CASE STUDY 01

Optimising Missed Bin Collection Operations

Process mapping | Workflow automation | AI-assisted route optimisation

Context

A real local-government waste service where missed-bin jobs move between system, paper and field.

My role

Mapped the current process, found the delays, designed a future state and assessed the benefits.

Status

Conceptual improvement. Subject to technical, operational and financial validation.

Phil Black | Business Analyst • Process Improvement • AI Workflow Automation

A manual hand-off chain limits daily capacity

The problem

Missed-bin jobs start in TechnologyOne, are printed, planned by hand, handwritten in the field and re-keyed into TechnologyOne. Every hand-off adds delay and risk.

What I found

Route planning, Google Maps entry and end-of-day updates consume time that could complete jobs. Some jobs carry over to the next day and residents get no completion update.

What I propose

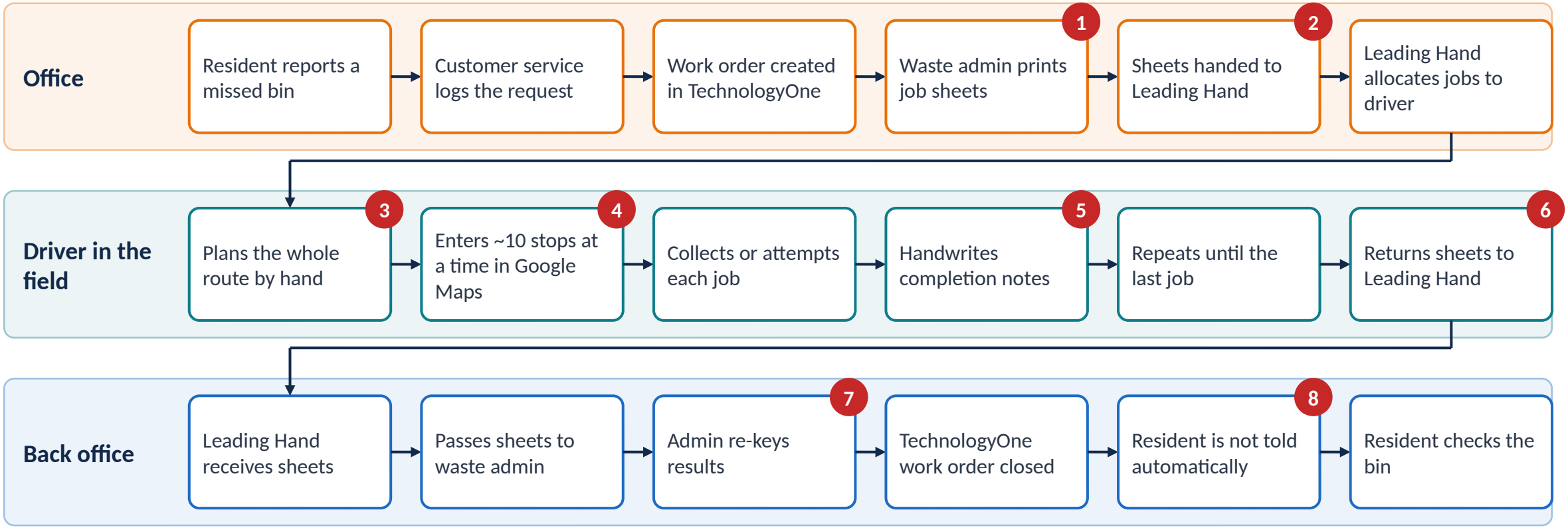
A digital job list with AI-assisted route sequencing, navigation without retyping, in-field completion, automatic TechnologyOne updates and an exceptions dashboard.

Potential impact (estimates)

About 60–75 minutes of driver capacity per day and potentially 10–15 more jobs completed daily, reducing next-day carry-over. To be validated in a pilot.

Opportunity: the value is not only administration savings. Better routing and digital field updates may increase daily completion capacity.

Current state: from system to paper and back



1. Paper job sheets create hand-offs
2. Information leaves TechnologyOne
3. Manual route planning: 30–45 min/day
4. Google Maps entry: ~30 min/day
5. Handwritten notes risk errors
6. End-of-day sheet handover delay
7. Manual TechnologyOne update: ~30 min/day
8. No proactive completion status

Root causes: manual planning and re-entry

Issue	Root cause	Operational effect	Business impact	Priority
Paper job sheets	TechnologyOne data is printed for field use	Physical hand-offs and delayed updates	Lost information and extra administration	**High**
Manual route planning	No integrated route optimisation	Missed or poorly sequenced addresses	Extra km and fuel, lower daily capacity	**High**
Repeated Google Maps entry	Driver keys in groups of up to ~10 stops	Interruptions, rework, lost productive time	~30 min/day lost, fewer jobs completed	**High**
Jobs carried over	Manual planning and inefficient routing	Some jobs slip to the next day	Service delay, backlog and follow-up work	**High**
Handwritten notes	No mobile completion interface	Unclear notes and data-quality risk	Rework, clarification calls, wrong updates	**Medium**
Duplicate TechnologyOne update	Field completion is not synchronised	Admin re-enters completion notes	~30 min/day admin and transcription errors	**High**
No customer status update	Status not linked to customer contact	Resident checks the bin or calls council	More enquiries, poorer experience	**Medium**

Root cause: once a work order is printed, everything downstream depends on manual planning, paper notes and later re-entry. Priorities are an initial assessment to validate with stakeholders.

Where the time goes each day

30–45 min

Manual route planning by the driver

~30 min

Re-entering stops into Google Maps

~30 min

Manual TechnologyOne update by admin

~30 min

Potential travel saving from optimised sequencing

What it could mean

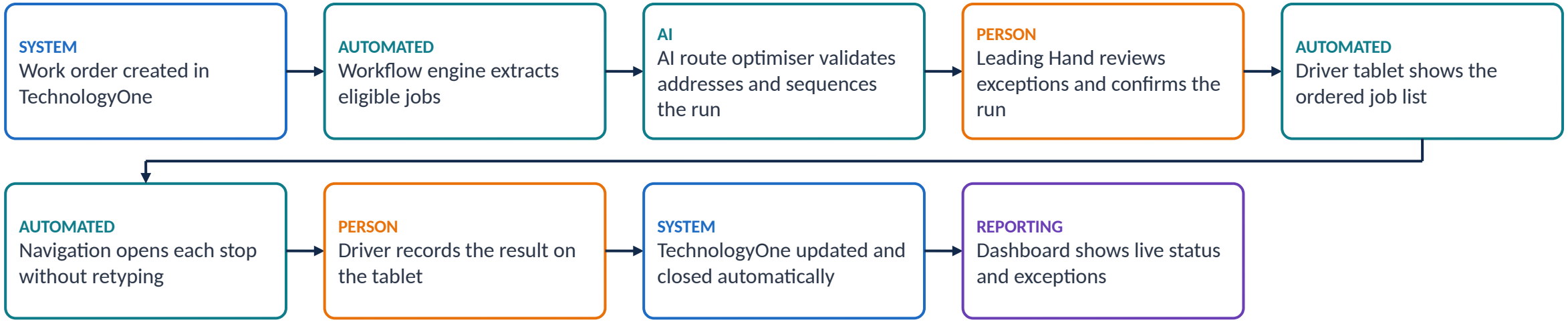
- Removing planning and cutting wasted travel may return ~60–75 min of driver time per day
- That is the basis for the 10–15 extra jobs per day estimate
- Implied marginal time is about 5–6 min per job

To confirm during discovery

- Some time components may overlap, so they are not simply additive
- Actual minutes per job and daily job volumes
- Whether enough waiting jobs exist to use the extra capacity

Observed baseline: combined manual administration was estimated at 90–115 min/day. These are observation-based estimates, not measured data.

Future state: an AI-assisted workflow



Automation points

- Work-order transfer and address validation
- Optimised sequencing and navigation
- Digital completion and automatic TechnologyOne update

Human control retained

- Leading Hand reviews exceptions
- Driver keeps field judgement
- Business rules govern closures; customer notices need approval

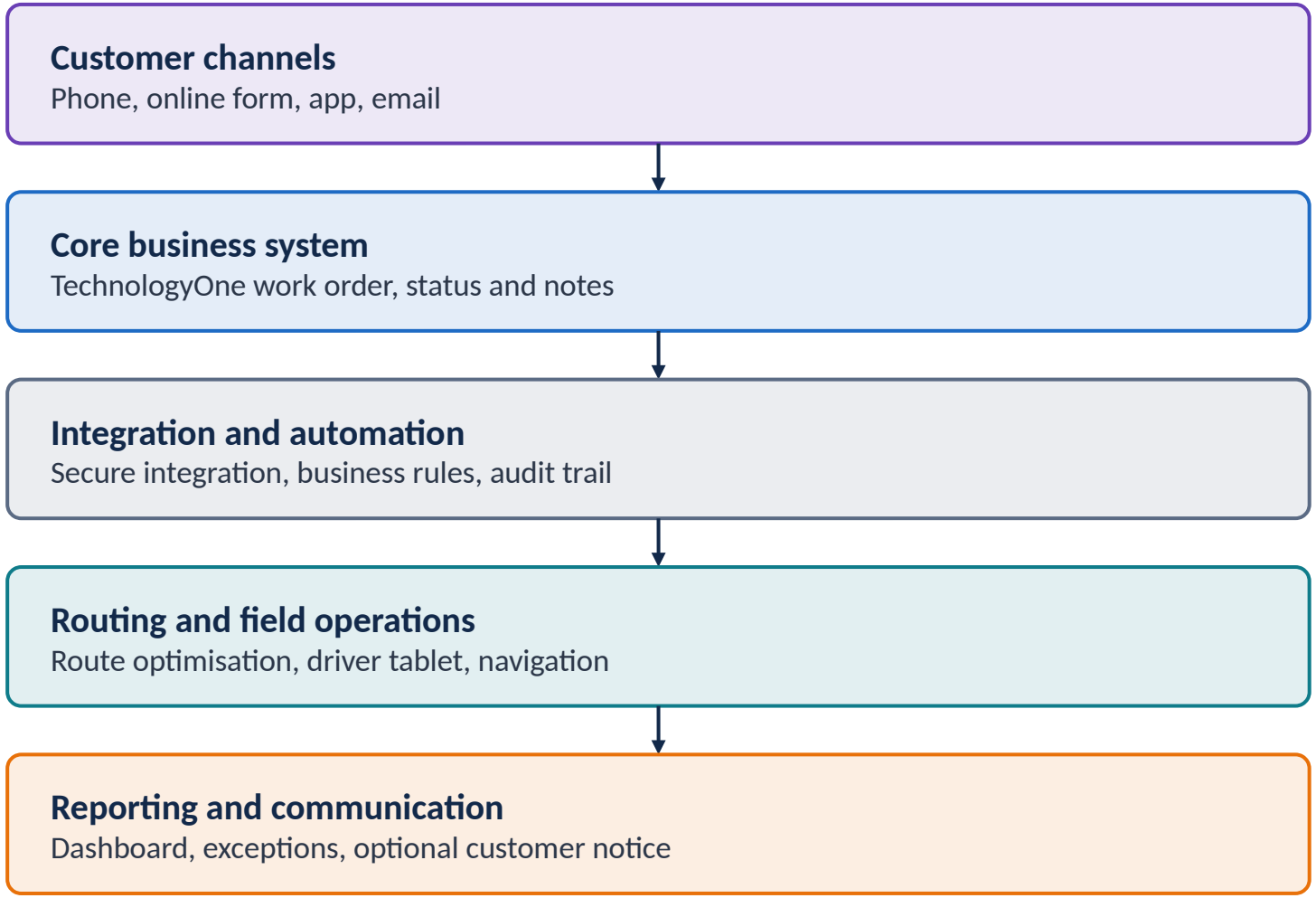
Conceptual: automation handles repetitive sequencing and data transfer while people keep control of exceptions and field decisions.

What changes for each role

Role	Today	Future state
Driver	Plans the route, retypes stops, handwrites notes	Receives a sequenced list, taps to navigate, records the result on a tablet
Leading Hand	Allocates paper sheets and collects them back	Reviews exceptions and confirms each run
Waste admin	Prints sheets and re-keys completion notes	Handles exceptions only. No re-keying
Resident	Checks the bin or calls the council	Optional completion notice, subject to approval
Operations management	Little live visibility of progress or carry-over	Dashboard of progress, carry-over and exceptions

Design intent: the information is entered once, at the point of work, and flows to every role that needs it.

Conceptual solution architecture



Governance

- Access control and privacy
- Device security
- Data retention and audit history
- Integration monitoring
- Business continuity

Questions for ICT and vendors

- Which TechnologyOne integration options exist?
- Which routing engine and licence model?
- How does the app work offline?

Business case: how benefits will be valued

Annual labour benefit
= operating days × hours saved per day × loaded hourly cost

Annual vehicle benefit
= operating days × km avoided × vehicle cost per km

Capacity benefit
= additional completed jobs × avoided next-day processing and follow-up cost

Payback
= one-off investment ÷ net annual benefit

Inputs to baseline first

- Jobs per day and carry-over rate
- Minutes per job
- Planning, entry and admin minutes
- Loaded hourly cost
- Kilometres per route
- Vehicle cost per km

No dollar value is claimed until the baseline is measured and validated.

Implementation roadmap: prove it, then scale

1. Discovery
Measure job volumes, manual time and carry-over

2. Solution design
Define workflow, integration and exception rules

3. Prototype
Test route sequencing and the digital job list

4. Controlled pilot
Compare completion, time saved and carry-over

5. Rollout
Training, devices, support, staged go-live

6. Improvement
Tune routing rules and monitor performance

Stage gates

- Baseline approved
- Design approved
- Prototype successful
- Pilot benefits confirmed
- Rollout approved

Delivery risk	Proposed mitigation
Integration complexity	Confirm integration options in design
Address quality	Validate and clean addresses first
Mobile connectivity	Offline-capable tablet app
User adoption	Involve drivers in the prototype
Vendor costs	Pilot before committing
Privacy and governance	Access controls, retention rules
Over-automation	Leading Hand reviews exceptions

What this case study demonstrates

Business analysis

- Current-state mapping
- Future-state design
- Requirements thinking
- Exception analysis

Process improvement

- Waste identification
- Workflow simplification
- Route efficiency
- Baseline measurement

Digital and AI

- AI opportunity assessment
- Workflow automation
- Route optimisation
- Integration concepts

Business case

- Benefit identification
- Assumption management
- ROI method
- Executive communication

I can observe an operational process, find the inefficiencies, map the current state, design a practical future state and explain the opportunity to operational and executive stakeholders.

Based on a real local-government process. Proposed improvements and estimates are conceptual and subject to validation.

CASE STUDY 02

RFID Bin Audit and Tagging Workflow Automation

Field audit redesign | Tablet-enabled workflow | Live asset data | Exception reporting

Context

A field project to tag and verify waste bins with RFID stickers and correct serial numbers.

My role

Analysed why the paper process failed and designed a tablet-enabled future state.

Status

Concept with functional requirements and a pilot-first roadmap. Not a built system.

Phil Black | Business Analyst • Process Improvement • AI Workflow Automation

The tagging failed because the workflow was static

Business problem

- Printed pages and a paper map
- Address order was hard to follow
- Bin and RFID data was unreliable
- Field exceptions could not be seen live

Field reality

- Bins often not out
- Some addresses did not exist
- Some bins already tagged
- Residents retrieved bins after collection

Project objective

- Improve data accuracy
- Reduce wasted field visits
- Remove manual re-entry
- Show live status and exceptions

Solution theme

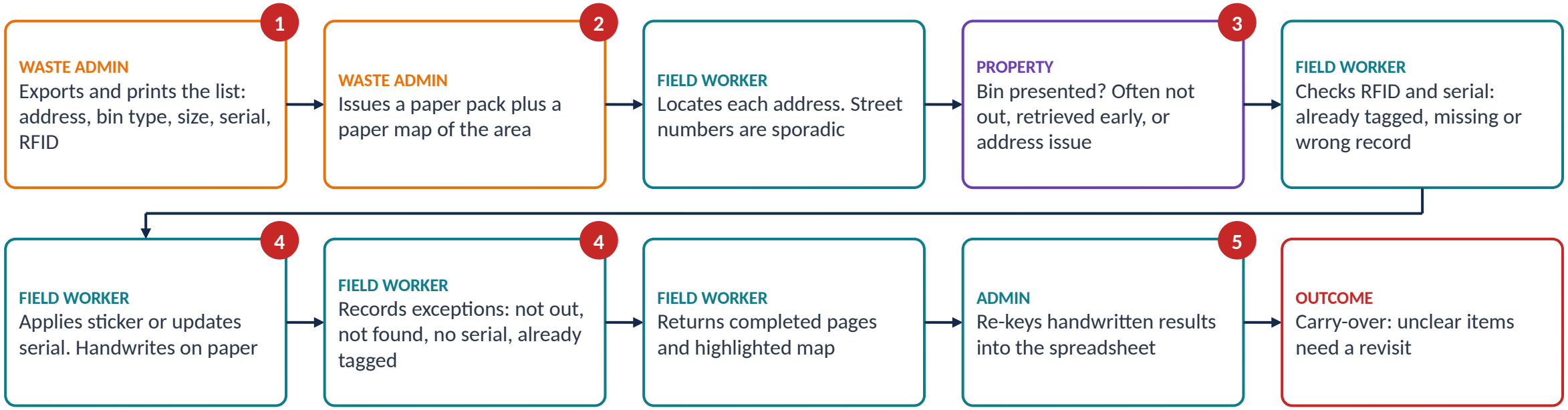
- Tablet worklist
- Live asset data
- Scan and validate
- Sync updates immediately

Recommended future state

Live data from the source system → Optimised field list by route and area → Tablet capture of RFID, serial, status → Exception dashboard → Immediate sync, no end-of-day re-entry

Key message: the issue was not the RFID stickers. It was the disconnected paper workflow and an unreliable data foundation.

Current state: a live data task run on static paper



1. Static printed list: corrections wait until later
2. Paper map tracking: no live view
3. Bins or addresses unavailable at the visit
4. Handwritten RFID, serials and notes can be unclear
5. Manual re-entry: delay and error risk

Paper-based: Yes | **Live progress: No** | **Manual re-entry: Yes** | **Carry-over risk: High**

Key observation: field staff did a live data-quality task with static paper. Issues found at the bin could not update the source record until later re-entry.

Root causes: unreliable data and paper tools

Issue	Root cause	Operational effect	Business impact	Priority
Unreliable source data	Bin serial numbers often wrong or missing	Worker could not trust the printed records	Data-quality risk and repeated corrections	**High**
Paper worklist	Spreadsheet printed and taken into the field	No live updates, filtering or validation	Delayed corrections, manual handling	**High**
Poor address ordering	Street numbers sporadic, not in order	Inefficient movement through streets	More travel time, missed addresses	**High**
Bins not presented	Earlier tagging only captured bins out on collection day	Many properties could not be completed	Carry-over and incomplete coverage	**High**
Already tagged bins	Some had RFID stickers but needed serial updates	Task varied from sticker install to data correction	Confusion, rework, inconsistent outcomes	**Medium**
Manual map highlighting	Progress tracked on a paper map	No real-time management visibility	Hard to monitor coverage and exceptions	**Medium**
Manual re-entry	Admin typed handwritten updates at end of day	Duplicate effort and transcription risk	Delay, errors, administration cost	**High**

Note: priorities are an initial assessment to validate with operations, waste services, ICT and data-governance stakeholders.

Future state: tablet-enabled RFID audit



Tablet decision logic



Management visibility

- Live counts of completed, skipped and exception records
- Street-level coverage map
- Carry-over list generated automatically

Field worker benefit

- No paper pages to update
- Less manual searching through lists
- Clear next address and required action on screen

Digital field app: functional requirements

Live worklist

- Address and property reference
- Bin type and size
- Known serial and RFID
- Collection day and area
- Current status and priority

Field capture

- RFID sticker serial scan
- Serial capture or no-serial flag
- Bin type and size check
- Notes and exception reason
- Photo, GPS and time stamp if required

Validation rules

- Warn on duplicate RFID
- Warn on unexpected bin type
- Require a reason if not completed
- Separate already tagged from new sticker
- Flag missing or unreadable serials

Sync and control

- Offline capable
- Conflict handling
- Audit log
- User ID and time stamp
- Supervisor review queue

Design principle: capture the field truth once, at the point of observation, and synchronise it to the live source record with an audit trail.

Recommended field statuses

Sticker Applied

RFID installed and record updated

Already Tagged

Existing RFID recorded or verified

Serial Updated

RFID present, serial corrected

Bin Not Out

Resident did not present the bin

Address Not Found

Property could not be located

No Serial

Bin lacks a readable serial

Wrong Bin Type

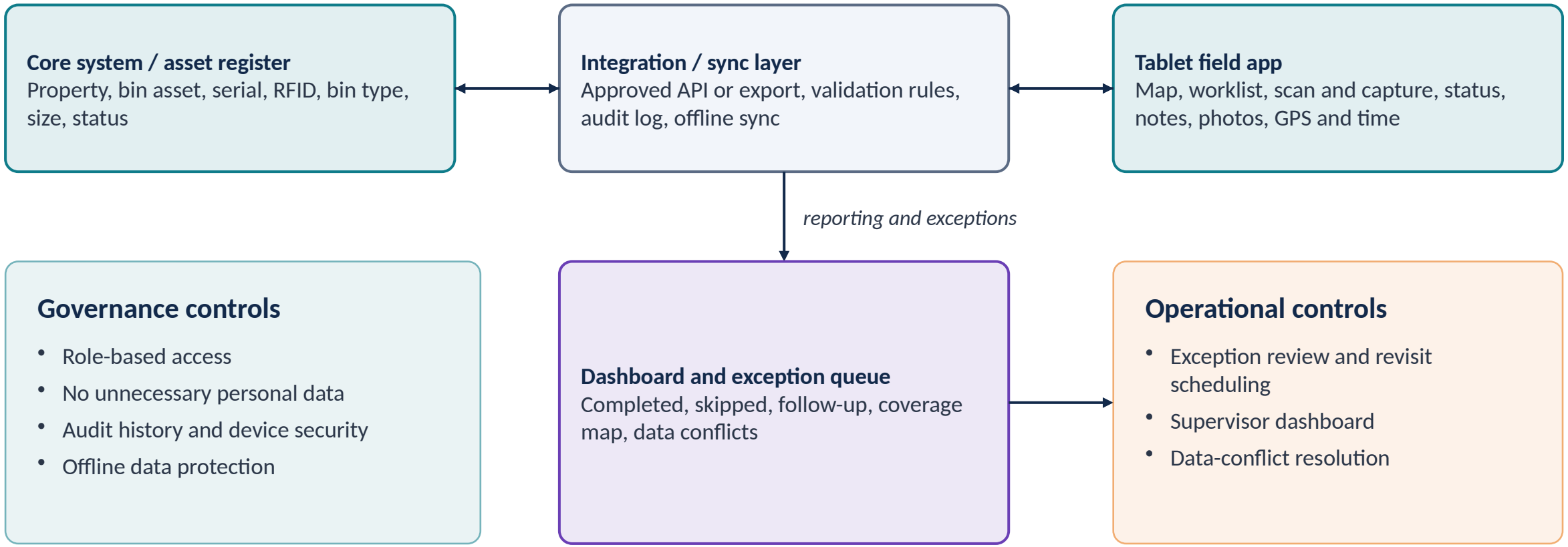
Expected and actual bin differ

Follow-Up Required

Supervisor or admin review needed

Why separate statuses matter: each status tells operations what happens next: finished, revisit, correct the data, or escalate. Unclear items become actionable jobs instead of silent carry-over.

Conceptual solution architecture



Architecture note: the integration method needs confirming with ICT, system owners and vendors. This shows the target information flow, not a confirmed implementation.

Benefits: better completion and data quality

Less manual admin

Higher data accuracy

Fewer revisits

Live progress view

Better coverage

Benefit area	How it improves	Business value
Field efficiency	Live map and worklist reduce searching, wrong addresses, paper handling	More productive time on valid jobs
Data quality	Serial and RFID captured once at the bin and validated before sync	Cleaner asset register, fewer correction tasks
Completion rate	Exceptions recorded immediately. Revisits scheduled from a filtered list	Less carry-over: unclear items separated from actionable jobs
Administration	No end-of-day transcription from handwritten pages	Less duplicate entry, lower error risk
Management visibility	Dashboard of completed, skipped, not-out and conflict records	Better planning, reporting and accountability

Financial model: quantify admin time, revisits, kilometres, completion rate and rework after a small pilot validates the baseline.

Implementation roadmap: pilot before scaling

1. Baseline
Confirm steps, volumes, error types, time spent

→

2. Data review
Audit bin, property, RFID and serial quality

→

3. Prototype
Tablet mock-up, status list, sample worklist

→

4. Pilot
Small-area trial with field feedback

→

5. Refine
Improve statuses, map flow, validation, reporting

→

6. Rollout
Train staff, migrate, monitor adoption

Pilot success measures

- Completed records per day
- Not-out and exception rate
- Manual re-entry time removed
- Data-conflict rate
- Revisit backlog reduced
- Field worker feedback

Delivery risk	Proposed mitigation
Poor source-data quality	Audit existing data in stage 2
Integration limitations	Confirm sync options with ICT early
Offline mobile coverage	Offline-capable app with later sync
Change adoption by field staff	Involve staff in the prototype
Device handling and security	Device policy and role-based access
Unclear exception ownership	Assign queue owners before the pilot

Recommendation: start with a controlled pilot in one suburb or collection area, validate the data model and status options, then scale once usability and data quality are proven.

What this case study demonstrates

Business analysis

- Current-state analysis
- Problem definition
- Requirements thinking
- Exception handling

Process improvement

- Waste identification
- Paper-to-digital redesign
- Rework reduction
- Field-work optimisation

Digital solution design

- Tablet workflow
- Live worklists
- Validation rules
- Dashboard concepts

Operational insight

- Collection-day constraints
- Bin asset data
- Field exceptions
- Implementation risks

I can observe a failing field process, find the underlying data and workflow issues, design a practical tablet-enabled future state and present it to operational and executive stakeholders.

Concept only. Integration method, costs and benefits need confirming with ICT, system owners and a pilot.

CASE STUDY 03

Automated Cryptocurrency Trading Bot

Python development | API integration | Backtesting | Signal automation | Cloud concepts

Context

A personal learning project that automates a repeated, rules-based decision process.

My role

Directed requirements and strategy logic, built modules with AI-assisted programming, tested and debugged.

Important

Not financial advice. No claim of guaranteed or optimised trading performance.

Turning a manual routine into a controlled workflow

Business problem

Manual trading needs constant chart monitoring, repeated indicator review, rapid order entry and emotional discipline. Decisions become inconsistent and hard to scale.

Project objective

Build a rules-based system that retrieves market data, calculates indicators, generates buy and sell signals, backtests and executes approved spot trades.

Solution approach

A modular Python platform with separate components for market data, indicators, signal logic, execution, logging and backtesting.

Outcome

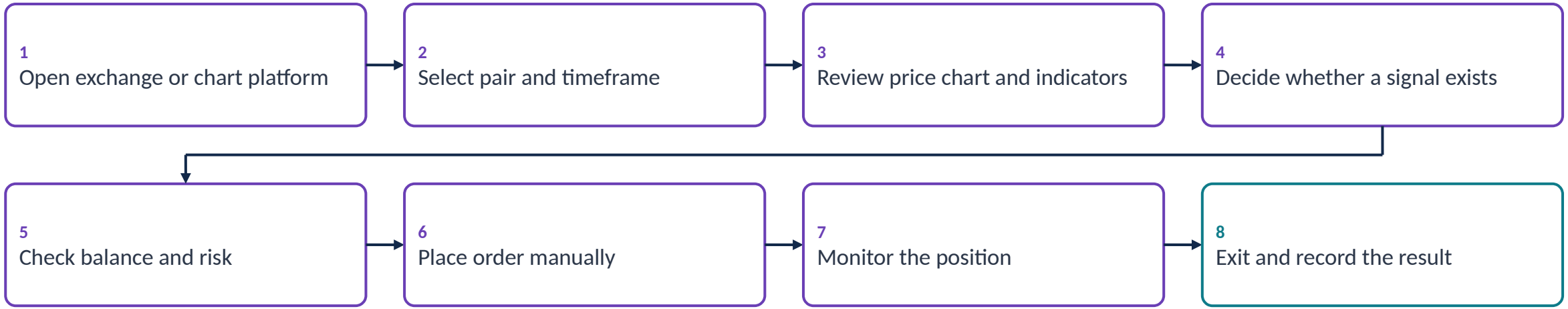
A working personal project showing automation, API integration, troubleshooting, an exchange migration and cloud-deployment concepts.

The pattern: one controlled workflow



Transferable: the same break-down mindset applies to business systems, operations and process-improvement work.

As-is: the manual trading process



Repeat continuously as markets change: difficult to scale, test or monitor consistently

Key limitations

- Time intensive
- Constant monitoring
- Slow reaction to signals
- Manual order-entry errors

Decision-quality risks

- Emotional bias
- Inconsistent criteria
- Limited repeatability
- Hard to compare scenarios

Process opportunity

- Convert chart review into rules
- Automate calculations
- Backtest before use
- Log actions and exceptions

Requirements and design objectives

Trading scope

- Spot trading only
- Long-only strategy
- No leverage
- Buy and sell by rule

Data and indicators

- Retrieve candles
- Support 5m, 1h, 2h
- Calculate indicators
- Handle exchange formats

Strategy engine

- Deterministic signals
- Fee allowance
- Balance validation
- Duplicate-signal protection

Operations

- Logging
- Error handling
- API keys in environment variables
- Cloud-ready runtime

Design principle: separate exchange-specific code, indicator calculations, strategy decisions and execution so each can be tested and changed independently when APIs or assumptions change.

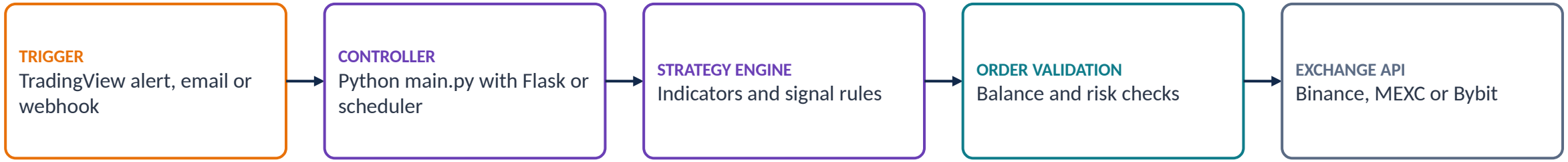
Modular
separated responsibilities

Spot only
no leverage

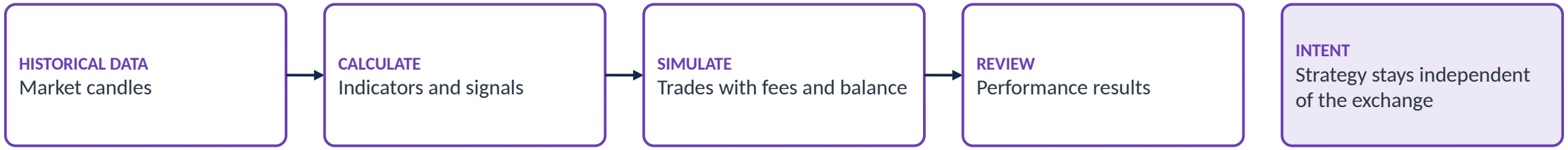
Backtest
before live use

Env vars
credential control

Solution architecture



Backtesting workflow



Important note: this was built as a personal project and learning system. Any real-money use would need stronger controls, monitoring, credential protection, paper-trading validation and risk limits.

Modular Python design

Module	Purpose	Capability demonstrated
binance_data_fetcher.py	Retrieve historical Binance candlestick data	API use and data extraction
MEXC data module	Retrieve exchange data after migration	Exchange adaptation
heikin_ashi.py	Convert standard candles into Heikin Ashi values	Data transformation
chandelier_exit.py	Calculate Chandelier Exit logic	Indicator implementation
zlsma.py	Calculate Zero-Lag LSMA	Indicator implementation
check_buy_sell_signals.py	Evaluate combined strategy rules	Decision logic
execute_trade.py	Submit buy and sell orders through the API	Execution automation
backtest.py	Simulate historical trades and fees	Testing and validation
main.py / main_rsi.py	Coordinate the workflow and RSI variation	System orchestration

Design value: market data, indicators, strategy, execution and testing can change independently rather than living in one large script.

Trading strategy logic

Heikin Ashi

Reduces short-term noise and smooths candle behaviour

Chandelier Exit

ATR-based exit logic. Period tested around 22, multiplier around 3

ZLSMA

Trend filter, commonly tested around a 50-period setting

RSI

Momentum confirmation. 14-period window, thresholds tested around 50–55

Rules-based decision flow



Tested assumptions

- Spot, long-only. 100% simulated capital per trade
- Full position sold on exit signal. Fee allowance ~0.1%

Positioning: these are tested settings and design choices, not guaranteed or optimised trading parameters.

Backtesting process



8. Refine strategy, then repeat

Backtest assumptions

- Starting balance 1,000 USDT
- Fee rate 0.001
- Full allocation per simulated trade
- Full sell on exit signal

What it measures

- Trade history
- Balance change
- Behaviour by timeframe
- Impact of fees and signals

What it does not prove

- Future profitability
- Market stability
- Perfect execution
- Exchange availability

Business-analysis lesson: backtesting turns assumptions into measurable test scenarios, which supports better decisions than intuition alone.

Development challenges and resolutions

Challenge	Investigation and response	Capability demonstrated
Exchange data formats differed	Revised dataframe mapping and parsing rules	Problem diagnosis
MEXC returned 8 columns, not 12	Adjusted column expectations and validation	Data handling
APIs needed epoch timestamps	Converted dates to exchange-compatible values	Technical troubleshooting
Libraries had different methods	Compared clients and exchange-specific APIs	Research and adaptation
Cloud start-up failures	Checked the Flask health endpoint and deployment logs	Runtime debugging
API keys missing in cloud runtime	Moved credentials to environment variables	Security awareness
Email alert filtering	Processed unread alerts with specific subjects	Automation design
Exchange migration changed rules	Separated exchange logic from strategy logic	Modular design

Why it is included: real projects are rarely perfect. This shows troubleshooting, structured learning and persistence through ambiguity.

Cloud and automation architecture

External trigger
TradingView alert, email alert or webhook

↓

Cloud service
Google Cloud Run with a Python Flask runtime

↓

Endpoints and monitoring
/health, /trigger and logs

↓

Bot workflow
Signal handling, validation, execution decision

↓

Exchange connection
CCXT or exchange client with secure credentials

Deployment objective

Let the bot run without a home PC staying on. The cloud service receives triggers, runs validation and records logs.

Minimum production controls

- Managed environment variables
- Logging, alerts and duplicate-signal protection
- API keys without withdrawal rights

Not production-grade without

- Monitoring and recovery rules
- Paper-trading validation
- Error escalation, position and exposure limits

Risk, security and governance

Risk area	Example risks	Practical controls
Market risk	Losses, volatility, strategy overfitting	Spot only, paper trading, conservative limits
Credential risk	API key exposure or misuse	Environment variables, no withdrawal permission
Execution risk	Wrong quantity, duplicate signal, partial fill	Validation, logs, duplicate checks
Exchange risk	Outage, rate limit, changed API rules	Error handling, retry logic, monitoring
Data risk	Bad candles, missing values, precision errors	Input validation, exchange-specific parsing
Operational risk	Cloud failure, missed alerts, unstable runtime	Health checks, logs and alerting
Decision risk	Backtest not representative of the future	Test across timeframes, avoid profit claims

Governance: automation should reduce repetitive work, but it must include validation, monitoring and human oversight.

What this project shows a hiring manager

What I did in the project	Business analyst skill it demonstrates
Broke a manual routine into explicit rules	Process decomposition and requirements definition
Separated exchange logic from strategy logic	Designing for change and reducing rework
Backtested assumptions before any execution	Evidence-based decisions and test scenarios
Diagnosed data-format, timestamp and deployment faults	Root cause analysis and structured troubleshooting
Kept credentials out of code, no withdrawal rights	Security and risk awareness
Documented architecture, limits and risks plainly	Communicating complex systems to non-specialists

Honest scope: a personal learning project with AI-assisted coding. I directed the design and testing and can explain every component.

What this case study demonstrates

Systems analysis

- Process decomposition
- Requirements thinking
- Rule-based design
- Exception handling

Technical learning

- Python and APIs
- CCXT and dataframes
- Flask and Cloud Run concepts
- AI-assisted development

Testing

- Historical backtesting
- Parameter testing
- Error diagnosis
- Iterative refinement

Governance

- Environment variables
- API-key controls
- Risk identification
- Human oversight

I can learn unfamiliar technology, design modular automation, troubleshoot real technical issues and explain a complex system in a structured business-analysis format.

Personal learning project. Not financial advice. No guaranteed trading performance.

CASE STUDY 04

Improving Animal Management Incident Data Quality and Reporting

Excel data validation | Controlled data entry | Power BI reporting | Hotspot analysis

Context

A dog-attack incident register kept in Excel and used for management reporting.

My role

Redesigned data entry with controlled lists, then built Power BI reporting on the cleaner data.

Result

A controlled capture process and an interactive report for patrol and hotspot decisions.

Phil Black | Business Analyst • Process Improvement • AI Workflow Automation

Fix the data at entry, not at reporting time

Key message

The improvement moved the process from manual clean-up after entry to preventing incorrect data from being entered at all.

Business problem

Incidents were typed into free-text Excel fields. Misspellings and inconsistent entries made hotspot analysis and reporting unreliable.

Solution

Controlled suburb and breed lists, suburb-specific street and park dropdowns, required officer details and structured location categories.

Outcome

Cleaner incident data and a Power BI report that helps management interrogate trends, locations and hotspots consistently.

Current state: free text feeds unreliable reports

1. OFFICER
Records the incident

→

2. EXCEL REGISTER
Suburb, street and breed typed manually

→

3. DATA QUALITY
Spelling variations and duplicate categories

→

4. REPORTING
Manual filtering and correction

→

5. MANAGEMENT
Hotspots unclear, patrols lack location insight

Illustrative: one suburb, four entries, four rows in a report

Springfield	Sprinfield	Springfeild	Springfield (trailing space)

Poor data examples

- Suburb misspellings
- Street-name variations
- Streets linked to wrong suburbs
- Inconsistent breed names

Reporting impact

- Same place split across categories
- Hotspots not visible
- Trends unreliable
- Manual interpretation needed

Operational impact

- Hard to target patrols
- Slower monthly reporting
- Weaker evidence for decisions
- More follow-up and cleaning

Key observation: the spreadsheet acted as a database but allowed uncontrolled entry, so this was a data-capture problem.

Root causes: uncontrolled entry

Issue	Root cause	Operational effect	Business impact	Priority
Inconsistent suburbs	Free-text entry	Duplicate suburb values	Unreliable geographic reporting	**High**
Incorrect streets	No suburb-to-street validation	Wrong or misspelt addresses	Incorrect hotspot mapping	**High**
Inconsistent breeds	Free-text breed field	Duplicate breed categories	Inaccurate breed analysis	**High**
Unclear attack location	No standard location classification	Descriptions cannot be grouped	Cannot compare location types	**High**
Limited accountability	Officer details not consistently captured	Difficult follow-up	Weak audit trail and record ownership	**Medium**
Manual reporting	Repeated spreadsheet manipulation	Reporting delays	Reduced management visibility	**High**

Root-cause statement: the issue was not the reporting tool. The source data was inconsistent, so quality had to be controlled at the point of entry.

Improved data entry: controlled inputs

1. Create incident record
2. Select suburb from approved list
3. Select street, filtered to suburb
4. Select breed from standard list
5. Select location type
6. Enter officer name and number

Illustration: register entry form

Suburb	Select from approved list ▼
Street or park	Filtered to the chosen suburb ▼
Dog breed	Select from standard list ▼
Location type	Private property / outside property / park ▼
Officer name	Required
Employee number	Required

Dependent dropdowns

The suburb choice controls which streets and parks appear.

Controlled vocabularies

Breed and location fields accept standard values only.

Mandatory accountability

Officer name and number support audit and follow-up.

Clean source data

Power BI receives consistent fields it can group reliably.

Data-control model

Reference tables

- Approved suburbs
- Streets and parks mapped to suburb
- Standard dog breeds
- Location categories
- Officer register

Validated Excel register

- Dropdown-only fields
- Suburb-to-street dependency
- Required officer details
- Consistent categories
- Fewer manual corrections

Power BI dataset

- Clean categories
- Reliable filters
- Drill-down reporting
- Hotspot analysis
- Management dashboards

Data quality rule

Users choose approved values instead of typing free text.

Validation rule

The street and park list changes with the selected suburb.

Reporting rule

Power BI uses standard categories and cleaner relationships.

Insight: a simple master-data model inside Excel made reporting reliable without replacing the whole system.

Power BI reporting

Geographic analysis

- Incidents by suburb and street
- Hotspot map
- Drill-down to location

Incident analysis

- Trends over time
- Breed and location type
- Filters and slicers

Management reporting

- Monthly reporting
- Patrol targeting
- Data completeness and officer follow-up

Power BI management view

Consistent categories enable filtering, drill-down and evidence-based hotspot analysis

Decision support

Target patrols using reliable incident clusters.

Reporting efficiency

Less manual manipulation before monthly reports.

Governance

Track data completeness and officer responsibility.

Benefits and how to measure them

Business benefits

- Data quality: fewer spelling errors and duplicate categories
- Efficiency: less time cleaning and correcting data
- Insight: more reliable hotspots by suburb, street and location type
- Accountability: officer name and number improve the audit trail
- Targeted patrols: evidence-based decisions on where patrols are needed
- Scalable reporting: the Power BI report can be refreshed and reused

Proposed success measures (to baseline)

Measure	What it shows
Records with a valid suburb and street pair	Whether dependent dropdowns are used
Records with officer name and number	Accountability and audit trail
Duplicate suburb or breed values	Whether free-text variants are gone
Hours spent cleaning data per month	Efficiency gain
Time to produce the monthly report	Reporting speed

Note: these measures are proposed. Baselines were not recorded in the original project.

Possible next steps (not yet delivered)

Keep the lists current

- Name an owner for the reference tables
- Agree how new streets, parks and breeds are added
- Review lists on a set schedule

Tighten validation

- Validate dates and required fields
- Protect reference sheets from edits
- Schedule the Power BI refresh

Review the platform

- Consider a form or database if volumes grow
- Capture location by map or GPS
- Confirm privacy and retention rules

Why list these: a good solution has an owner and a path to improve. These are suggestions for discussion, not part of what was delivered.

What this case study demonstrates

Business analysis

- Identified root causes
- Turned management needs into data requirements
- Redesigned the capture process

Excel solution design

- Data validation
- Dependent dropdown lists
- Reference tables
- Controlled vocabularies

Power BI reporting

- Dashboard design
- Filters and drill-down
- Hotspot analysis
- Management reporting

Data governance

- Standardised categories
- Required officer details
- Improved audit trail
- Reusable reference data

I improve business outcomes by controlling data quality before it reaches reporting and management decision-making.

Success measures in this deck are proposed. Original baselines were not captured.

Operational reporting for local government managers

A selection of the Power BI report designs I created for department managers, shown here with illustrative sample data.

The business problem

Reports for managers to track review and appeal wait times against targets, understand why infringements were withdrawn, and monitor prosecution outcomes and costs.

What the dashboards do

Four report pages: wait times against target, parking review volumes and withdrawals, review backlog by department, and prosecution results and costs. Each page answers one set of management questions.

Design choices

Targets are shown as lines on every trend. A single week-ending slicer controls the period. KPI cards give the number managers look at first. This sample also highlights values that are above target.

Skills shown

Power BI report design, KPI and target tracking, trend and variance analysis, and presenting data to non-technical stakeholders.

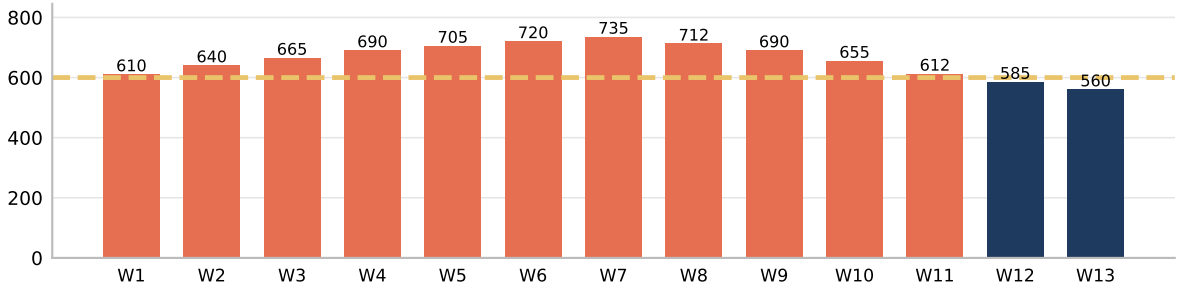
Data note: all figures, dates and labels in this document are illustrative. Layouts are based on reports I designed; no organisation names or real data are included.

Infringement Review | Wait Time Dashboard

Wait Times | Parking | Reviews | Prosecutions

Orange bars are above target. Dashed gold line is the target.

REVIEW WAIT TIME < 30 DAYS (outstanding vs target 600)



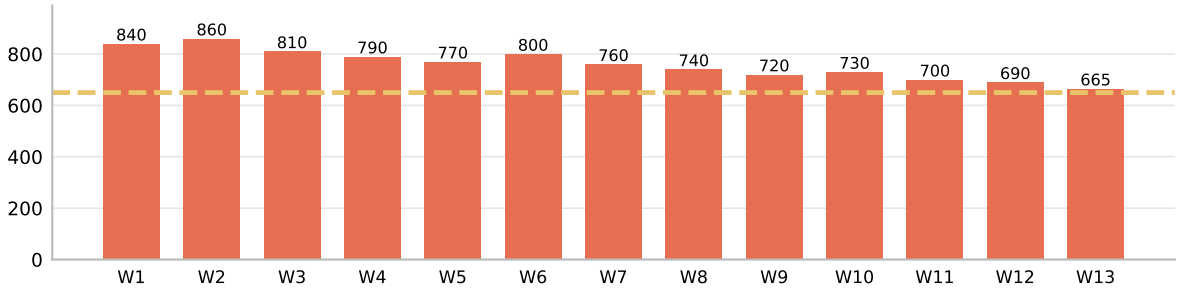
Week ending

Last 13 weeks (slicer)

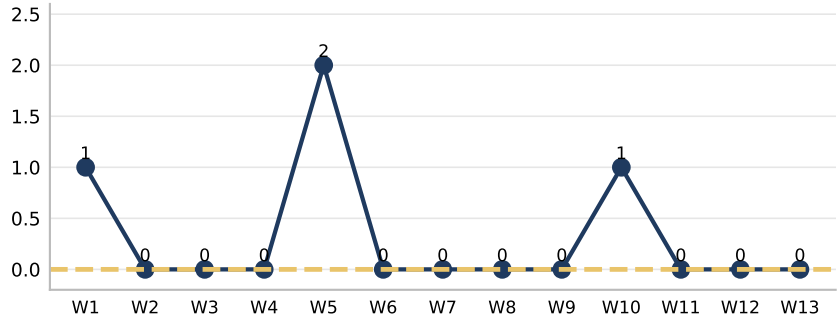
21

DAYS TILL 90 DAYS

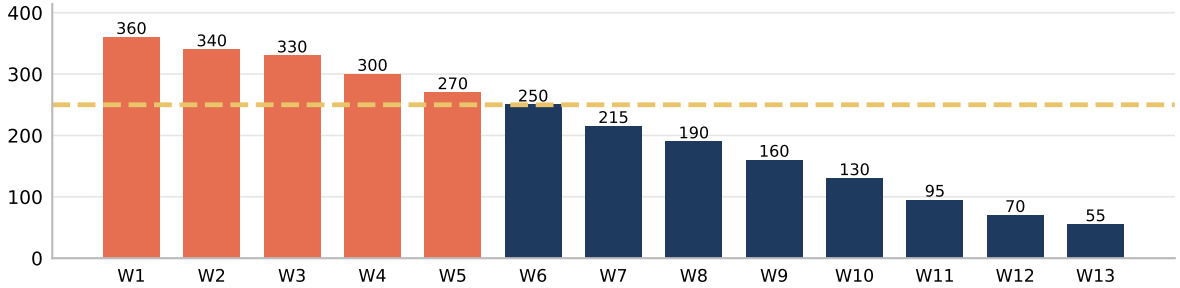
REVIEW WAIT TIME 31-60 DAYS (outstanding vs target 650)



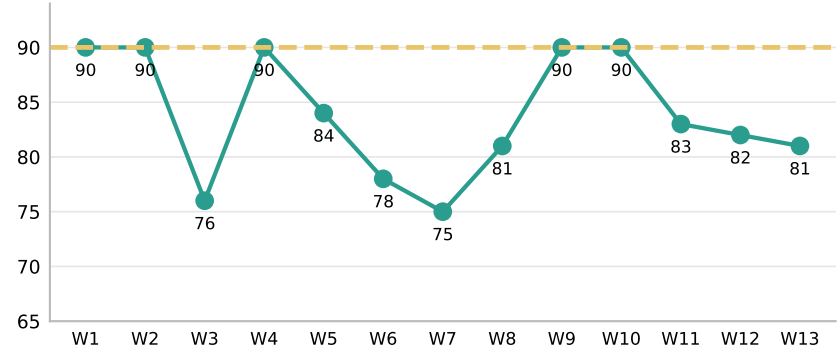
APPEAL WAIT TIME OVER 90 DAYS



APPEAL WAIT TIME 61-90 DAYS (outstanding vs target 250)



PARKING PIN: DAYS BEFORE 90-DAY TIME FRAME



Sample dashboard built to illustrate Power BI report design. All figures are illustrative; no organisation data is shown.

Infringement Review | Parking Dashboard

Week ending

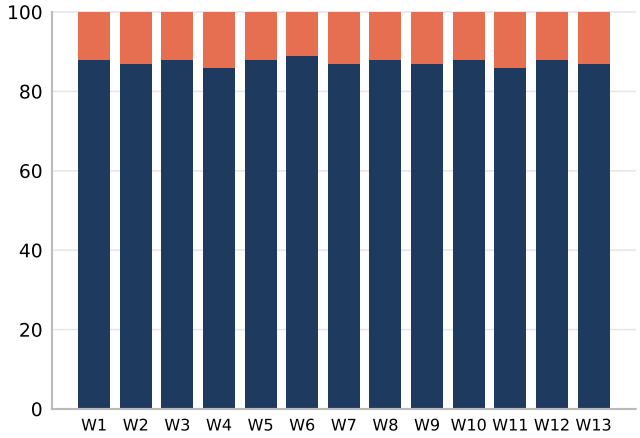
Last 13 weeks (slicer)

REASONS FOR WITHDRAWAL



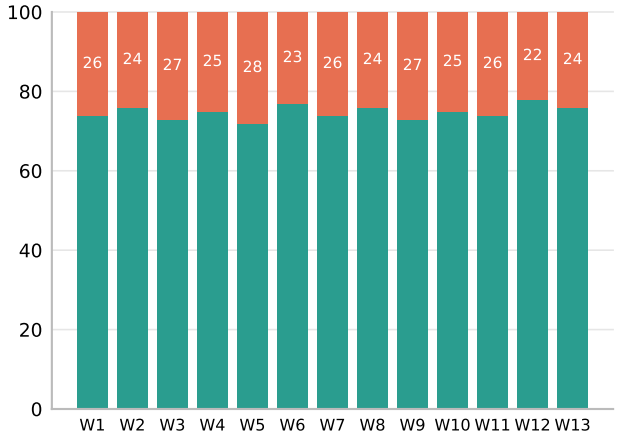
- Mistake of identity: 36 (24%)
- Special circumstances: 31 (21%)
- Exceptional circumstances: 30 (20%)
- Contrary to law: 28 (19%)
- Officer error: 24 (16%)

ISSUED vs REVIEWS RECEIVED (%)



Navy: no review Orange: review received (about 12%)

COMPLETED vs WITHDRAWN (%)

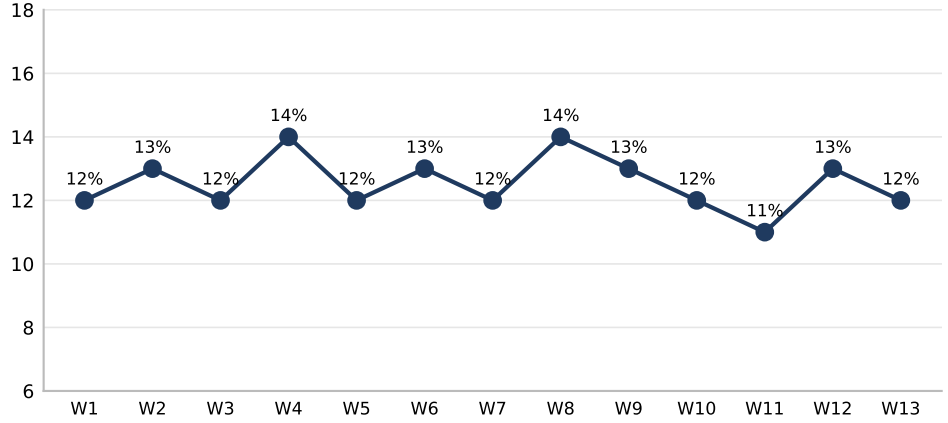


Teal: completed Orange: withdrawn (%)

REVIEWS RECEIVED, COMPLETED AND OUTSTANDING



REVIEWS RECEIVED AS % OF INFRINGEMENTS ISSUED



Sample dashboard built to illustrate Power BI report design. All figures are illustrative; no organisation data is shown.

Infringement Review | Review Dashboard

Week ending

Last 13 weeks (slicer)

REVIEW WAIT TIME < 30 DAYS

600	560
TARGET	OUTSTANDING

REVIEW WAIT TIME 31-60 DAYS

650	665
TARGET	OUTSTANDING

REVIEW WAIT TIME 61-90 DAYS

250	55
TARGET	OUTSTANDING

21

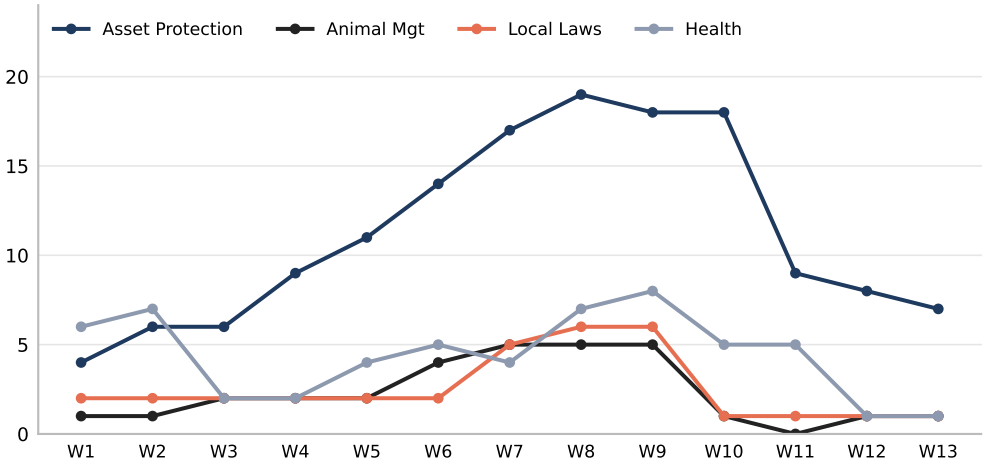
DAYS TILL 90 DAYS

REASONS FOR WITHDRAWAL

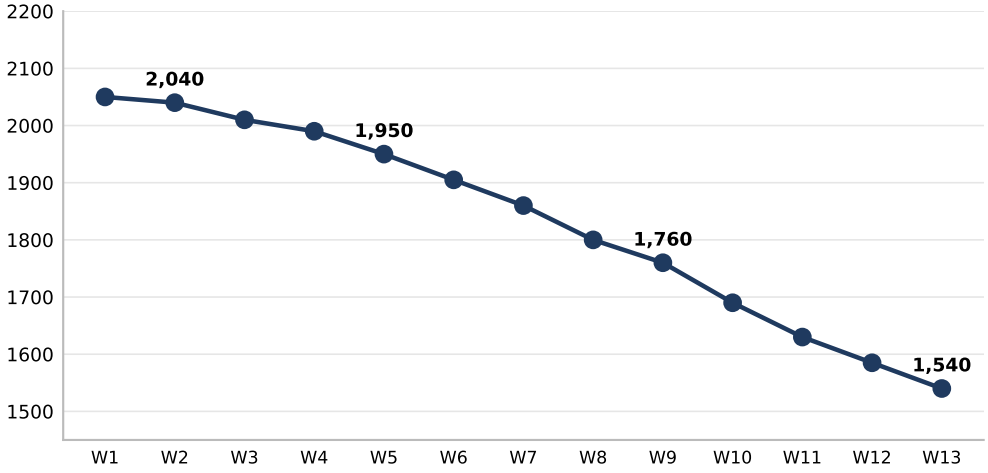


- Mistake of identity (34)
- Special circumstances (31)
- Exceptional circumstances (28)
- Contrary to law (26)

REVIEWS OUTSTANDING BY DEPARTMENT



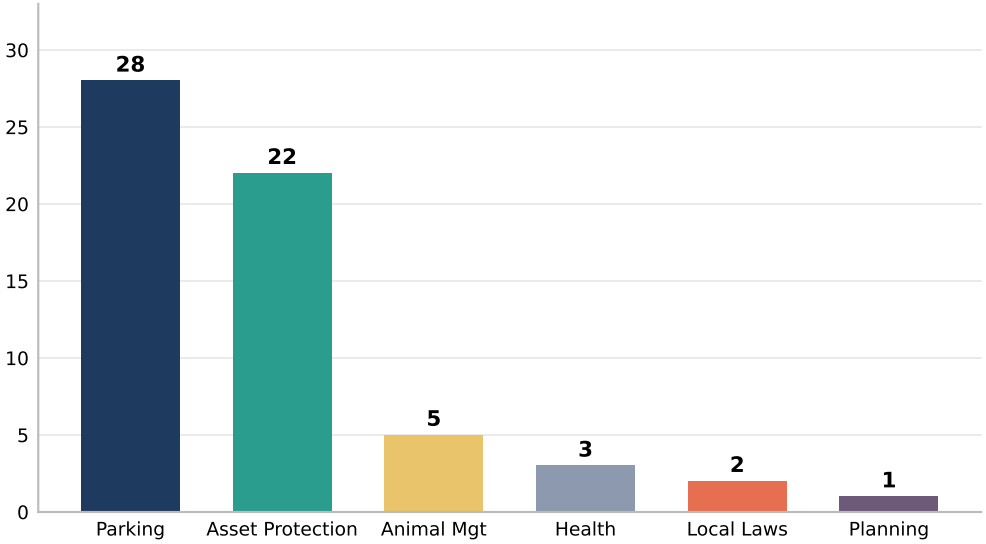
PARKING REVIEWS OUTSTANDING



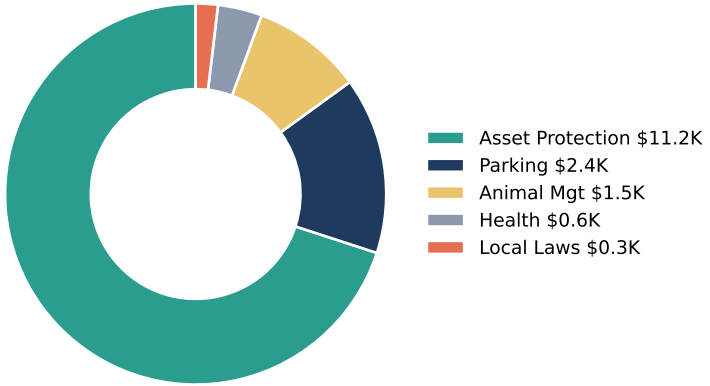
Sample dashboard built to illustrate Power BI report design. All figures are illustrative; no organisation data is shown.

Prosecution Outcomes Dashboard

NUMBER OF CASES BY DEPARTMENT

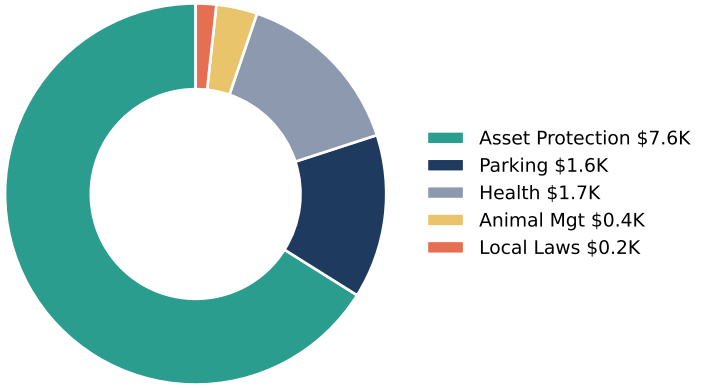


FINES IMPOSED BY COURT



61 TOTAL MATTERS HEARD	**59** TOTAL MATTERS SUCCESSFUL	**97%** SUCCESS RATE
$3,100 COSTS FOR PROSECUTING	**$27K** TOTAL AWARDED (INC COSTS)	**$24K** NET POSITION
18 SETTLED OUT OF COURT	**$15K** REVENUE OUT OF COURT	

COSTS AWARDED



Sample dashboard built to illustrate Power BI report design. All figures are illustrative; no organisation data is shown.