

5.3.2 Server Architecture

We considered two compute models for the central server: serverless computing with AWS Lambda and persistent compute using Amazon Elastic Compute Cloud (EC2) or containerized services. AWS Lambda scales automatically and eliminates idle costs, making it a cost-efficient choice for event-driven workloads like our use case. It also reduces maintenance overhead, as AWS manages provisioning and updates. One potential drawback is cold-start latency (100ms to 500ms), but our system is expected to have frequent API requests, minimizing the impact. In contrast, persistent compute, such as Amazon EC2, provides continuous availability and avoids cold-start delays, making it better suited for high-throughput applications. However, it incurs higher costs because instances remain active even during low-traffic periods and require manual scaling and infrastructure management. Since our system experiences peak hours and quiet periods, persistent compute would often be underutilized, leading to unnecessary costs. AWS Lambda was selected for its automatic scaling, lower cost, and minimal maintenance, ensuring efficient resource use while handling variable workloads.

5.3.3 Interface Design

In designing the interface for external interaction with the PlatePatrol server, we evaluated two primary options: building a dedicated user interface (UI) or adopting an API-first architecture.

A dedicated UI would offer an intuitive interface for watchlist management, match history browsing, and configuration. However, it would require significant frontend development effort, introduce potential inconsistencies across clients, and limit seamless integration with existing law enforcement databases [6] and tip line services [7]. These systems are already widely deployed, and forcing users to adopt a new interface would risk redundancy and hinder adoption.

By contrast, an API-first approach exposes all functionalities via RESTful APIs, such as watchlist registration, query, and alerting. This enables programmatic access by authorized external systems and supports automation, customization, and integration into existing software stacks. It also aligns with our serverless and scalable backend strategy.

We selected an API-first design to maximize flexibility, streamline development, and integrate cleanly with existing workflows—avoiding the need to reinvent the wheel for public safety organizations that already have backend tools in place.

5.3.4 Database Model

To meet the need for low-latency lookups and scalable operations, we transitioned to a fully NoSQL backend. We evaluated both relational (Amazon RDS) and NoSQL (Amazon DynamoDB) databases. While relational databases offer strong query flexibility, they are less suited to the high-throughput, key-based access patterns typical of watchlist matching. DynamoDB, a NoSQL key-value store, provides millisecond-level lookups, built-in TTL support, and seamless horizontal scaling.

Table 2: Comparison of Cloud vs. Edge approaches

Approach			ML Latency	Network Load	Concerns
Detection	**OCR**	**Watchlist Query**			
Cloud	Cloud	Cloud	20 ms[1]	Frequent full image (∼162 KB) upload (470 s)	Network bandwidth
Edge	Cloud	Cloud	129 ms[2]	Frequent cropped image (∼5 KB) upload (7.3 s)	Network bandwidth
Edge	Edge	Edge	257 ms	Moderate watchlist sync (2.16 s)[3], Rare cropped image (∼5 KB) upload (7.3 s)	Security & watchlist synchronization
Edge	Edge	Cloud	257 ms	Frequent watchlist query (469 ms), Rare cropped image (∼5 KB) upload (7.3 s)	—

[1] 14 ms detection on NVIDIA T4 GPU in TensorRT format, and 6 ms OCR on NVIDIA T4 GPU in Paddle format.
[2] 123 ms detection on RPi 5, and 6 ms OCR on NVIDIA T4 GPU in Paddle format.
[3] Assuming 500 entries based on the size of the Digitpol Stolen Car Database.

Although AMBER Alerts are rare (fewer than 250 cases per year [3]), our system addresses a broader class of incidents—including stolen vehicles and hit-and-runs—which account for over 850,000 cases annually in the U.S [8]. In these situations, real-time alerting is critical to maximize recovery rates and reduce investigative delays.

Given our access patterns—frequent reads for watchlist queries and fast writes for match logs—we adopted a DynamoDB-only architecture. The global watchlist is indexed by plate number for constant-time lookups, while match events are logged with timestamps and GPS coordinates for auditability. This approach simplifies the system, reduces query latency, and ensures responsiveness under variable load conditions.

5.3.5 Notification Method

To deliver real-time alerts to external services, we compared polling, push notifications, and webhooks. Polling requires external systems to repeatedly query our server, increasing load and latency. Push notifications via services like Amazon SNS offer real-time delivery but are limited to specific channels (e.g., SMS, email) and do not integrate well with backend systems.

We selected webhooks as our notification mechanism. Webhooks allow external services to register callback URLs that PlatePatrol invokes when a match occurs. This approach enables real-time, backend-to-backend integration with minimal latency and overhead. Webhooks are also stateless and scalable, fitting well with our serverless design. Compared to WebSockets, which require persistent connections and session management, webhooks are more reliable for loosely connected clients like law enforcement platforms.

5.3.6 Image Upload Strategy

The dash cam captures license plate images that must be transmitted reliably over constrained cellular networks. We considered two upload methods: direct upload to Amazon S3 using pre-signed URLs, and indirect upload through the central server via Amazon API Gateway. While direct S3 uploads reduce backend involvement, they complicate chunking and access control, and do not easily support fine-grained payload management.

To address the 2KB payload limit imposed by the dash cam's cellular module, we adopted an indirect upload strategy using API Gateway. Images are split into small chunks and transmitted as a sequence of HTTP PATCH requests. Each chunk is validated and reassembled server-side using AWS Lambda functions. This design ensures robust handling of partial uploads, supports retries, and integrates cleanly with our serverless backend while remaining compatible with mobile network constraints.

6 SYSTEM IMPLEMENTATION

6.1 Dash Cam

Figure 2 presents the detailed implementation of our dash cam with an onboard ALPR system.

6.1.1 Hardware Subsystem

The dash cam primarily derives power from the vehicle's 12V cigarette lighter socket, converted to a stable 9V supply through a dedicated voltage adapter to power our Waveshare Uninterruptible Power Supply (UPS). Our RPi 5 is then powered by this UPS for a stable 5V power supply. In addition, this UPS safeguards against abrupt voltage drops and provides a safe shutdown mechanism when the vehicle is powered off, thus protecting critical system components such as the RPi and camera module.

The RPi 5 serves as the central hardware platform, interfacing seamlessly with peripheral components. A RPi Camera Module 3, connected via MIPI CSI-2 for data transmission and I2C for control, captures 2K-resolution images at 30fps continuously for analysis. Additionally, a Notecarrier Pi hat with a Notecard Cellular module and antenna ensures wireless connectivity. Communication between the RPi and the Notecard uses an I2C interface.

For GPS functionality, we integrated the SparkFun ZED-F9R GPS-RTK module. This module delivers precise location coordinates and high-accuracy heading estimates via a USB serial connection to the RPi. This positioning data is critical for associating detected license plate matches with specific geographic locations and for supporting future functionality such as route logging and location-based filtering.

A physical switch connected to a GPIO pin allows drivers to manually enable or disable ALPR functionality, ensuring direct control over the system's operation.

6.1.2 Software Subsystem

The RPi 5 runs the Raspbian OS, configured to automatically launch the ALPR software at startup. A custom shutdown script monitors the UPS hat's voltage level, triggering a safe and graceful system shutdown when a critical voltage drop is detected, thus preventing potential file system corruption.

The ALPR software pipeline begins with a YOLOv11n model, fine-tuned on a global license plate dataset sourced from Kaggle [9], to detect and localize license plates accurately. Detected plate regions are cropped and forwarded to a PaddleOCR-based SVTR with CTC head model for text recognition, trained on 20,000 images scraped from platesmania.com (15,000 for training, 5,000 for testing). This approach compensates for the limited availability of large-scale U.S. plate datasets and accommodates the unique characteristics of U.S. plates.

For communication, the software establishes a handshake with the central server through Blues Notehub. The Blues Notecard communicates with Notehub over an encrypted private VPN channel, ensuring that data is not exposed on the public internet. Notehub, in turn, securely communicates with the central server via AWS PrivateLink. The board sends the detected license plate text to the server, which performs a database query against the watchlist to find a match. When a match is found, the board is notified and compiles a data packet containing the cropped plate image, timestamp, and GPS coordinates, and transmits this packet via HTTP over the Notecard's cellular connection.

When the ALPR system is disabled via the physical switch, the software halts the detection and OCR processes while continuing video recording, ensuring that no plate data is transmitted and thereby preserving driver privacy.

6.2 Central Server

The PlatePatrol central server is built on a fully serverless AWS stack, designed to deliver real-time performance, scalable throughput, and robust data security. The backend is organized into three functional layers: the Watchlist Subscription Layer, the Watchlist Query Layer, and the Image Upload and Notification Layer. Figure 3 illustrates the key components and data flows across these layers.

6.2.1 Watchlist Subscription Layer

External administrators can subscribe to specific license plates by registering webhooks using an API key. Each API key is uniquely assigned to an authorized organization, enabling usage tracking and strict access control. When a request is received—such as `POST /plates/{plate_number}/webhooks` to subscribe to notifications for a specific plate, or `DELETE /plates/{plate_number}/webhooks` to remove a subscription—the Watchlist Lambda function authenticates the API key, updates the global watchlist stored in Amazon DynamoDB, and logs the action in an access log table for auditability. This API-first design supports dynamic, fine-grained subscriptions and enables secure, real-time alert delivery through webhooks.

6.2.2 Watchlist Query Layer

When a dash cam detects a license plate, it issues a watchlist query to the central server by calling `GET /detections/{plate_number}`. This request is routed through Amazon API Gateway and handled by the Detection Lambda, which performs a key-value lookup in the global watchlist stored in Amazon DynamoDB. If a match is found, a new match record is created, and a unique match ID is returned to the dash cam. This match ID is then used to tag the associated image upload and streamline downstream processing. By leveraging DynamoDB's low-latency access patterns, this layer ensures that watchlist checks are completed in near real-time, supporting responsive alert delivery and scalable field deployment.

6.2.3 Image Upload and Notification Layer

Due to the 2 KB payload size limitation of the Blues Notecard Cellular module, dash cams upload license plate images using a custom chunked upload protocol. When a match is found, the dash cam receives a server-generated `match_id` in response to the watchlist query and initiates the upload by splitting the captured image into chunks (typically <2 KB each). Each chunk is sent as a `POST` request to `/uploads/{match_id}`, routed through Amazon API Gateway to the Chunk Upload Processing Lambda.

Each request includes metadata—`match_id`, `chunk_id`, `total_chunks`, and base64-encoded image data. The processing Lambda stores each chunk in a temporary Amazon S3 path under `uploads/{match_id}/chunk_{chunk_id}` and updates an Upload Status record in DynamoDB to track completeness. The system adheres to an "at least once" delivery model: dash cams automatically retry failed uploads, and the server-side logic safely handles out-of-order or duplicated chunks by deduplicating based on `chunk_id`.

Once all chunks have been received, a separate Assembly Lambda is triggered. This function retrieves the stored chunks from S3, reassembles them in order, and saves the final image to a permanent S3 location at `images/{match_id}.jpg`. It then logs the complete match event—including the recognized plate number, timestamp, GPS coordinates, and image reference—in the Match Log Table and deletes the temporary chunks.

To complete the workflow, the Assembly Lambda sends the full match payload to all registered webhooks associated with the matched plate. These outbound alerts include the plate number, location, timestamp, and a link to the reassembled image, enabling real-time responses by external systems.

Security is enforced across this layer via AWS KMS for encryption at rest, AWS PrivateLink for secure data transit, and API key-based access control at all endpoints. Operational monitoring is provided by Amazon CloudWatch, and the backend is validated through automated Jest-based integration tests covering webhook registration, upload logic, and edge cases.

7 TEST & VALIDATION

To validate that our system fulfills both its intended operational use cases and the detailed design requirements, we conducted a series of lab simulations and followed by real-world road tests with our dash cam installed in an actual car.

7.1 Dash Cam

We evaluated the dash cam's power reliability through both lab and field testing. In the lab, we simulated unstable 12V inputs and confirmed that our adapter and the UPS module consistently maintained a 5V output with minimal fluctuation, protecting the RPi and its peripherals. The unit was also powered reliably in a 30-minute road test using a vehicle's cigarette lighter socket.

For usability, the dash cam weighed only 0.586 lbs and took about 5 minutes to install by a single person. After installation, it began recording within 41 seconds of ignition and safely shut down just 332ms after power-off. Startup and shutdown sequences executed successfully across five simulated and five in-vehicle power cycles, with no file corruption.

7.2 ALPR

The ALPR testing phase included both simulated and real-world evaluations. In the simulated testing, the YOLOv11n model was benchmarked using a curated set of 386 real-world images sourced from the Kaggle-based dataset [9], achieving 90.4% mAP50. Concurrently, the PaddleOCRv3rec component was evaluated on a scraped dataset from platesmania.com, which included 4,000 cropped synthetic U.S. license plate images and 1,000 cropped real-world images, achieving 93.2% recognition accuracy.

For end-to-end simulation, We confirmed that detection and OCR on the dash cam completed in just 257ms, achieving a 3.9 FPS ALPR processing rate. For 5,000 full-frame, real-world images from platesmania.com were passed through the system. The YOLOv11n model detected and cropped license plate regions from these images, and the resulting crops were processed by the PaddleOCRv3rec model. The pipeline achieved 79% accuracy, with 17% false positive rate and 4% false negative rate. Although this fell 1% short of our 80% benchmark, we attributed the gap to challenging image conditions, including skewed angles and inconsistent text formatting. Given its strong performance in field testing, we considered this level of accuracy acceptable for deployment.

7.3 Security and Privacy

We validated privacy and security controls through structured functional testing. The physical opt-in/opt-out switch was toggled ten times, and in each case, the system updated its inference state within 100ms. We confirmed that all locally stored data was erased upon shutdown, ensuring no residual information remained on the dash cam. On the server side, non-match records were immediately discarded, while matched records were encrypted and retained for 21 days using AWS KMS.

To evaluate system security, we simulated various unauthorized access attempts. The server consistently rejected invalid or missing API keys at the API Gateway, blocked unauthorized database queries, and denied improper access to Amazon S3 resources. Additionally, all watchlist modifications made via the API were properly authenticated and logged for auditability.

7.4 End-to-End Integration

We conducted full integration tests to validate component interoperability and performance. The system reliably captured video, performed ALPR inference, queried the watchlist, and sent alerts, all without manual intervention. Detection accuracy, hardware responsiveness, and data integrity were maintained throughout extended field operation.

The system securely transmitted plate matches and associated metadata using Notehub over VPN and AWS PrivateLink. Upon match confirmation, the cropped plate image, timestamp, and GPS coordinates were uploaded to the cloud server successfully. All timestamps and transmissions were verified during testing.

We measured a total end-to-end system latency of approximately 8s from image capture to officer notification.

[1]From the server log, the average API response time is 33,ms with a 500-entry watchlist, meaning 436,ms out of the 469,ms watchlist latency is attributed to Blues network overhead. For image uploads, the server log reports an average response time of 132,ms per 2,KB chunk. For an average 5,KB cropped image (three chunks), 6,936,ms out of the 7,332,ms total upload time is attributed to Blues network

Although this exceeds our original 1.1s capture-to-alert design requirement, we still consider the system to operate in near real time for practical ALPR scenarios.[1]

During a 30-minute road test, the driver verbally identified license plates while safely operating the vehicle, and the entire drive was recorded for later analysis. After the test, we cross-referenced the video footage with the system's ALPR detections. The driver was able to identify 25 license plates, all of which were correctly recognized and logged by the system. In addition, the system detected and recorded 117 additional license plates that the driver missed—demonstrating a 468% improvement over human observation. We also confirmed accurate, end-to-end communication between the dash cam and backend server.

These results confirm that PlatePatrol meets the key goals of high ALPR accuracy, reliable and safe dash cam operation, secure cloud integration, and strong privacy guarantees, making it viable for real-world deployment in vehicle-based surveillance and enforcement applications.

8 PROJECT MANAGEMENT

8.1 Schedule

Our schedule and responsibility is shown in Fig. 4.

8.2 Team Member Responsibilities

All team members contributed to system integration, road testing, final video production, the written report, and the presentation documentation.

- **Vicky:** ALPR algorithm development and optimization, dash cam hardware integration
- **Andy:** Dash cam hardware setup
- **Christine:** Central server implementation

8.3 Bill of Materials and Budget

We included our bill of materials in Table 3. All items listed in the table were used, excluding the RPis, which we borrowed and returned afterward.

8.4 TechSpark Usage

We conducted some circuit testing in TechSpark, where we soldered our opt-in switch to the board. No materials were taken, and all lab equipment was returned to its original position.

8.5 Risk Management

We proactively identified and mitigated key risks across hardware, software, and deployment. This section outlines how we handled technical and operational risks to ensure a stable, reliable system.

8.5.1 Dash Cam Technical Risks

- **Hardware Stability:** We followed an iterative development process to validate core hardware components, including power delivery, camera interface, and GPS integration. Early testing helped catch voltage instability and connectivity issues before full system assembly.

- **Safe Shutdown:** We incorporated a UPS module to protect the RPi from abrupt power loss. This reduced the risk of SD card corruption and prolonged device lifespan.

- **Thermal and Power Constraints:** We benchmarked RPi 4 and 5 for power draw, thermal behavior, and inference latency to ensure they could support real-time ALPR workloads within automotive environments.

8.5.2 Software and Integration Risks

- **Integration Testing:** To avoid integration failures, we developed unit and integration tests using Jest on our Node.js backend. Mocked API requests allowed us to validate system logic, input validation, and error handling before connecting to live dash cam data.

- **Inference Performance:** We tested our machine learning models (YOLOv11n and PaddleOCR) early on edge hardware to validate latency and ensure they could meet our frame rate requirements under real-world conditions.

8.5.3 Operational and Testing Risks

- **Phased Deployment:** We structured our development around key integration milestones—including ALPR inference, watchlist querying, full image upload, and chunked image upload—each followed by targeted validation. This incremental integration approach allowed us to test interfaces between components early, reduce system-level bugs, and build confidence before full deployment.

- **Safe Testing Environments:** All initial testing was conducted in lab conditions before road deployment. This protected hardware and allowed debugging without the risks of in-vehicle testing.

- **Real-World Validation:** We planned multiple road tests, each with defined test cases and fallback procedures, ensuring safe operation and reliable data collection even under variable driving conditions.

overhead. Note that image uploads are relatively infrequent, as they are only triggered when a license plate matches a target in the watchlist.

Table 3: Bill of Materials

Description	Manufacturer	Quantity	Cost @	Total
Camera Module 3	Raspberry Pi	1	$25.00	$25.00
Camera Cable	Raspberry Pi	1	$1.00	$1.00
Notecard Cellular	Blues	1	$53.00	$53.00
Notecarrier Pi Hat	Blues	1	$15.00	$15.00
LTE and GPS Antenna	Blues	1	$3.25	$3.25
Waveshare UPS HAT (B)	Waveshare	1	$33.99	$33.99
18650 Rechargeable Battery	BENKIA	1	$12.96	$12.96
USB-C to USB-C Cable	Anker	1	$9.99	$9.99
Circuit Wire to Car Cigarette Lighter Outlet	HATMINI	1	$9.99	$9.99
Cigarette Lighter Adapter	Outtag	1	$14.99	$14.99
PERMA-PROTO HAT FOR PI MINI KIT	Adafruit Industries	3	$4.95	$14.85
SWITCH SLIDE DPDT 300MA 6V	C&K	5	$0.77	$3.85
Shipping (all sources incl. DigiKey)				$46.19
Tariff (DigiKey)				$2.38
				$246.44

9 Ethical Issues

9.1 Public Benefit and Societal Impact

PlatePatrol supports public safety by helping law enforcement locate vehicles involved in crimes such as auto theft, hit-and-runs, and AMBER Alerts. By leveraging a network of crowdsourced dash cams, the system expands ALPR coverage beyond fixed infrastructure and enables near real-time alerts. This can reduce response times, assist with evidence gathering, and potentially save lives during time-critical incidents.

9.2 Privacy and System Security

Despite its benefits, PlatePatrol introduces privacy concerns if license plate detections are aggregated or misused. For example, an unauthorized party could exploit the system to monitor vehicle movements or build travel histories of individuals. Additionally, a malicious user could attempt to upload falsified detection data to trigger false alerts or mislead law enforcement. To reduce this risk, PlatePatrol enforces strict access control: only authorized third-party users with valid API keys can access the system, and only registered dash cams can upload detection data. All watchlist activity is logged for auditability.

While our system implements robust security controls, we acknowledge that no system is immune to advanced threats such as infrastructure-level breaches or insider misuse. Continued monitoring and regular security audits are essential to uphold public trust.

9.3 User Consent and Data Retention

PlatePatrol is designed to respect individual autonomy and minimize unnecessary data collection. Participation in the system is fully voluntary: drivers retain control through a physical opt-in/opt-out switch, which halts all ALPR activity and data transmission when toggled off. On the backend, data retention is minimized by design—non-matching detections are discarded immediately and never stored. Only matched license plate records are retained, and even then, they are encrypted and stored for no more than 21 days in accordance with Pennsylvania privacy law. This ensures user consent, limits exposure, and prevents the accumulation of long-term surveillance data.

10 RELATED WORK

Several related projects and products offer insights into the feasibility and impact of our solution.

10.1 Similar ECE Capstone Projects

AutoAlert [10], a Fall 2024 CMU ECE Capstone project, explored real-time image processing using a dash cam and a mobile app for lane detection, traffic light alerts, and collision warnings. Its approach to embedded computer vision and mobile integration inspired us to develop a similar system for license plate recognition and real-time law enforcement alerts. While AutoAlert focused on driver safety, PlatePatrol adapts the concept for crowdsourced ALPR and cloud-based data processing.

10.2 Commercial ALPR Solutions

Several commercial ALPR systems helped shape our approach: 1) Rekor Scout [11] applies ALPR to existing surveillance cameras, demonstrating the efficiency of centralized processing. This influenced our decision to aggregate crowdsourced dash cam data into a cloud-based system. 2) Leonardo ELSAG [12], used by law enforcement, aids in stolen vehicle recovery and missing person searches. Its reliance on fixed and mobile-mounted cameras highlighted the need for broader coverage, reinforcing our distributed dash cam model. Additionally, ELSAG's

$15,000–$25,000 per-unit cost led us to focus on affordability, exploring consumer-grade dash cams and cloud-based processing as a scalable alternative.

10.3 Cloud-Based Vehicle Tracking Systems

Platforms like Samsara Fleet Tracking [13] and Geotab [14] use GPS and cloud computing for real-time vehicle monitoring. Their scalable cloud infrastructure influenced our cloud-based approach, ensuring efficient real-time license plate matching and law enforcement alerts.

11 SUMMARY

11.1 Future Work

Several enhancements can extend PlatePatrol's capabilities and impact. On the edge, upgrading to more advanced single-board computers or integrating dedicated AI accelerators could improve inference speed and enable additional models, such as vehicle make/model classification or color recognition. From a hardware perspective, designing a custom PCB could simplify wiring, improve reliability, and reduce form factor. Likewise, a more durable and compact enclosure would enhance heat dissipation, weather resistance, and overall usability in real-world driving environments.

For cloud-side infrastructure, replacing the Blues Notehub with a more flexible cellular module could eliminate current payload constraints and streamline the image upload pipeline.

To expand real-world applicability, future iterations could support integration with national law enforcement databases and provide jurisdiction-based access control. A broader deployment would benefit from scalable provisioning workflows to register and manage new dash cams, as well as robust analytics for alert auditing, system health monitoring, and usage tracking across agencies or regions.

11.2 Lessons Learned

Throughout the development of PlatePatrol, we learned the importance of balancing technical ambition with practical design. Early in the project, our focus was heavily tilted toward individual subsystems—achieving high ALPR accuracy, designing a reliable power supply, or optimizing cloud performance. However, as we moved toward full system integration, we realized that a successful product depends equally on how these components interact in real-world conditions.

One major takeaway was the value of thinking like both engineers and designers. Engineering solutions must be robust, but they also need to be intuitive, maintainable, and user-aware. For example, designing a one-switch opt-in mechanism for ALPR addressed not just privacy policy but actual driver behavior.

Integration was another key lesson. Building subsystems in isolation led to unforeseen issues when combining them, especially when network latency, hardware timing, or asynchronous logic were involved. In response, we adopted earlier testing with mock APIs, tighter feedback loops, and incremental integration milestones. This significantly reduced the complexity of debugging and improved final system reliability.

Glossary of Acronyms

Include an alphabetized list of acronyms if you have lots of these included in your document. Otherwise define the acronyms inline.

- ALPR – Automatic License Plate Recognition
- API – Application Programming Interface
- AWS – Amazon Web Services
- DC – Direct Current
- FPS – Frames Per Second
- GPS – Global Positioning System
- HTTP – Hypertext Transfer Protocol
- KMS – Key Management Service
- LoRa - Long Range
- mAP – Mean Average Precision
- OCR – Optical Character Recognition
- PCB – Printed Circuit Board
- RDS – Relational Database Service
- RPi – Raspberry Pi
- SD – Secure Digital
- S3 – Simple Storage Service
- SQL – Structured Query Language
- TTL – Time To Live
- UPS – Uninterruptible Power Supply
- VPN – Virtual Private Network

References

[1] Coplogue. *How Automatic License Plate Recognition (ALPR) Technology Has Transformed Law Enforcement.* Jan. 2025. URL: `https://coplogue.com/2025/01/07/how-automatic-license-plate-recognition-alpr-technology-has-transformed-law-enforcement/`.

[2] C. S. Koper and C. Lum. “The Impacts of Large-Scale License Plate Reader Deployment on Criminal Investigations”. In: *Police Quarterly* 22.3 (2019), pp. 305–329. DOI: `10.1177/1098611119828039`.

[3] National Center for Missing Exploited Children. *2023 AMBER Alert Report: Analysis of AMBER Alert Activations in 2023.* Tech. rep. National Center for Missing Exploited Children, 2024. URL: `https://www.missingkids.org/content/dam/missingkids/pdfs/2023_Annual_AMBER_Alert_Report.pdf`.

[4] A. Harlin. *A Guide to Vehicle Power Outlets.* Feb. 2024. URL: `https://www.carparts.com/blog/a-guide-to-vehicle-power-outlets`.

[5] *An Act Amending Title 75 (Vehicles) of the Pennsylvania Consolidated Statutes.* 2021. URL: `https://www.legis.state.pa.us/CFDOCS/Legis/PN/Public/btCheck.cfm?txtType=PDF&sessYr=2021&sessInd=0&billBody=H&billTyp=B&billNbr=0133&pn=0099`.

[6] Digitpol. *Stolen Car Database.* `https://digitpol.com/stolen-car-database`. Accessed: 2025-05-02. 2025. URL: `https://digitpol.com/stolen-car-database`.

[7] WeTip. *Submit a Crime Tip Anonymously.* `https://www.wetip.com/submit-a-crime-tip/`. Accessed: 2025-05-02. 2025. URL: `https://www.wetip.com/submit-a-crime-tip/`.

[8] Aliza Vigderman and Maya Afilalo. *The State of Hit-and-Runs in the U.S.* AutoInsurance.com, last accessed April 30, 2025. Apr. 2025. URL: `https://www.autoinsurance.com/research/hit-and-run/`.

[9] F. Elmenshawii. *Large License Plate Detection Dataset.* 2024. URL: `https://www.kaggle.com/datasets/fareselmenshawii/large-license-plate-dataset`.

[10] A. Lenka, E. Szabo, and E. Lee. *Team A4: AutoAlert.* Carnegie Mellon ECE Capstone, Fall 2024. 2024. URL: `https://course.ece.cmu.edu/~ece500/projects/f24-teama4/`.

[11] Rekor Systems. *Rekor Scout: Vehicle Recognition Platform.* 2025. URL: `https://www.openalpr.com/software/scout`.

[12] Leonardo US Cyber and Security Solutions. *ELSAG License Plate Reader Products.* 2024. URL: `https://www.leonardocompany-us.com/lpr/products`.

[13] Samsara. *Samsara for Logistics.* 2025. URL: `https://www.samsara.com/industries/logistics`.

[14] Geotab. *Fleet Dash Cams: Enhance Safety with AI-Powered Video Solutions.* 2025. URL: `https://www.geotab.com/fleet-management-solutions/fleet-dash-cams/`.

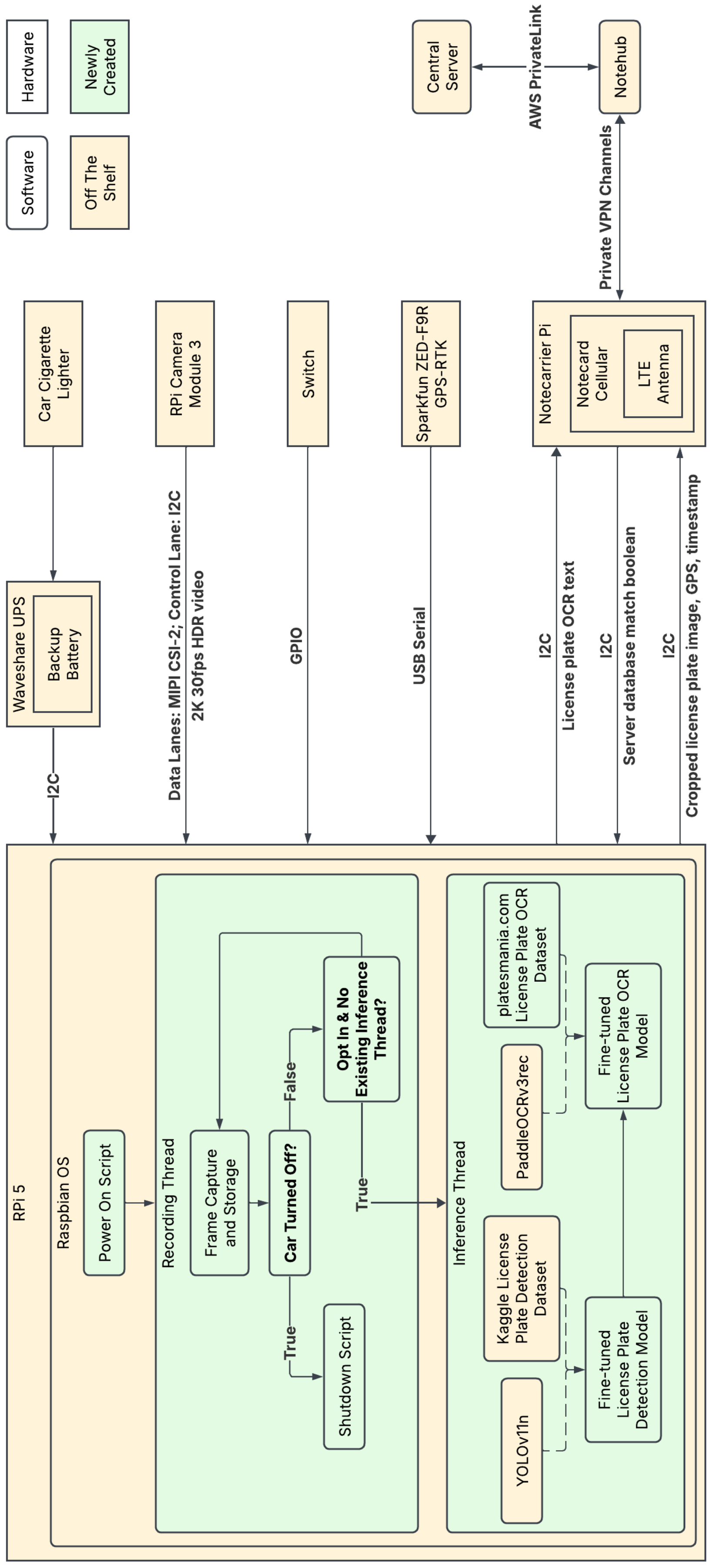


Figure 2: Dash Cam & ALPR Block Diagram

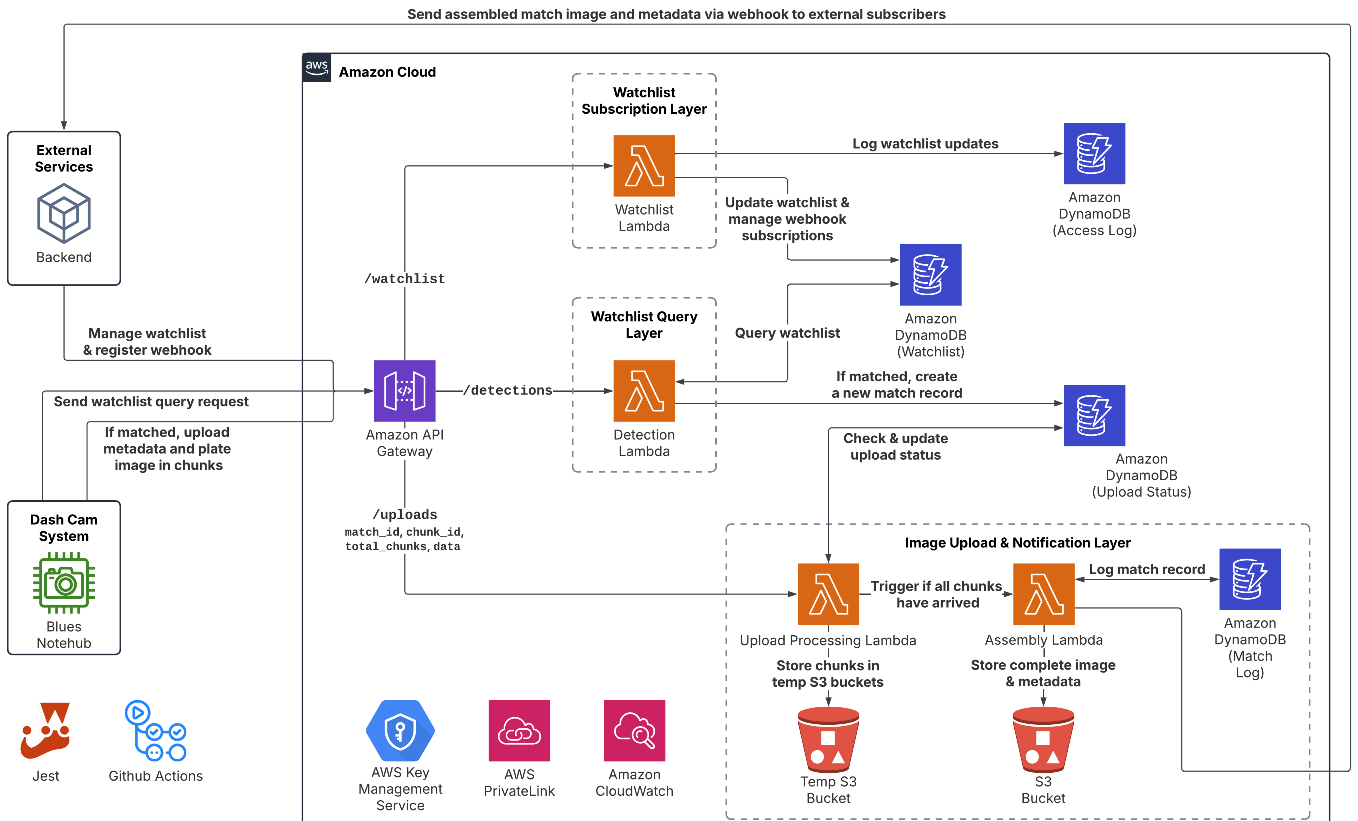
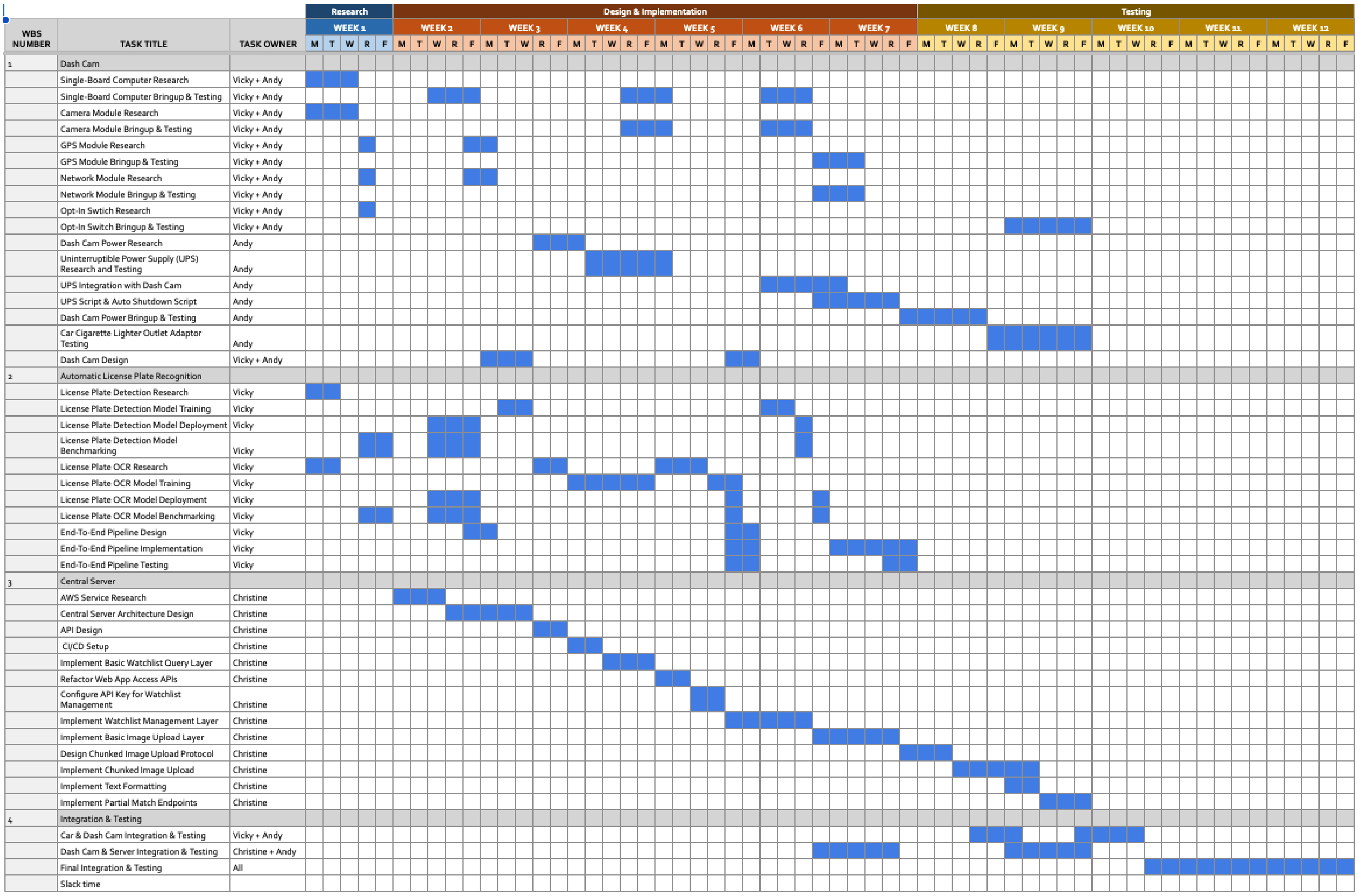

Send assembled match image and metadata via webhook to external subscribers
Amazon Cloud
Watchlist Subscription Layer
Watchlist Lambda
Log watchlist updates
Amazon DynamoDB (Access Log)
Update watchlist & manage webhook subscriptions
Amazon DynamoDB (Watchlist)
External Services
Backend
/watchlist
Watchlist Query Layer
Query watchlist
Manage watchlist & register webhook
Send watchlist query request
/detections
Amazon API Gateway
Detection Lambda
If matched, create a new match record
If matched, upload metadata and plate image in chunks
Check & update upload status
Amazon DynamoDB (Upload Status)
/uploads
match_id, chunk_id, total_chunks, data
Dash Cam System
Blues Notehub
Image Upload & Notification Layer
Trigger if all chunks have arrived
Log match record
Amazon DynamoDB (Match Log)
Upload Processing Lambda
Assembly Lambda
Store chunks in temp S3 buckets
Store complete image & metadata
Temp S3 Bucket
S3 Bucket
Jest
Github Actions
AWS Key Management Service
AWS PrivateLink
Amazon CloudWatch

Figure 3: Central Server Block Diagram



WBS NUMBER	TASK TITLE	TASK OWNER
1	Dash Cam	
	Single-Board Computer Research	Vicky + Andy
	Single-Board Computer Bringup & Testing	Vicky + Andy
	Camera Module Research	Vicky + Andy
	Camera Module Bringup & Testing	Vicky + Andy
	GPS Module Research	Vicky + Andy
	GPS Module Bringup & Testing	Vicky + Andy
	Network Module Research	Vicky + Andy
	Network Module Bringup & Testing	Vicky + Andy
	Opt-In Swtich Research	Vicky + Andy
	Opt-In Switch Bringup & Testing	Vicky + Andy
	Dash Cam Power Research	Andy
	Uninterruptible Power Supply (UPS) Research and Testing	Andy
	UPS Integration with Dash Cam	Andy
	UPS Script & Auto Shutdown Script	Andy
	Dash Cam Power Bringup & Testing	Andy
	Car Cigarette Lighter Outlet Adaptor Testing	Andy
	Dash Cam Design	Vicky + Andy
2	Automatic License Plate Recognition	
	License Plate Detection Research	Vicky
	License Plate Detection Model Training	Vicky
	License Plate Detection Model Deployment	Vicky
	License Plate Detection Model Benchmarking	Vicky
	License Plate OCR Research	Vicky
	License Plate OCR Model Training	Vicky
	License Plate OCR Model Deployment	Vicky
	License Plate OCR Model Benchmarking	Vicky
	End-To-End Pipeline Design	Vicky
	End-To-End Pipeline Implementation	Vicky
	End-To-End Pipeline Testing	Vicky
3	Central Server	
	AWS Service Research	Christine
	Central Server Architecture Design	Christine
	API Design	Christine
	CI/CD Setup	Christine
	Implement Basic Watchlist Query Layer	Christine
	Refactor Web App Access APIs	Christine
	Configure API Key for Watchlist Management	Christine
	Implement Watchlist Management Layer	Christine
	Implement Basic Image Upload Layer	Christine
	Design Chunked Image Upload Protocol	Christine
	Implement Chunked Image Upload	Christine
	Implement Text Formatting	Christine
	Implement Partial Match Endpoints	Christine
4	Integration & Testing	
	Car & Dash Cam Integration & Testing	Vicky + Andy
	Dash Cam & Server Integration & Testing	Christine + Andy
	Final Integration & Testing	All
	Slack time	

Figure 4: Gantt Chart