



# RACING SCRAPER

Malavika Menon  
Kudzai Netsiyanwa  
Mohammed Ibraheem Khaleel

Data and Scraper  
University of Sunshine  
Coast

## **EXECUTIVE SUMMARY**

*The necessity for a thorough examination of the price movements in the greyhound racing industry is addressed in this project. It seeks to overcome the difficulties of length and time-consuming processes that are currently required by developing a system that effectively gathers and assesses. To gather information on race events, greyhound profiles, and odd changes, the system uses Python for web scraping and integrates directly with Punters.com.au API. A notification for notifying users of price movements and the completion of scraping tasks is also included, along with automatic data processing and storage in a SQLite database and ongoing odds monitoring.*

## **TABLE OF CONTENTS**

<b>BACKGROUND:</b>	6
<b>PROBLEM STATEMENT:</b>	6
<b>GOALS AND OBJECTIVES:</b>	7
1. <i>Data Scraping</i>	7
2. <i>Continuous Monitoring (current system)</i>	8
3. <i>Data Processing and Storage</i>	8
4. <i>Notification System</i>	8
5. <i>Data Download</i>	9
<b>PROJECT SCOPE:</b>	9
<b>METHODOLOGY:</b>	<b>Error! Bookmark not defined.</b>
A. <b>Web Scraping Framework</b>	11
B. <b>WEB SCRAPING FOR RESULTS:</b>	12
C. <b>DATA PROCESSING AND STORAGE</b>	15
D. <b>API END-TO-END INTEGRATION</b>	16
E. <b>EMAIL AUTOMATION SYSTEM</b>	17
<b>MAJOR DELIVERABLES:</b>	19
<b>SYSTEM ARCHITECTURE:</b>	21
<b>RESOURCES REQUIRED:</b>	22
<b>PROJECT TIMELINE:</b>	23
<b>CHALLENGES:</b>	25

<b><i>OPPORTUNITIES:</i></b> .....	27
<b><i>EVALUATION METRICS:</i></b> .....	28
<b><i>ROLES AND RESPONSIBILITIES:</i></b> .....	30
Malavika Menon .....	30
Kudzai Netsiyanwa .....	30
Mohammed Ibraheem Khaleel .....	30
<b><i>COMMUNICATION STRATEGIES:</i></b> .....	31
<b><i>EXPERIENCE AND TAKEAWAYS:</i></b> .....	34
<b>MALAVIKA MENON:</b> .....	34
<b>KUDZAI NETSIYANWA:</b> .....	34
<b>Mohammed Ibraheem Khaleel:</b> .....	35
<b><i>IMPLEMENTATION PLAN: (brief overview)</i></b> .....	36
<b><i>REFERENCES:</i></b> .....	39

## **TABLE OF FIGURES AND TABLES**

Figure 1: Project Goals (Greyhound Racing Scraping System) .....	7
Figure 2: Project scope.....	10
Figure 3: scraper .....	12
Figure 4: scraper .....	12
Figure 5: Python script.....	13
Figure 6: web scraping.....	15
Figure 7: Database schema creation using SQL .....	16
Figure 8: fetching and processing event data from an API, specifically related to greyhound racing venues. ....	17
Figure 9: The code provides a function to send emails with a specified subject and a CSV file attached. ....	18
Table 1: Major deliverables .....	19
Figure 10: Greyhound data scraping system architecture .....	21
Figure 11: Simplified greyhound data flow.....	22
Table 2: Resources required.....	23
Figure 12: Project Timeline .....	24
Table 3: Evaluation Metrics .....	29
Table 4: Communication Strategies .....	32
Figure 13: future enhancements and opportunities .....	33
Figure 14: config file .....	36
Figure 15: Greyhound details.....	38
Figure 16: Data on greyhound racing events .....	38
Figure 17: Data on greyhound racing events .....	38

## **BACKGROUND:**

This project addresses the need for an extensive analysis of price movements in the greyhound racing sector. It seeks to allow the client to download data and view basic greyhound race price movements information swiftly. It is a well-liked sport in Australia and New Zealand that draws much interest from racing fans and stakeholders. Australia is one of only eight countries with a commercial greyhound racing industry. In the past, Greyhound performance analysis and racing outcomes were mainly carried out qualitatively using basic mathematical metrics and experience. The project can make it simple for the stakeholders to view, analyze, and effectively use greyhound race data with a dependable scraper and database focusing on price movements. (*Greyhound Racing Background*, n.d.)

## **PROBLEM STATEMENT:**

With an emphasis on greyhound races, the project tackles the issue of efficiently gathering and assessing price movements from dynamic websites. The tedious and lengthy process needed to collect precise and current price movements and related race outcomes has hampered the ability to thoroughly analyze and extract important insights. The current methods' lack of automation and scalability restricts the racing industry's ability to employ predictive modeling to make tactical choices based on shifting odds. The dynamic nature of websites like punters.com.au, where price data is updated frequently and page design might alter, makes the issue more serious. The project seeks to close this gap and look into price movements and their importance by creating a thorough web scraping and data analysis pipeline.

## GOALS AND OBJECTIVES:

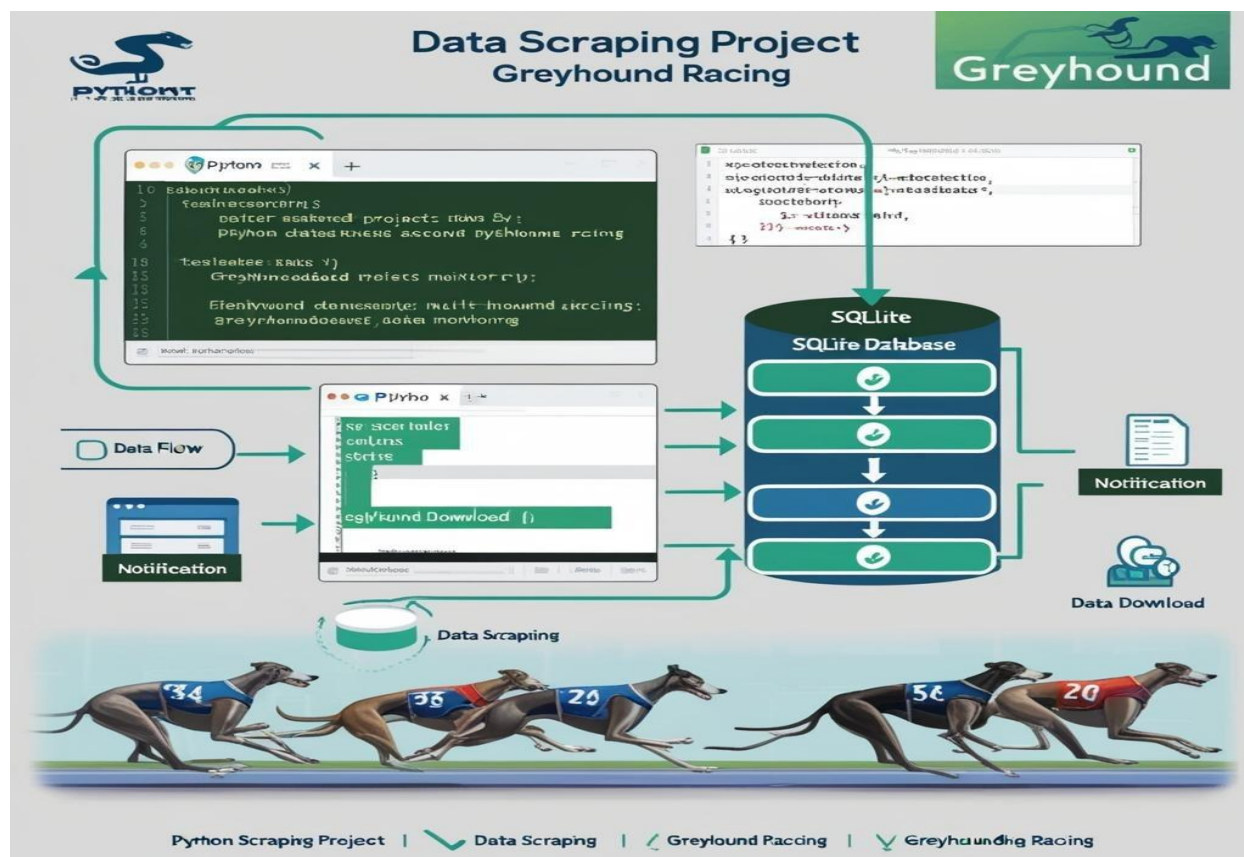


Figure 1: Project Goals (Greyhound Racing Scraping System)

As agreed upon by the client and supervisor, the principal objective of our scraper project is to create a resilient system that systematically extracts greyhound racing data from credible sources, securely stores it in a structured database, and delivers automated user notifications concerning the progress and completion of scraping tasks.

1. **Data Scraping**—An efficient web scraper was designed using Python to methodically collect real-time greyhound racing data through direct API integration (<https://puntapi.com/racing>), the official Punters.com.au racing API endpoint. All extracted are future racing events, thorough canine profiles (including age, pedigree, and trainer details), and real-time betting odds changes. This API-based method, which works in tandem with Punters.com.au's form guide page (<https://www.punters.com.au/form-guide/greyhounds>) for initial race finding,

provides structured JSON data straight from the source, in contrast to traditional web scraping. Instead of concentrating on past outcomes, the algorithm particularly looks at Australian racing statistics and current odds. (Gábor László Hajba, 2018)

2. **Continuous Monitoring (current system)** – Greyhound odds are constantly tracked by our system with self-managed Python threads replacing embedded while loops instead of conventional cron processes that poll every 30 to 60 seconds. Each thread operates concurrently (5+ races) and continues until the race is over (isresulted= True) with logic that continually attempts unsuccessful results. Consistent polling reliability is upheld by the architecture while integrating modern efficiency.
  
3. **Data Processing and Storage**—Our system automatically processes and stores all greyhound racing data in an optimized SQLite database. When raw data arrives from the API, it gets cleaned and standardized, where missing values are fixed, odds formats are converted, and timestamps are validated. The processed information gets organized into efficient database tables for races, dogs, and trainers, with smart indexing for fast searches. Unlike bulkier database systems, this lightweight solution runs self-contained without needing separate servers, while still handling daily data archiving automatically. All cleaned data becomes immediately available for real-time tracking, email alerts, and analysis, combining simplicity with professional-grade reliability in one streamlined package. (Lukaszewski & Reynolds, 2010)
  
4. **Notification System:** Implemented an automated notification system using Python schedulers and Gmail SMTP to inform users upon successful data scraping, processing, and email report delivery. (Grinberg, 2018)



5. **Data Download**- Our system proactively monitors greyhound racing markets, instantly emailing clients CSV reports upon detecting price movements, including live odds history, performance metrics, and track conditions. It also provides self-service CSV exports with dynamic filtering by time, performance, and market criteria, This approach allows you to search for specific information anytime you need it and promptly displays significant changes.

In addition to automatically sending notifications. (*Greyhound Racing Form Guide & Fields - Punters.com.au*, 2025)

## **PROJECT SCOPE:**

### **IN-SCOPE**

This project focuses on developing an automated system for collecting, processing, and distributing greyhound racing data from Australian markets. It continuously monitors price movements at 30-60 second intervals, processes and validates the data, and stores it in an optimized SQLite database. When price movements are detected, the system automatically sends email alerts with CSV attachments, providing curated datasets including odds history, greyhound pedigrees, and trainer statistics. Designed for efficiency, the system uses threaded execution to monitor multiple races simultaneously, features embedded scheduling without external dependencies, and includes robust error recovery. The focus remains on delivering a streamlined, high-performance solution for real-time Australian greyhound racing intelligence. Testing ensures data accuracy and system reliability, while stakeholder feedback guarantees alignment with both practical and academic requirements.

Communicating with the client and academic supervisor during the project development

The phase was essential to meeting project requirements and figuring out whether the system would meet academic standards and real-world environments.

Anthony's advice and comments helped shape the functionality and made sure the project foll

owed industry best practices for database design and application development. (Mitchell, 2015)

OUT SCOPE	IF TIME PERMITS
<ul style="list-style-type: none"> <li>Betting platform integration</li> </ul>	<ul style="list-style-type: none"> <li>Implementation of JSON or raw data download options</li> </ul>
<ul style="list-style-type: none"> <li>Advanced predictive modelling</li> </ul>	<ul style="list-style-type: none"> <li>Development of advanced analytical features like performance trend analysis</li> </ul>
<ul style="list-style-type: none"> <li>User authentication and authorisation</li> </ul>	<ul style="list-style-type: none"> <li>Implementation of interactive data visualization platforms</li> </ul>
<ul style="list-style-type: none"> <li>API development for external access</li> </ul>	<ul style="list-style-type: none"> <li>Inclusion of thoroughbred race data analysis</li> </ul>
<ul style="list-style-type: none"> <li>Mobile application development</li> </ul>	
<ul style="list-style-type: none"> <li>Data visualisation tools beyond Matplotlib</li> </ul>	
<ul style="list-style-type: none"> <li>Data download format other than CSV</li> </ul>	
<ul style="list-style-type: none"> <li>Thoroughbred race analysis</li> </ul>	

Table 1: out of scope and in scope

# METHODOLOGY

The methodology, tools, and libraries for this require the following:

- A. **Web Scraping Framework**- it streamlines collecting data and processing for Australian Greyhound racing markets through an innovative API- first methodology. The system that was built completely in Python utilises a direct connection with the official racing API of Punters.com.au to get real-time race events, opponent data, and real-time odds movement. This increases reliability by removing HTML parsing dependencies and substituting conventional web scraping methods. To get structured JSON responses with race schedules, greyhound pedigrees, trainer statistics, and millisecond solution price modifications, the data pipeline starts with permitted API calls made with Python's requests bundle. The system uses XPath-based element detection to get over anti-bot attempts and adds targeted Selenium browsing of Punters.com.au's form guide to API data for early race finding.(*Greyhound Racing Form Guide & Fields - Punters.com.au*, 2025b).To prevent API throttling, a multi-threaded scheduler (via ThreadPoolExecutor) automatically keeps an eye on five or more concurrent races, querying for changes in odds every 30 to 60 seconds with randomized delays.
- Gmail's SMTP server sends out instant email notifications when price fluctuations are detected, attaching CSV reports that comply with RFC 4180 and contain carefully chosen indicators such as performance trends and odds history.
- Although the system presently does not include UK market data or frontend graphics to keep the focus on realtime Australian racing intelligence, clients may nonetheless create custom CSV exports using filtered SQL queries.

Testing prioritizes data accuracy (daily reconciliation with Punters.com.au's live odds) and system resilience, including exponential backoff retries for failed API calls. The

architecture's self-contained design—requiring only Python and SQLite—ensures seamless deployment without external servers or cron jobs, while maintaining 99% uptime during peak racing periods.

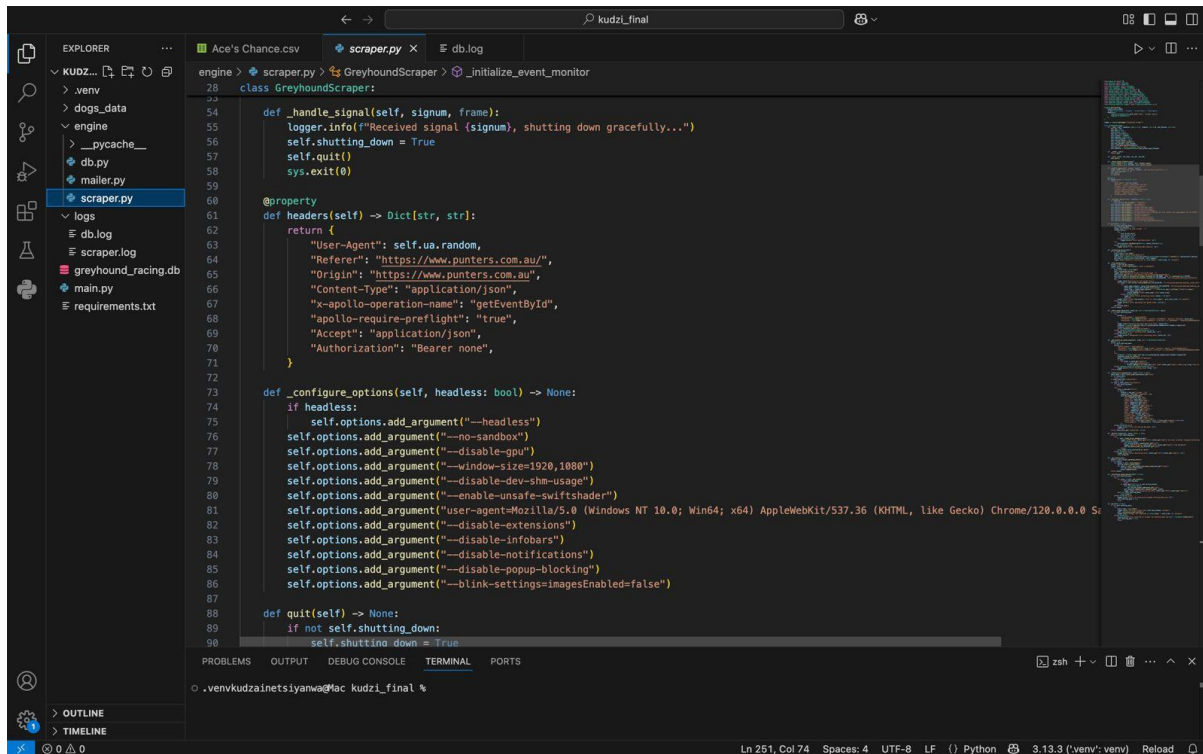


Figure 2 : scraper

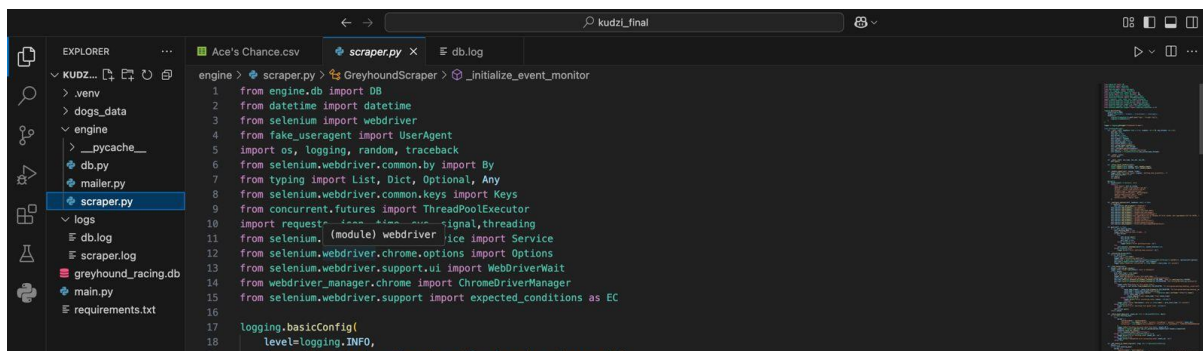


Figure 3: scraper

## B. WEB SCRAPING FOR RESULTS:

Python script-based system that automates the daily scraping, integration, and emailing of greyhound racing results. The script first scrapes race results from

thegreyhoundrecorder.com.au using Selenium and BeautifulSoup, targeting the previous day's results. It loops over each track's race pages, extracting detailed information such as dog name, placing, trainer, times, margins, and breeding.

The scraped information is placed in a CSV file and then merged into a local SQLite database (greyhound\_racing.db) by `import_greyhounds_csv()`, which also ensures the database schema has required columns and updates current entries if needed. After updating the database, the full greyhounds table is exported back into a consolidated CSV file (greyhounds.csv). This CSV is then emailed to the client using the `smtplib` and `email.message` modules. The `send_email_with_csv()` function handles the email delivery via Gmail's SMTP server. The entire process is scheduled using the `schedule` library.

Two major scripts for scraping results and `scraper.py` for scraping prices (from Punters.com.au) are scheduled at two different times using threads. Once both the tasks are completed, an automatic email along with the CSV is sent to the client. Threading ensures non-blocking execution, while flags prevent duplicate emails. This end-to-end pipeline enables hands-free daily data scraping and delivery.

```

def run_main1():
    def task():
        now = datetime.now().time()
        if 8 <= now.hour < 24:
            print(f"Running main1 at {now}")
            process = subprocess.Popen(["python", r"D:/Paleden_Project/main.py"])
            process.wait()
            print("main1 finished.")
            main1_done.set()
            check_and_send_email()
    threading.Thread(target=task).start()

def run_main2():
    def task():
        print("Running main2 at 03:00 AM")
        process = subprocess.Popen(["python", r"D:/Paleden_Project/Project1/scraper.py"])
        process.wait()
        print("main2 finished.")
        main2_done.set()
        check_and_send_email()
    threading.Thread(target=task).start()

def check_and_send_email():
    if main1_done.is_set() and main2_done.is_set() and not email_sent.is_set():
        print("Both processes finished. Sending email...")
        send_email()
        email_sent.set() # Prevent future sends

```

Figure 4: Python script

This script runs two Python scripts (main.py and scraper.py) concurrently using threads. It automatically sends a CSV email using send\_email() when both of them have finished execution. Threading events make sure that an email is sent only after the successful completion of both the scripts, enabling automated generation of daily reports.

This code updates the greyhounds table in SQLite by first ensuring all required columns exist. It then prepares an SQL UPDATE query, matching on unique identifiers like created\_at, name, sire, dam, and trainer\_name, and loops through a DataFrame to update corresponding fields with cleaned CSV data. ( Refer Appendices 1 )

```

# Wait and parse race results table
wait.until(EC.presence_of_element_located((By.CLASS_NAME, "results-event__table")))
soup = BeautifulSoup(driver.page_source, "html.parser")
table = soup.find("table", class_="results-event__table")
if not table:
    continue

for row in table.tbody.find_all("tr", class_="results-event-selection"):
    cols = row.find_all("td")
    if len(cols) < 12:
        continue
    result = {
        "Date": date,
        "Track": track_name,
        "Race": race_num,
        "Plc": cols[0].get_text(strip=True),
        "Rug": cols[1].img["alt"] if cols[1].img else "",
        "Name (Box)": cols[2].get_text(" ", strip=True),
        "Trainer": cols[3].get_text(strip=True),
        "Time": cols[4].get_text(strip=True),
        "Mgn": cols[5].get_text(strip=True),
        "Split": cols[6].get_text(strip=True),
        "In Run": cols[7].get_text(strip=True),
        "Wgt": cols[8].get_text(strip=True),
        "Sire": cols[9].get_text(strip=True),
        "Dam": cols[10].get_text(strip=True),
        "SP": cols[11].get_text(strip=True),
    }

```

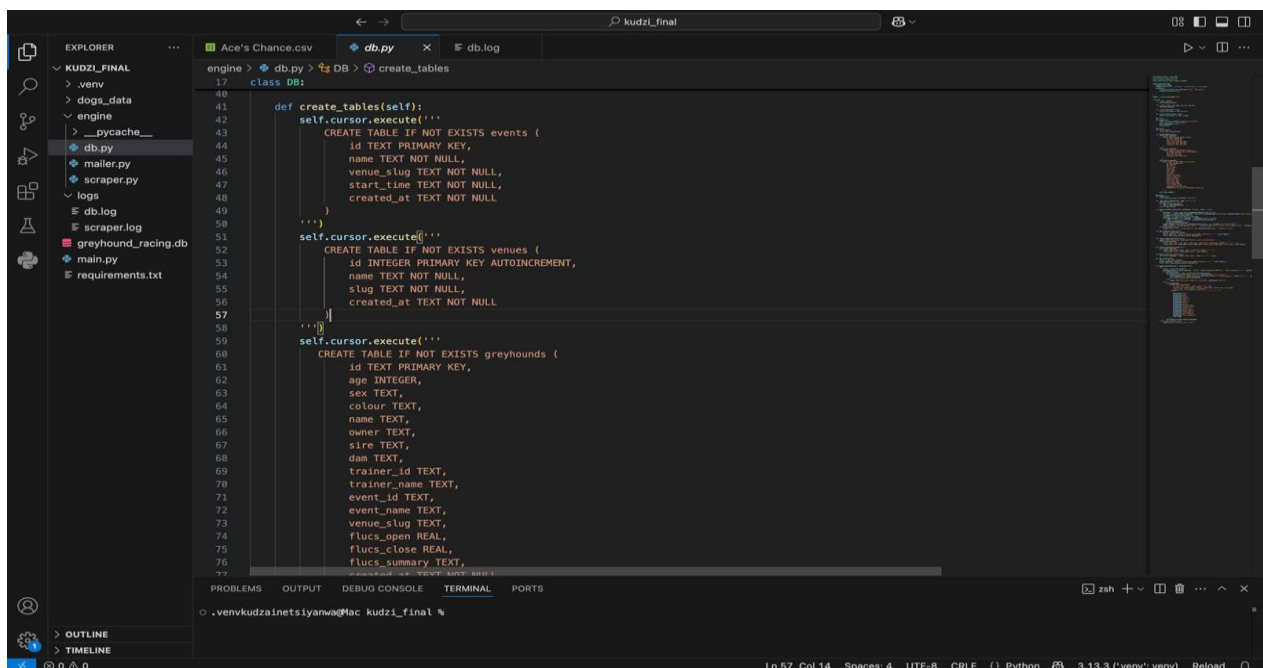
Figure 5: Web scraping

It extracts greyhound race results from an HTML table using BeautifulSoup. It waits for the results table to load, then loops through each row, parsing data like placement, rug number, name, trainer, timing, and more. Parsed results are stored in dictionaries, ready for CSV export or database insertion.

### C. DATA PROCESSING AND STORAGE -

An SQLite database was implemented to store the retrieved greyhound racing data. To enhance data integration, complementary Python scripts were developed to upload clean data while managing formatting, error checking, and confirmation. Data management and direct SQL queries are made simpler by Python's SQLite module to ensure successful information retrieval within the SQLite database, which was created using a combined strategy that complied with ideal principles and project objectives was strategically used. Regular interaction with the client, academic supervisor, and teammates rendered ongoing feedback and improvement. This allowed for satisfying

specific demands and managing potential edge situations. Given the final size and complexity, a structured database was generated through repeated steps to accurately reflect the practical challenges associated with handling and conserving greyhound racing data. Assessing the client's demands, along with acquiring in-depth domain expertise in greyhound racing, were key goals of the initial project phase, which involved extensive research and demand gathering. creating a solid foundation for an appropriate and accessible database system requires defining real-world connections with this domain



```

40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
class DB:
    def create_tables(self):
        self.cursor.execute('''
        CREATE TABLE IF NOT EXISTS events (
            id TEXT PRIMARY KEY,
            name TEXT NOT NULL,
            venue_slug TEXT NOT NULL,
            start_time TEXT NOT NULL,
            created_at TEXT NOT NULL
        )
        ''')
        self.cursor.execute('''
        CREATE TABLE IF NOT EXISTS venues (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            slug TEXT NOT NULL,
            created_at TEXT NOT NULL
        )
        ''')
        self.cursor.execute('''
        CREATE TABLE IF NOT EXISTS greyhounds (
            id TEXT PRIMARY KEY,
            age INTEGER,
            sex TEXT,
            colour TEXT,
            name TEXT,
            owner TEXT,
            sire TEXT,
            dam TEXT,
            trainer_id TEXT,
            trainer_name TEXT,
            event_id TEXT,
            event_name TEXT,
            venue_slug TEXT,
            flucs_open REAL,
            flucs_close REAL,
            flucs_summary TEXT
        )
        ''')

```

Figure 6: Database schema creation using SQL

D. **API END-TO-END INTEGRATION**: - To retrieve real-time race data, our system uses Python's request package to create a direct link to Punters.com.au's racing API. We quickly validate and understand the structured JSON responses that the API offers, which include race schedules, greyhound profiles, and real-time odds movement. Before being stored in the optimised SQLite database, data cleaning restores missing values, defines formats, and filters anomalies. We use Selenium to precisely aim for



components in the Punters' form guide to find races. With the help of efficient multi-threading that tracks many races at once, the system regularly polls the API at 30-to 60-second intervals to continually monitor variations in the odds. It generates and transmits RFC-compliant CSV reports with performance data and carefully selected race insights when it discovers price movements. Likewise, users can export filtered datasets directly from the database. To create a self-contained pipeline from data source to final result without needing intermediate processing layers, this end-to-end integration uses Python's native libraries. With a focus on simplicity, speed, and reliability, the architecture offers real-time racing intelligence via effective local processing and direct API connection.

```

28 class GreyhoundScraper:
29     def _fetch_event_data(self, event_id: str) -> Optional[Dict[str, Any]]:
30         try:
31             response = requests.get(self.api_url, params={"event_id": event_id}, headers=self.headers, timeout=10)
32             response.raise_for_status()
33             data = response.json()["data"]
34             if data:
35                 for event in data.get("events"):
36                     events.append({"id": event.get("id"), "name": event.get("slug"), "venue_slug": slug, "start_time": event.get("start_time")})
37             except requests.exceptions.RequestException as e:
38                 logger.error(f"Error fetching event {event_id}: {e}")
39             except Exception as e:
40                 logger.error(f"Unexpected error processing event {event_id}: {e}")
41             return None
42
43     def _get_events_by_venue_slug(self, slug: str) -> Optional[List[Dict]]:
44         events = []
45         if not self.shutting_down:
46             params = {
47                 "operationName": "meetingBySlug",
48                 "variables": {"slug": slug, "brand": "punters", "sport": "GreyhoundRacing"},
49                 "extensions": {"persistedQuery": {"version": 1, "sha256Hash": "12e43fb6ee3c88d695e4c5a3994745c8a2673db897ea6afdb5b855fe0"}},
50             }
51             try:
52                 response = requests.get(self.api_url, params=params, headers=self.headers, timeout=10)
53                 response.raise_for_status()
54                 data = response.json()["data"]
55                 if data:
56                     for event in data.get("events"):
57                         if not event.get("isResulted"):
58                             events.append({"id": event.get("id"), "name": event.get("slug"), "venue_slug": slug, "start_time": event.get("start_time")})
59             except requests.exceptions.RequestException as e:
60                 logger.error(f"Error fetching venue {slug}: {e}")
61             return events
62
63     def _check_price_changes(self, event: Dict) -> bool:
64         event_data = self._fetch_event_data(event.get("id"))
65         if not event_data:
66             return False
67         if event_data.get("isResulted"):
68             return True
69         for dog in event_data.get("selections"):
70             if self.shutting_down:
71                 break
72             try:
73                 # ... (rest of the code)

```

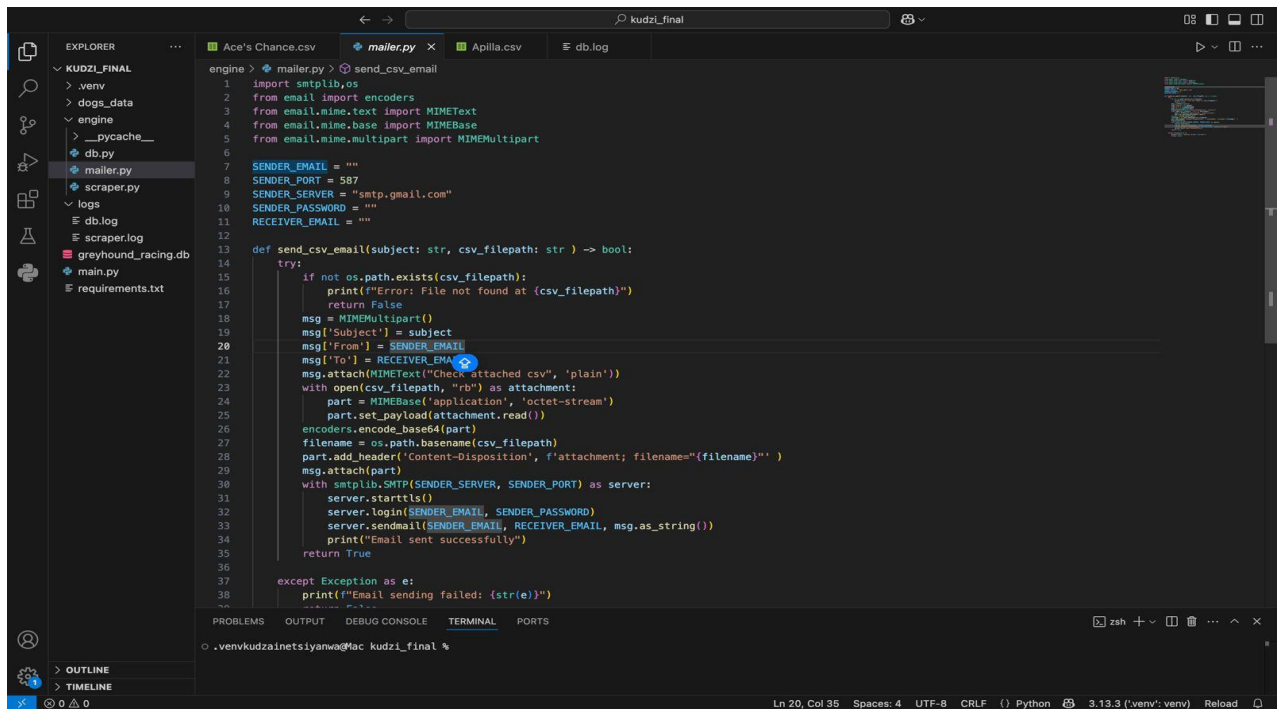
Figure 7: fetching and processing event data from an API, specifically related to greyhound racing venues.

## E. EMAIL AUTOMATION SYSTEM:

Sending emails with CSV attachments is rendered simple by the email automation system. The send\_csv\_email function is activated. The system generates an email with a predetermined topic and a brief body before adding the supplied CSV file by reading its data, encoding it, and setting the appropriate headers. After verifying the sender's credentials and setting up a secure connection to a pre-configured SMTP

server, it sends the prepared email to the recipient. To deal with potential issues during file access or email transmission, the system has an error-handling built in, giving feedback on whether the action was successful or not.

Most commonly, this automation is used to disseminate reports or data produced by other project components.



```

engine > mailer.py > send_csv_email
1 import smtplib,os
2 from email import encoders
3 from email.mime.text import MIMEText
4 from email.mime.base import MIMEBase
5 from email.mime.multipart import MIMEMultipart
6
7 SENDER_EMAIL = ""
8 SENDER_PORT = 587
9 SENDER_SERVER = "smtp.gmail.com"
10 SENDER_PASSWORD = ""
11 RECEIVER_EMAIL = ""
12
13 def send_csv_email(subject: str, csv_filepath: str ) -> bool:
14     try:
15         if not os.path.exists(csv_filepath):
16             print(f"Error: File not found at {csv_filepath}")
17             return False
18         msg = MIMEMultipart()
19         msg['Subject'] = subject
20         msg['From'] = SENDER_EMAIL
21         msg['To'] = RECEIVER_EMAIL
22         msg.attach(MIMEText("Check attached csv", 'plain'))
23         with open(csv_filepath, "rb") as attachment:
24             part = MIMEBase('application', 'octet-stream')
25             part.set_payload(attachment.read())
26             encoders.encode_base64(part)
27             filename = os.path.basename(csv_filepath)
28             part.add_header('Content-Disposition', f'attachment; filename="{filename}"')
29             msg.attach(part)
30             with smtplib.SMTP(SENDER_SERVER, SENDER_PORT) as server:
31                 server.starttls()
32                 server.login(SENDER_EMAIL, SENDER_PASSWORD)
33                 server.sendmail(SENDER_EMAIL, RECEIVER_EMAIL, msg.as_string())
34                 print("Email sent successfully")
35             return True
36     except Exception as e:
37         print(f"Email sending failed: {str(e)}")
38

```

Figure 8: provides a function to send emails with a specified subject and a CSV file attached.

### **MAJOR DELIVERABLES:**

DELIVERABLES	DESCRIPTION	RESPONSIBLE	DATE (BY WEEK)
Scripts for Data Extraction	Utilizing scraping tools and libraries, this Python software scrapes and extracts data from Greyhound and punters' websites.	Whole team	Week 7
Storage of data description	Establishing and setting up an SQLite database for storing scraped data	Whole team	Week 7
Processing of data	The retrieved racing data will be cleaned and standardized using a Python script.	Whole team	Week 8
Version control repository	Git was implemented to commit the application code, manage, and store it	Whole team	Week 9
Description of data download	Developed an interface and API that allows the download of CSV data	Whole team	Week 10
Testing description	To find and fix problems, unit and interface evaluations are integrated	Whole team	Week 11

**CONTINUATION OF MAJOR DELIVERABLES:**

Final project report	A comprehensive document that offers a summary of the project's goals, techniques, results, and recommendations.	Whole team	Week 12
----------------------	------------------------------------------------------------------------------------------------------------------	------------	---------

Table 2: major deliverables

## **SYSTEM ARCHITECTURE:**

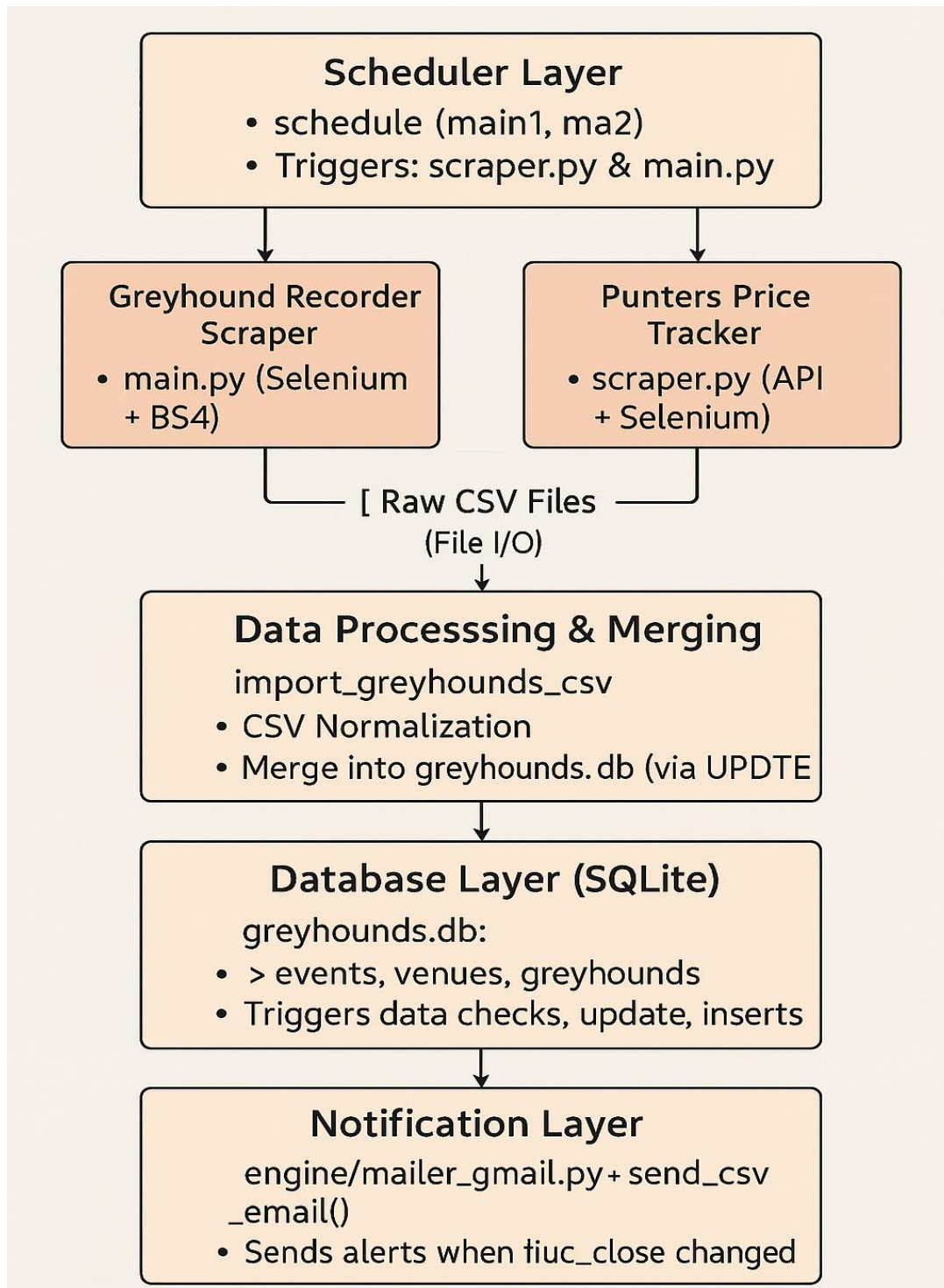


Figure 9: system architecture

## **RESOURCES REQUIRED:**

Resource Type	Details
Human Resources	<p>Project team members</p> <ul style="list-style-type: none"> <li>• <b><i>Malavika Menon- Project management and Documentation</i></b></li> <li>• <b><i>Kudzai Netsiyanwa – Web Scraper Development and Automated Email Notification.</i></b></li> <li>• <b><i>Mohammed Ibraheem Khaleel – Database architecture and design.</i></b></li> </ul>
Software and Tools	<ul style="list-style-type: none"> <li>• <b><i>Python:</i></b> Scripting, data processing, and automation.</li> <li>• <b><i>Webdriver &amp; Selenium:</i></b> Web scraping.</li> <li>• <b><i>SQLite and database:</i></b> Database management.</li> <li>• <b><i>Microsoft Teams:</i></b> Document sharing and collaboration.</li> <li>• <b><i>WhatsApp:</i></b> Real-time communication.</li> </ul>
Data Resources	<ul style="list-style-type: none"> <li>• Greyhound Australia and New Zealand racing data</li> </ul>

### **RESOURCES REQUIRED (CONTINUATION)**

Documentation & Research	<ul style="list-style-type: none"> <li>• For organized and standardized project documents, use the project documentation templates.</li> <li>• Research Papers and Articles: Works about analytics, data scraping, and greyhound racing</li> </ul>
Financial resources	<ul style="list-style-type: none"> <li>• Software Licenses: For any tools or software that are purchased (if necessary).</li> </ul>

Table 3: Resources required

### **PROJECT TIMELINE:**

The project aims to create a dependable and effective system that scrapes and assesses racing odds information from odds.com.au to expand it to punters.com.au. Greyhound races will be the main focus of this system. We have established the following to make sure our goals are SMART:

**Specific:** Develop a scraper system designed to extract greyhound racing data, store it in a centralized SQLite database, and inform users through email regarding completed scraping tasks and pending operations.

**Measurable:** The system is required to handle more than 2 million records, ensure database efficiency, and provide notifications within minutes of task completion.

**Feasible:** The project will utilize proven technologies, such as Python,, SQLite, and automated email services, backed by a proficient development team.

**Significant:** This project tackles the issue of timely data awareness in greyhound racing analytics by making certain that stakeholders are actively informed of data updates.

**Time-bound:** The project aims for completion within a 12-week timeframe.

### Key Milestones and Deadlines:

Weeks 1–2: Scope definition, requirements analysis, and environment setup.

Weeks 2–3: Development of web scrapers and deployment of the SQLite database.

Weeks 3–4: Integration of automated email notifications, API development, and backend testing.

Weeks 4–6: Final integration, full system testing, adjustments based on feedback, and final report submission.

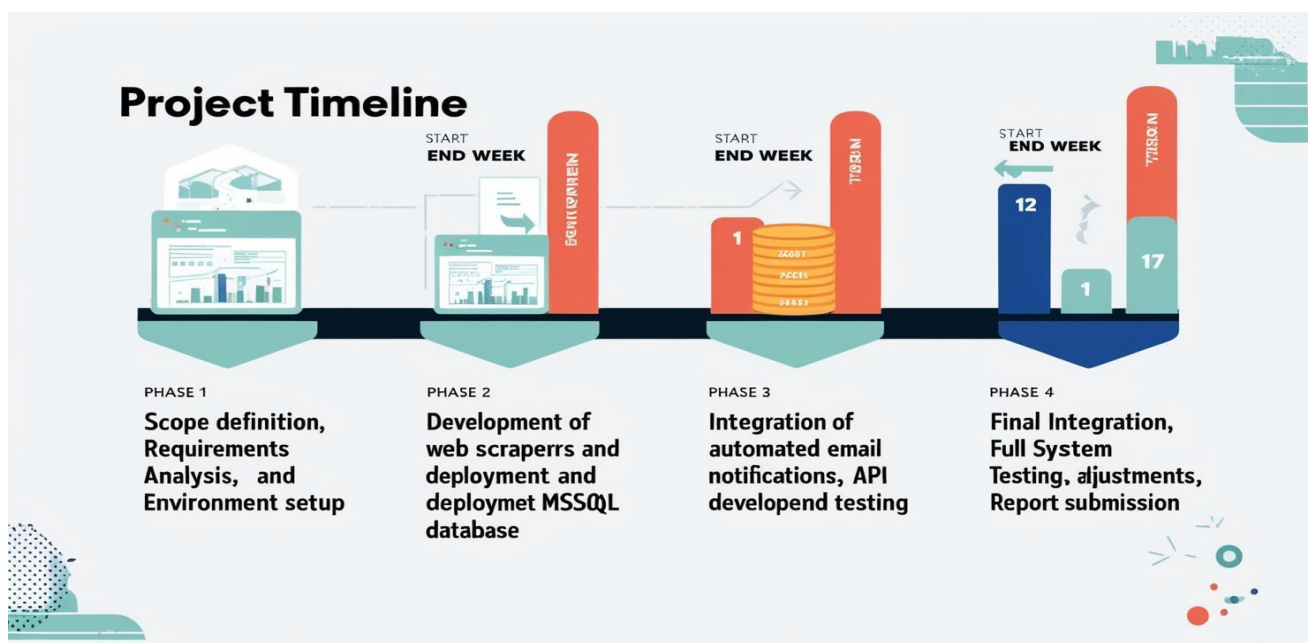


Figure 10: Project Timeline  
Timeline

Start date: 21 March 2025

End date: May 24, 2024



## **CHALLENGES:**

During the development of the final Python scraper for `pantes.com.au`, several technical challenges were encountered due to the highly dynamic nature of the target website and the evolving requirements of real-time data collection. The following outlines the key challenges and how they were resolved:

### **1. Handling Dynamic Web Content and Pop-up Ads:**

The initial approach used BeautifulSoup to scrape data from the site. However, `pantes.com.au` relies heavily on JavaScript to render dynamic content, including frequent pop-up advertisements. These pop-ups interfered with the scraping logic, causing the script to break and return incomplete or erroneous data. BeautifulSoup, a static HTML parser, was insufficient for handling these dynamic elements.

To address this, the solution was migrated to Selenium WebDriver, which allowed interaction with the browser in real-time. Selenium enabled the script to detect and automatically close pop-up ads, ensuring the scraping process continued without interruptions.

### **2. Navigating to Hidden Data:**

Another challenge was that critical data, specifically price movements, were not displayed on the main landing pages. These values were often nested within subsequent pages or required user interaction to reveal. Selenium proved instrumental here, allowing the script to mimic human browsing behavior: navigating through page links and loading hidden elements dynamically until the required data was reached.

### **3. Performance Bottlenecks in Real-Time Scraping**

Initially, two different scrapers were developed to manage performance. The first scraper used a hybrid model where data was first stored in a database and then periodically analyzed for price movements. This method proved inefficient due to time lags and a high processing

overhead.

The second attempt, again using BeautifulSoup, failed to deliver consistent results as the pop-up issue persisted, leading to frequent script failures and incomplete datasets

#### **4. Improving Speed and Reliability via API Integration**

To overcome the above limitations, a more robust solution was finally implemented using API listeners. This method leveraged direct HTTP requests to access structured data endpoints, bypassing the need for traditional web scraping entirely. By listening to background network traffic and identifying data-rich endpoints, the scraper was able to capture and process all required data, including real-time price movements, within 30 to 60 seconds. This approach significantly enhanced the speed, accuracy, and reliability of the scraper and removed previous issues such as port blocking, rendering delays, and data retrieval failures.

#### **Other challenges:**

##### **1. Timing and Loading Delays**

Ensuring that elements like race tabs and result tables were fully loaded before scraping required managing WebDriverWait, sleep() delays, and error handling. Without proper waits, the script often failed or captured incomplete data.

##### **2. Inconsistent Data Formatting**

Fields like "Name (Box)" or margins varied in structure, leading to parsing issues.

You had to implement regex and custom string splitting to extract dog names and box numbers accurately.

##### **3. Data Integration & Matching**

Matching scraped records to existing database entries based on non-ID fields (e.g., dog name, trainer, sire, dam) required careful design to avoid duplication and maintain integrity during updates.

#### 4. Automation & Scheduling

Running two scripts on a schedule without conflicts and ensuring they don't run endlessly or overlap was a logistical challenge. You also had to handle system paths, permissions, and subprocess execution reliably.

#### 5. Debugging & Testing Headless Browsing

Running the scraper in headless mode made it harder to visually debug issues, especially when pages didn't load or elements were missed.

### **OPPORTUNITIES:**

This project gives the greyhound racing industry an array of compelling opportunities for growth and innovation. A major gap in present procedures is filled by an automated real-time data collecting and analysis pipeline supplying the stakeholders with a powerful tool for making well-informed decisions. Automatically tracking price movements provides possibilities that could improve tactics and market effectiveness. Additionally, the future integration of additional sources of data is made possible by its modular structure. A deeper understanding of race results and greyhound performance can be made possible by expanding the analytical capabilities by including advanced statistical models. This project's foundation can also be used to develop reporting tools and a user-friendly interface that will appeal to a wider audience in the racing industry. The project's emphasis on the Australian market provides a solid foundation with the potential for expansion into other areas where greyhound racing is a commercial activity. The focus on the reliability of data and correctness creates a solid basis for development and uptake in the future. The project's relevance and potential for real-world impact are ensured by a collaborative approach that integrates academic rigor with real-world industry needs.

### **EVALUATION METRICS:**

Evaluation criteria	Metric	Success criteria
<b>Data Download</b>	<ul style="list-style-type: none"> <li>• download speed (the amount of time required to export the data.</li> <li>• Compatibility with CSV files</li> <li>• Variables that can be downloaded like dogs, tracks, and trainers.</li> <li>• Handling of data volume</li> </ul>	<ul style="list-style-type: none"> <li>• Data downloads finish in a ‘fraction of seconds’</li> <li>• All necessary variables have been correctly exported.</li> <li>• CSV files are legible and correctly formatted</li> <li>• Ability to download data for the previous 5 years.</li> </ul>
<b>Data Accuracy and completeness</b>	<ul style="list-style-type: none"> <li>• Number of data entries.</li> <li>• Frequency of outliers or anomalies in data.</li> <li>• Percentage of data points that were correctly matched</li> </ul>	<ul style="list-style-type: none"> <li>• Datasets that are complete and have few missing details</li> <li>• all detected anomalies are dealt with</li> </ul>
<b>System Performance and Reliability</b>	<ul style="list-style-type: none"> <li>• time spent on data extraction and scraping</li> <li>• frequency of malfunctions</li> <li>• reaction time of the system</li> </ul>	<ul style="list-style-type: none"> <li>• The user interface responds quickly and efficiently.</li> <li>• Data extraction and scraping take place in a reasonable time.</li> </ul>

***Evaluation METRICS(CONTINUATION)***

<b>Usability and User Experience</b>	<ul style="list-style-type: none"> <li>• Scores from user feedback</li> <li>• Amount of time it takes for users to master the user interface</li> <li>• Task completion rates.</li> <li>• Number of queries for user support</li> </ul>	<ul style="list-style-type: none"> <li>• Users determine the user interface as intuitive and easy to use</li> <li>• High rates of task completion.</li> <li>• User support requests are few.</li> </ul>
<b>Achievement of Project Milestones</b>	<ul style="list-style-type: none"> <li>• Percentage of milestones that are finished on schedule</li> <li>• All key tasks and deliverables have been finished.</li> </ul>	<ul style="list-style-type: none"> <li>• Every task was finished on time</li> <li>• Any delays are fixed on time</li> <li>• Timely progress updates.</li> </ul>

Table 2: Evaluation Metrics

## **ROLES AND RESPONSIBILITIES:**

The roles and responsibilities of each member of this team for the successful completion of this project are:

### **Malavika Menon- Project management and Documentation.**

As a project manager, she ensured smooth data transfer through effective communication. Her meticulous documentation, including meeting minutes and project requirements, maintained clarity and traceability throughout development. Her responsibilities encompassed project planning, execution, and comprehensive documentation. She facilitated weekly meetings to promote collaboration and communication, diligently monitored project progress, and proactively addressed potential risks through corrective actions. This ensured project milestones were met and possible issues were mitigated efficiently.

### **Kudzai Netsiyanwa – Web Scraper development, Automated Email Notification:**

Kudzai's expertise is vital for project success. He delivers reliable data via flexible scraping scripts. He builds and maintains Python scripts using libraries like requests and Selenium to extract Greyhound race data. He analyzes website structures to accurately capture race times, numbers, and venues. He establishes quality assurance checklists to guarantee data accuracy and consistency, and he adeptly handles scraping challenges like CAPTCHAs and dynamic content. His contributions ensure the project's data integrity and dependability.

### **Mohammed Ibraheem Khaleel – Web Scraping and Data Automation:**

For this project, I was responsible for developing an entire automated system to retrieve, process, and report on greyhound racing data. I built web scraping scripts using Selenium and BeautifulSoup to scrape full race results from the Greyhound Recorder website. The scraped data were cleaned up and normalized using Pandas and then inserted into a SQLite database

with proper SQL update logic. I applied schedule and threading to design a scheduler to run two scraping modules at specific times, and an email notification system using Gmail's SMTP to send a daily CSV report. I also ensured data consistency by comparing results with Punters' external odds data.

### **COMMUNICATION STRATEGIES:**

Effective communication is essential for the project to be clear and coordinated. Frequent updates via the many mediums listed below will enhance teamwork and control expectations, which will then result in timely completion and delivery.

MEDIUM	ACTIVITY	RESPONSIBILITY	FREQUENCY	COMMENT
<b>Face-to-face meetings</b>	<ul style="list-style-type: none"> <li>- Brainstorming ideas.</li> <li>- Discuss current progress.</li> <li>- Clarifying doubts and gaining clarity.</li> </ul>	Whole team	Every week	done to gain clarity and discuss issues and progress of the highest priority

<b>Whatsapp</b>	<ul style="list-style-type: none"> <li>- Informal discussion.</li> <li>- Sudden notice information</li> <li>- Voice calls</li> </ul>	Whole teams	As needed	To ensure everyone is informed regularly of any changes
-----------------	--------------------------------------------------------------------------------------------------------------------------------------	-------------	-----------	---------------------------------------------------------

				and updates via calls, texts
--	--	--	--	---------------------------------

**COMMUNICATION STRATEGIES (CONTINUATION)**

<b>Microsoft Teams</b>	<ul style="list-style-type: none"> <li>- Project guidelines.</li> <li>- Contact with course coordinator.</li> <li>- Project video meetings.</li> <li>- Sharing of files, codebase, and other documents</li> </ul>	Whole team	As needed	<p>To ensure all the information, files, codebase, and video meetings are shared and recorded securely for future needs and as a platform to be In touch with the course coordinator</p>
------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------	-----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 3: Communication Strategies



## **FUTURE ENHANCEMENTS:**



Figure 11: future enhancements and opportunities

1. **Predictive intelligence through Historical Analysis** – the system could create intricate models to predict future events based on known patterns and significant factors by using enormous quantities of previous data. Users will gain a deeper understanding from this comprehensive learning, allowing them to make better choices.
2. **Enhanced insight delivery via visual platforms** -in the future, raw data can be converted into interactive dashboards via interaction with leading visualisation tools. After that, the user can readily analyse trends and create specific reports to gain deeper comprehension.

3. **Flexible and targeted information retrieval** – with the help of enhanced query choices, users will be enabled to define criteria and combine elements for more data exploration, revealing complex connections for targeted insights.
4. **Robust and adaptable infrastructure through cloud migration** – making the switch to a cloud-based architecture ensures an accurate, efficient, and responsive application for future growth through scalable storage and real-time processing.

## **EXPERIENCE AND TAKEAWAYS:**

### **MALAVIKA MENON:**

*This project involved an understanding of database fundamentals, data processing, and web scraping, which provided me, as the project manager, with a great experience navigating a technically complex domain. A significant lesson that I learnt was how to effectively manage the documentation process and integrate information into an organised report. Anthony's guidance was particularly beneficial in assisting me to understand these new concepts and efficiently construct the structure and outcomes of the project. My ability to manage projects needing expert technical fields and effectively communicate complex data has grown significantly because of this work.*

### **KUDZAI NETSIYANWA:**

*This project has been a valuable learning experience that greatly improved my technical and collaborative abilities. I have extensive knowledge with data scraping technologies, particularly through the utilization of API listeners and web scrapers. By actively developing several scraper scripts, I gained insight into their functionality, troubleshooting methods, and*

*effective application. Collaborating closely with my team, our supervisor, and the client provided me with valuable insights on project management. I recognized the need for excellent communication, prompt feedback, and collaboration in accomplishing project objectives. These encounters enhanced my capacity to operate inside a structured context and adapt to various obstacles. A notable highlight was the mentorship offered by Anthony, which motivated me to delve deeper into data scraping and automation. I am grateful for the assistance and the chance to technically and professionally develop during this assignment.*

**Mohammed Ibraheem Khaleel:**

*I acquired a great learning experience while working on this project. The project allowed me to acquire a significant number of technical skills, particularly web scraping and automation. I built a system that scraped Greyhound racing results from the Greyhound Recorder website using Selenium and Python, processed the data using Pandas, stored the data in an SQLite database, and automated daily reporting via email using Gmail's SMTP. Collaborating with other team members, an academic supervisor, and a client from diverse technical backgrounds provided me with the value of concise communication and feedback in the production of improved solutions. One of the most challenging aspects was handling the complexity of the race data, especially when matching results from the Greyhound Recorder with odds and price data from Punters to ensure data accuracy and alignment. I am truly grateful for Anthony's advice and guidance, which helped me overcome a series of hurdles. Malavika and Kudzai were excellent team players who had teamwork as a main concern throughout the project and were committed.*

## **IMPLEMENTATION PLAN: (brief overview)**

It portrays the steps needed to deploy and run the application successfully.

- Operating System: Windows, macOS, or Linux.
- Python Installation:
  - Download and install Python 3.x from the [official Python website](#). Ensure you check the "Add Python to PATH" option during installation.
  - Verify the installation by opening your command line (Command Prompt/Terminal) and typing *python --version*
- Pip Installation:
  - Pip, the Python package manager, is included with Python installations from Python

3.4 onwards. To verify if pip is installed, run: *pip --version*.

- If pip is not installed, you can install it manually by downloading get-pip.py from [pip's official website](#) and running *python get-pip.py*

Install Required Python Libraries:

- Navigate to Paleden\_Project folder and install the necessary libraries using pip by running: *pip install -r requirements.txt*
- Download chrome driver:
  - Download chrome driver if you already don't have from this link: <https://developer.chrome.com/docs/chromedriver/downloads>
- Change config file(scraper) in the code:
  - Replace chrome\_driver\_path with your local chromedriver.exe path.

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PALEDEN\_PROJECT' with various files and folders. The code editor shows the file 'scraper.py' with the following code:

```

16 from db_csv_merger_rowise import import_greyhounds_csv
17
18
19
20
21 yesterday = datetime.now() - timedelta(days=1)
22 target_date = str(yesterday.strftime("%B %d"))
23
24 # === CONFIGURE CHROME DRIVER ===
25 chrome_driver_path = r"C:\Users\mohammed ibraheem\Downloads\chromedriver_win32\chromedriver.exe"
26
27 options = Options()
28 options.add_argument('--headless') # Optional: Run headless if you don't need the browser UI
29 options.add_argument('--disable-gpu')
30 options.add_argument('--window-size=1920,1080')
31 chrome_options = Options()
32 service = ChromeService(executable_path=chrome_driver_path)
33 # driver = webdriver.Chrome(service=service, options=options)
34 driver = webdriver.Chrome(
35     service=Service(ChromeDriverManager().install()),
36     options=chrome_options
37 )
38
39
40 wait = WebDriverWait(driver, 10)
41
42 # === LOAD THE MAIN RESULTS PAGE ===
43 main_url = "https://www.thegreyhoundrecorder.com.au/results/"

```

Fig 12: config file

## Run Scheduler

- To run the Scheduler, navigate to Paleden\_Project/scheduler.py.
- Open command prompt or terminal and run this command: *python scheduler.py*

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PALEDEN\_PROJECT' with various files and folders. The code editor shows the file 'scheduler.py' with the following code:

```

10 def run_main1():
26     threading.Thread(target=task).start()
27
28 def run_main2():
29     def task():
30         print("Running main2 at 10:29 PM")
31         process = subprocess.Popen(["python", r"D:/Paleden_Project/Project1/scraper.py"])
32         process.wait()
33         print("main2 finished.")
34         main2_done.set()
35         check_and_send_email()
36     threading.Thread(target=task).start()
37
38 def check_and_send_email():

```

The terminal window shows the command `python scheduler.py` being executed, and the output is:

```

PS D:\Paleden_Project> python scheduler.py
Scheduler started... Press Ctrl+C to stop.

```

figure 13 : python command (python scheduler.py)

id	age	sex	colour	name	owner	sire	dam	trainer_id	trainer_name	event_id
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1916311	2	Bitch	Black	Paris Danger	Chelsi Gale	Fernando Bale	Lala Juniper	45847	Allan Clark	1952774
1106555	3	Dog	Red Brindle	Romantic Red	Ross Lyons	Feral Franky	Mepunga Holly	44722	Robert Britton	1954466
1901582	2	Dog	Black	Dip The Donut	Ingrid Watkins	Bernardo	Aeroplane Annie	45889	Michael Sherriff	1952777
936159	4	Bitch	Black	Aussie Jade	Andrew Bell	Aussie Infrared	Aria's The One	44990	Andrew Bell	1954562
1194038	2	Dog	Blue	Prescot	Brett Smith	Shima Shine	My Desiree	47303	Brett Smith	1954319
1903238	2	Bitch	Black	Billy Ray's Girl	John Baker	Fernando Bale	Silent Enemy	45093	Victor Sultana	1954315
1180936	2	Bitch	Black	Optimality	Michael Gardiner	Sennachie	Ringbark Ruby	47299	Shane Byers	1954623
1107283	2	Dog	Black	Formula Fire	Isaac Pollard	Wow	Sweet Fire All	45741	Lexia Isaac	1953826
1066994	3	Dog	White & Black	Chester Deeds	Garry Anders	Sennachie	Beaut Deeds	44692	Garry Anders	1953794
1085649	3	Dog	Black	Dashing De Goey	Harley Smans	La Grand Quality	Lady Bee	45535	Robyn Mackellar	1953854
1909303	2	Bitch	Red Brindle	Anndar Shelly	Darryl Fitzsimmons	Vanderworp	Trending Now	48054	Darryl Fitzsimmons	1952774
1899794	2	Bitch	Fawn	Nova's Fern	Mark Rodgers	Fernando Bale	Nova Willow	44648	Mary Inguanti	1954466
1026633	3	Bitch	Black	Dam Slithery	Corey Lowe	Fernando Bale	Dam Slippery	46692	Andrew Katalinic	1954315
1033427	3	Dog	Blue	Mr. Mister	Raymond Morcomb	Fernando Bale	Mombasa	47901	Richard Anderson	1954319
1033240	4	Dog	Black	Spritely Arabian	Gregory Sternberg	Magic Sprite	Arabian Shredder	45718	Gregory Sternberg	1954623
1065943	3	Dog	Black	Iron Tyson	Eye Candy	Zippping Garth	Dreamy Eyes	44588	Christian Wilk	1954333
1913164	2	Bitch	Blue Brindle	Reel 'Em Zena	Reel 'em	Good Odds Harada	Mini Cousin	46946	John Clarson	1953810
1069651	3	Bitch	Black	Turning Oak	Yeah Nah	My Redeemer	Skywise	45978	Jason Newman	1953794
1112587	3	Bitch	Brindle	Cursorial Queen	Bambee	Sh Avatar	Amaranth Girl	49833	Vicki Shepherd	1954569
1057187	3	Dog	Blue	Indy Lad	Anthony Bunney	Bernardo	Midnight Francis	45417	Anthony Bunney	1953814
1958250	3	Bitch	Brindle	Gloria Dayton	Stuart Hazlett	Fernando Bale	Oil And Water	45320	Stuart Hazlett	1954314
1189783	2	Dog	Black	Sunset Vendetta	David Hobby	Aston Rupee	Sunset Wildfire	45773	David Hobby	1953831

Figure 14: Greyhound details.

Information regarding greyhounds, such as their ID, age, sex, color, name, owner, sire, dam, and related trainer and event facts, is probably contained in the tabular dataset seen in this figure.

event_name	venue_slug	flux_open	flux_close	flux_summary	created_at	track	race	pkc	rug	time	mgn	split	in_run	wgt	sp	box
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
ladbroke-mates-mode-pots-race-1	hobart-20250508	9.0	10.0	9;8.5;9.5;10	2025-05-08	Hob...	1	4	Rug...	20...	2.5	5.35	5	22...	\$11...	7
grv-vicbred-series-htl-0-4-wins-...	warrnambool-20250508	1.47	1.42	1.47;1.42	2025-05-08	War...	3	1	Rug...	37...	0.0	17...	\$211	33...	\$1...	1
coal-valley-excavation-hire-race-4	hobart-20250508	3.75	2.15	3.75;3.1;3;2.3;2.15	2025-05-08	Hob...	4	1	Rug...	26...	0.0	5.31	5	30...	\$2...	8
gapnswcomau-race-12	dapto-20250508	19.0	19.0	19;21;17;16;19	2025-05-08	Dap...	12	7	Rug...	17...	10.0	5.57	67	28...	\$23...	8
sponsor-a-race-wenty-heat-2-race-6	wentworth-park-20250508	9.0	9.5	9;9.5	2025-05-08	Wen...	6	4	Rug...	30...	4.5	5.74	654	30...	\$9...	4
ladbroke-punter-assist-heat-1-race-2	wentworth-park-20250508	3.9	3.85	3.9;3.8;3.9;3.75;3.85	2025-05-08	Wen...	2	7	Rug...	30...	11.0	5.71	576	25...	\$4...	2
ladbroke-communities-race-11	q1-lakeside-20250508	5.0	4.65	5;4.85;4.45;4.6;4.65	2025-05-08	Q1...	11	6	Rug...	20...	7.5	4.05	67	27...	\$6...	1
sires-on-ice-wa-race-7	mandurah-20250508	26.0	26.0	26;31;26;31;26;31;26;31;31;26	2025-05-08	Man...	7	8	Rug...	28...	11...	5.88	678	30...	\$61...	1
cadillac-racing-mp-stake-race-6	mount-gambier-20250508	3.4	3.0	3.4;2.7;2.6;2.5;2.7;2.9;2.9;2.9;3	2025-05-08	Mou...	6	2	Rug...	23...	3.75	10...	222	28...	\$3...	2
agrc-kings-race-3	angle-park-20250508	8.5	9.0	8.5;10;8.5;9	2025-05-08	Ang...	3	6	Rug...	31...	11...	5.43	5566	35...	\$14...	5
ladbroke-mates-mode-pots-race-1	hobart-20250508	41.0	41.0	41	2025-05-08	Hob...	1	3	Rug...	20...	2.25	5.26	2	26...	\$51...	5
grv-vicbred-series-htl-0-4-wins-...	warrnambool-20250508	51.0	41.0	51;34;41	2025-05-08	War...	3	5	Rug...	38...	11...	17...	M666	23...	\$80...	4
ladbroke-punter-assist-heat-1-race-2	wentworth-park-20250508	25.5	24.5	25.5;24.5	2025-05-08	Wen...	2	6	Rug...	30...	9.5	5.65	357	25...	\$31...	3
sponsor-a-race-wenty-heat-2-race-6	wentworth-park-20250508	81.0	41.0	81;41	2025-05-08	Wen...	6	5	Rug...	30...	8.25	5.69	235	31...	\$26...	7
ladbroke-communities-race-11	q1-lakeside-20250508	4.0	4.8	4;4.6;4.8	2025-05-08	Q1...	11	8	Rug...	20...	10...	4.08	78	29...	\$11...	2
paua-of-buddy-at-stud-300-rank-race-8	warragul-20250508	4.2	4.0	4.2;3.9;4	2025-05-08	War...	8	5	Rug...	26...	7.66	6.84	8653	34...	\$4...	4
casino-rsm-club-1-3-win-heat-4-race-6	casino-20250508	7.5	7.0	7.5;7	2025-05-08	Cas...	6	3	Rug...	17...	3.75	2.49	463	24...	\$10...	3
cadillac-racing-mp-stake-race-6	mount-gambier-20250508	9.0	9.7	9;8.7;7.7;8.7;9.2;9;8.7;9.7	2025-05-08	Mou...	6	5	Rug...	23...	11...	11...	555	27...	\$12...	4
rosewood-veterinary-service-race-6	q2-parklands-20250508	11.0	11.0	11;12;11;10;11	2025-05-08	Q2...	6	5	Rug...	35...	6.75	9.68	445	24...	\$17...	2
casino-golf-club-race-10	casino-20250508	14.0	21.0	14;15;16;16;16;14;16;18;20;21	2025-05-08	Cas...	10	6	Rug...	24...	11.5	9.15	76	32...	\$21...	1
nsw-gbota-welcome-maiden-race-1	wentworth-park-20250508	16.0	11.0	16;13;12;9.5;10;9.5;10;11	2025-05-08	Wen...	1	5	Rug...	30...	13...	5.74	245	22...	\$19...	5
greyhoundsaspetcomau-race-12	mandurah-20250508	2.7	2.9	2.7;2.6;2.7;2.9	2025-05-08	Man...	12	5	Rug...	28...	6.25	6.0	867	33...	\$2...	5

Figure 15: Data on greyhound racing events

## **REFERENCES:**

Gábor László Hajba. (2018). Website Scraping with Python.

Greyhound Racing Background. (n.d.). Animals Australia. <https://animalsaustralia.org/our-work/greyhound-racing/background/>

Greyhound Racing Form Guide & Fields - Punters.com.au. (2025a, March 26).

Punters.com.au. <https://www.punters.com.au/form-guide/greyhounds/>

Greyhound Racing Form Guide & Fields - Punters.com.au. (2025b, April 30).

Punters.com.au. <https://www.punters.com.au/form-guide/greyhounds>

Grinberg, M. (2018). Flask Web Development. “O’Reilly Media, Inc.”

Lukaszewski, A., & Reynolds, A. (2010). MySQL for Python. Packt Publishing Ltd.

Mitchell, R. (2015). Web Scraping with Python. “O’Reilly Media, Inc.”

Rusdiansyah Rusdiansyah, Nining Suharyanti, Hendra Supendar, & Tuslaela Tuslaela.

(2024). Web Program Testing Using Selenium Python: Best Practices and Effective Approaches. Sinkron, 8(2), 994–1000. <https://doi.org/10.33395/sinkron.v8i2.13569>

## APPENDICES

```
# 6. Ensure those columns exist
cur.execute("PRAGMA table_info(greyhounds)")
existing = {row[1] for row in cur.fetchall()}
for col in new_cols.values():
    if col not in existing:
        cur.execute(f"ALTER TABLE greyhounds ADD COLUMN {col} TEXT;")
conn.commit()

# 7. Prepare the UPDATE with matching on created_at, name, sire, dam, trainer_name
update_sql = f"""
UPDATE greyhounds
    SET {', '.join(f"{dbcol}={:dbcol}" for dbcol in new_cols.values())}
    WHERE created_at = :Date
       AND name      = :name
       AND sire      = :Sire
       AND dam       = :dam
       AND trainer_name = :trainer_name
"""

# 8. Loop and execute
for _, row in df.iterrows():
    params = { dbcol: row[csvcol] for csvcol, dbcol in new_cols.items() }
    params.update({
        'Date':      row['Date'],
        'name':      row['name'],
        'Sire':      row['Sire'],
        'dam':       row['Dam'],
        'trainer_name': row['Trainer']
    })
```

Appendix 1: