

WEB SCRAPING



TAYARA.TN

TEAM

Eslém Sebri

Oumayma Abayed

Overview of the Solution

This document presents the design, system components, and data flow of our Tayara.tn vehicle listing scraper, extended with data encryption and decryption. The goal is to securely collect, store, and protect web-scraped vehicle data.

tayara



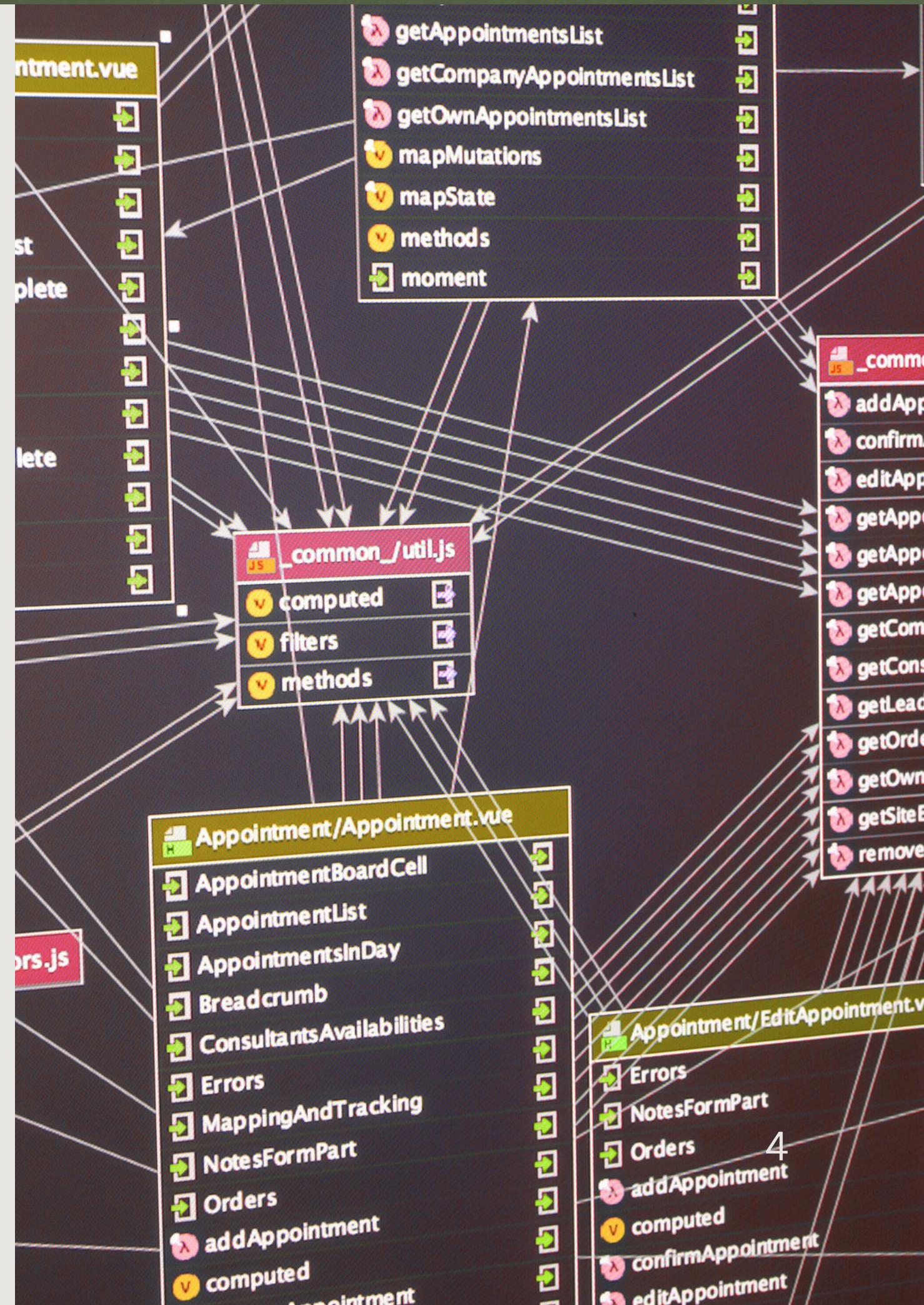
System Components & Roles

| Component | Role/Function |
|----------------------|---|
| Selenium Web Scraper | Automates interaction with Tayara.tn pages and extracts listing information |
| Parser (In-Loop) | Extracts title, price, and location from HTML elements |
| Storage (CSV Writer) | Uses Pandas to store structured data into a CSV file |
| Encryptor | Encrypts the resulting CSV using Fernet symmetric encryption |
| Decryptor | Decrypts the encrypted CSV when needed for access or processing |

Workflow & Data Flow

Step-by-Step Flow:

1. **Initialize WebDriver** with headless Chrome options.
2. **Navigate Tayara Pages** from 1 to 20 using a for loop.
3. **Extract Listings** from each page:
 - a. Extract `<h2>` tag (title)
 - b. Extract `<data>` tag (price)
 - c. Extract location from `.text-neutral-500`
 - d. Build and store full link to each listing
4. **Append Data** to a Python list.
5. **Write to CSV** using `pandas.DataFrame.to_csv`.

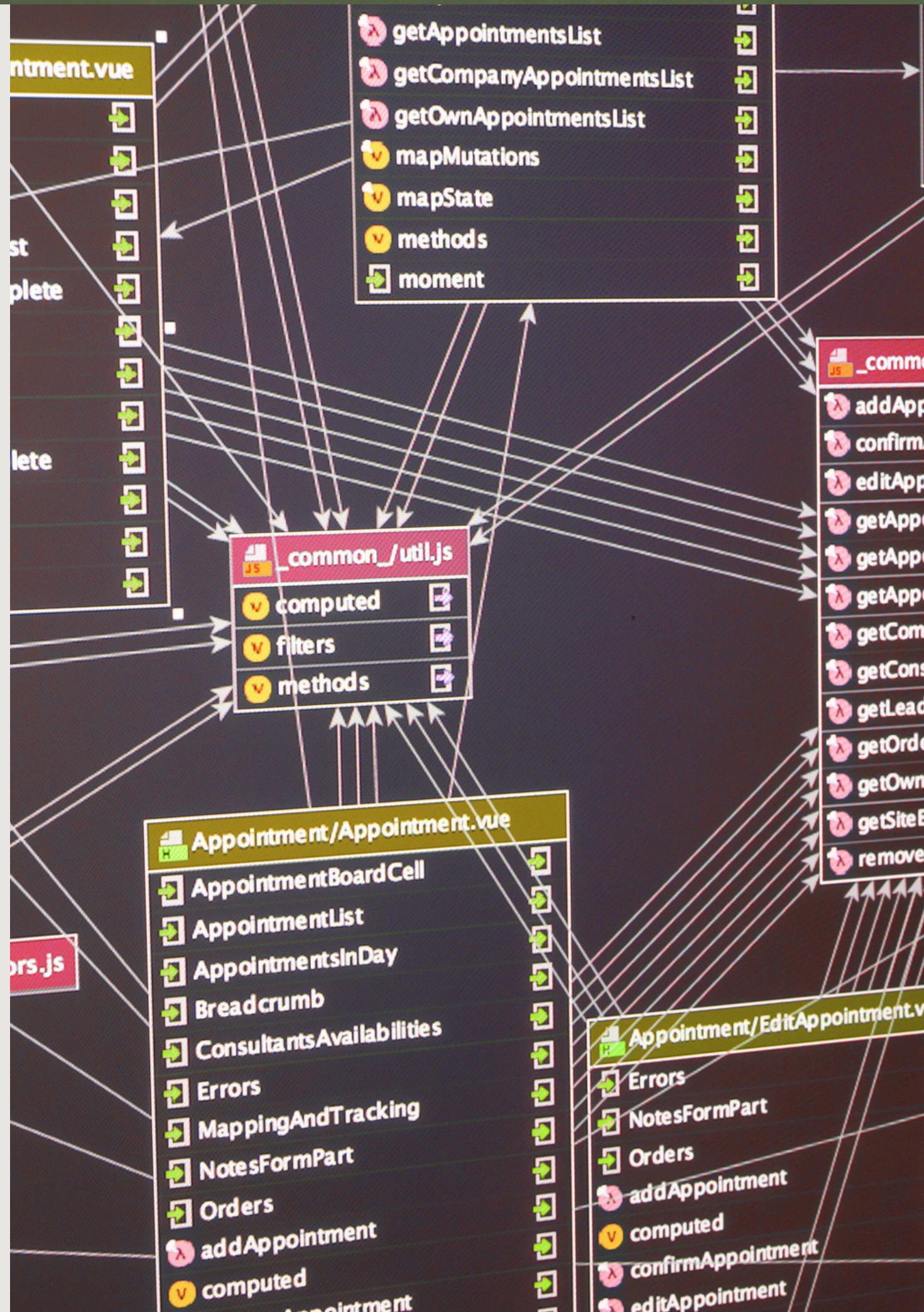


6. Encryption:

- Generate or load Fernet key
- Encrypt tayara_vehicles.csv content
- Save encrypted file as tayara_vehicles_encrypted.csv
- Save key as encryption.key.

7. Decryption:

- Load key from encryption.key.
- Read and decrypt tayara_vehicles_encrypted.csv
- Save result as tayara_vehicles_decrypted.csv



Exchanged Data

| From | To | Data | Format |
|------------|---------------|-------------------------|-------------|
| Selenium | Python Parser | HTML Element Data | DOM / Text |
| Parser | Data Store | Structured listing info | Python dict |
| Data Store | Disk | Final dataset | CSV file |
| Encryptor | Disk | Encrypted file | CSV file |
| Decryptor | CSV | Decrypted CSV data | CSV file |

Tools & Technologies

1

Python

Programming language

2

Selenium

Web automation & scraping

3

ChromeDriver

Headless browser automation

4

Pandas

Data manipulation & storage

5

CSV

Flat-file data format

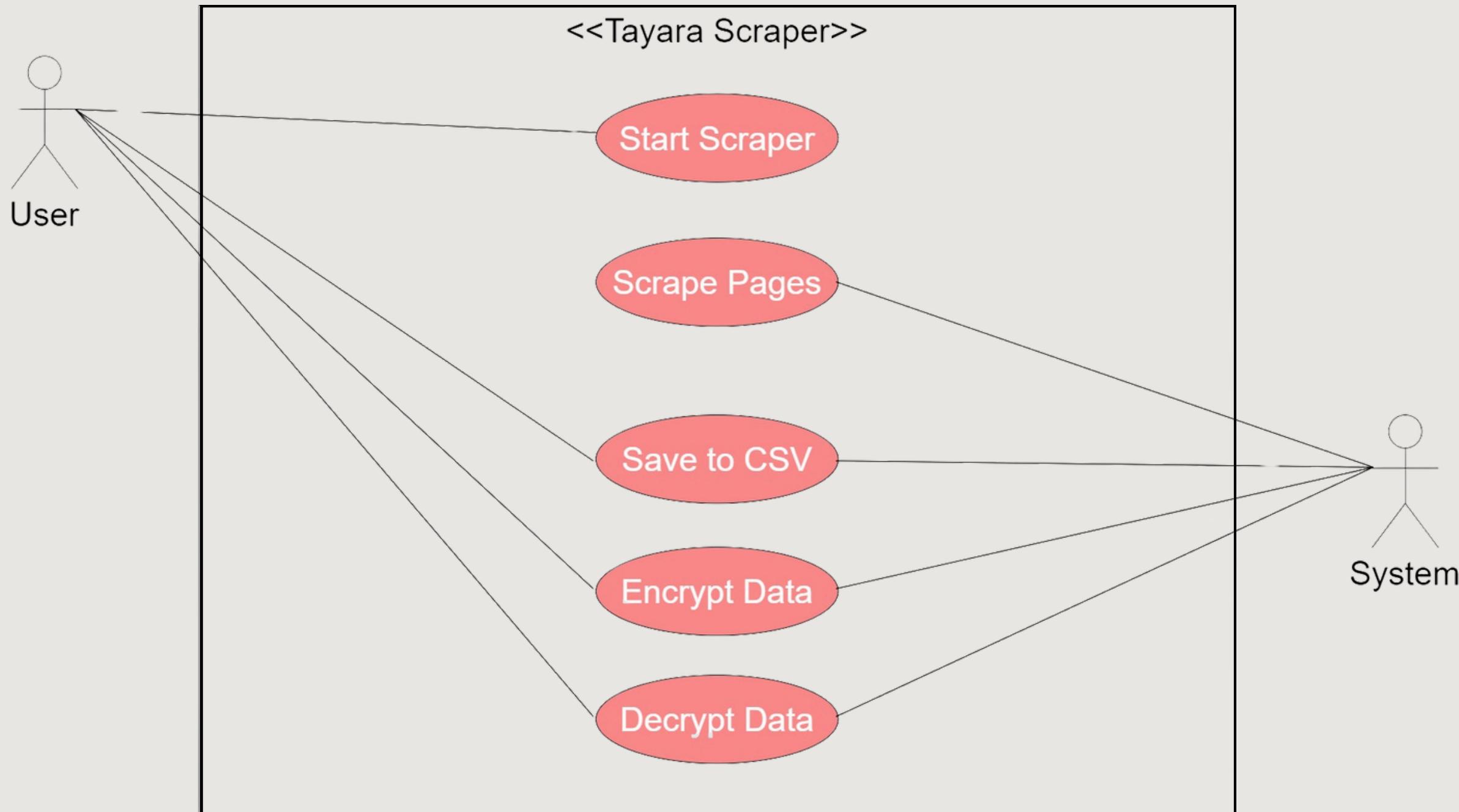
6

Cryptography

Fernet-based AES
encryption/decryption

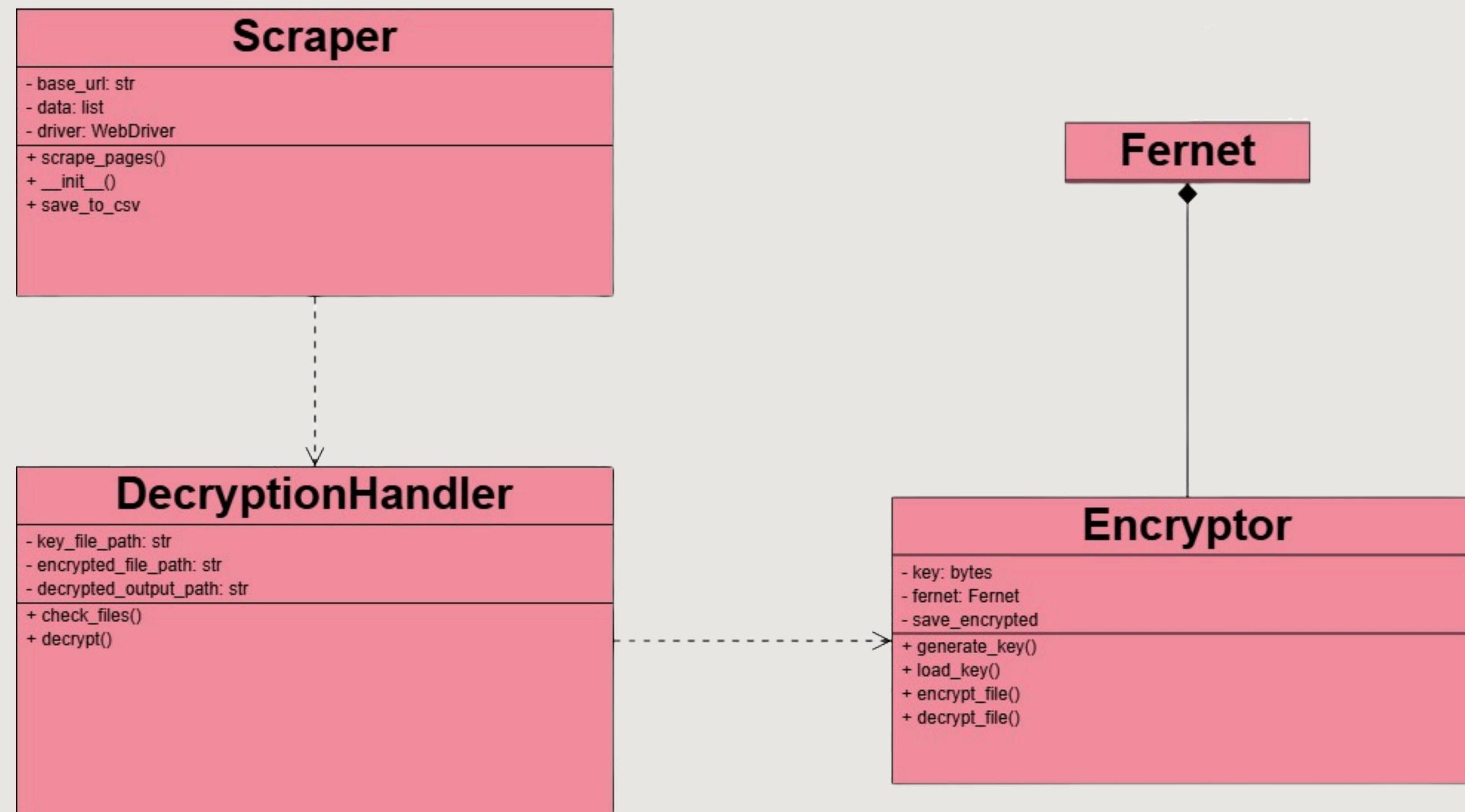
7

Actors & Use Cases Diagram



- **User (Scraper Operator):** Initiates and manages the overall process of scraping, encryption, decryption, and saving data.
- **System:** Executes the scraping, encryption, decryption, and file handling processes in the background

Class Diagram



Development Phases

Phase 1: Environment Setup

Install Python, Selenium, ChromeDriver, and Pandas

Phase 2: Build Initial Scraper

Create functional script to scrape one page

Phase 3: Expand to Pagination

Loop through multiple pages (20 pages)

Phase 4: Error Handling

Add try/except to skip malformed listings

Phase 5: CSV Export

Store clean structured data

```
4 # Prevent database truncation if the environment is not :test
5 abort("The Rails environment is running in production mode")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires below this line if necessary
22
23 # Requires supporting ruby files with custom matchers and helpers
24 # in _support/
25 # run as spec files by default. This means you will need
26 # to run twice. It is recommended that you name
27 # option on the command line after --tag
28 # mongoid
29
30 # No results found for 'mongoid'
```

Development Phases

Phase 6: Key generation

Generated and saved a Fernet encryption key.

Phase 7: CSV Encryption

Encrypted the CSV file using the saved key.

Phase 8: Decryption Logic

Decrypted the file using the same Fernet key.

Phase 9: File Handling Fixes

Resolved filename/path errors for smooth execution.

```
4 # Prevent database truncation if the environment is not :test
5 abort("The Rails environment is running in production mode")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create
14 Shoulda::Matchers.configure do |config|
15   config.integrate do |with|
16     with.test_framework :rspec
17     with.library :rails
18   end
19 end
20
21 # Add additional requires below this line if necessary
22
23 # Requires supporting ruby files with custom matchers and
24 # run as spec files by default. You can don't use this.
25 # in _spec.rb will both be required.
26 # run twice. It is recommended to always keep it in the middle of all
27 # action on the second line of this block. Also, when it
28 # option on the second line of this block.
29
30 # No results found for 'mongoid'
```

Python Code

Web Scraping

```
scraper.py 1 X
C: > Users > user > Downloads > scraper.py > ...
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3  from selenium.webdriver.chrome.service import Service
4  from selenium.webdriver.chrome.options import Options
5  import pandas as pd
6  import time
7
8  # Setup Chrome with headless option (remove headless if you want to see it run)
9  options = Options()
10 options.add_argument('--headless')
11 options.add_argument('--no-sandbox')
12 options.add_argument('--disable-dev-shm-usage')
13
14 # Path to ChromeDriver (you can modify if needed)
15 service = Service()
16
17 driver = webdriver.Chrome(service=service, options=options)
18
19 # URL base
20 base_url = 'https://www.tayara.tn/ads/c/V%C3%A9hicules/?page='
21
22 # Data container
23 data = []
24
```

```
25  # Loop over 20 pages
26  for page in range(1, 21):
27      driver.get(base_url + str(page))
28      time.sleep(2) # Let the page load
29
30  # Get all listings
31  listings = driver.find_elements(By.CSS_SELECTOR, 'a[href^="/item/"]')
32
33  for listing in listings:
34      try:
35          title = listing.find_element(By.TAG_NAME, 'h2').text
36          price = listing.find_element(By.TAG_NAME, 'data').text
37
38          # Get last span with neutral-500 (contains location and time)
39          location_time = listing.find_elements(By.CSS_SELECTOR, 'span.text-neutral-500')[-1].text
40          location = location_time.split(',') [0].strip()
41
42          # Build full link
43          partial_link = listing.get_attribute('href')
44
45          data.append({
46              'Title': title,
47              'Price': price,
48              'Location': location,
49              'Link': partial_link
50          })
51      except Exception as e:
52          # Skip any listings that are malformed
53          continue
54
55      print(f' ✅ Page {page} scraped')
56
57  # Close browser
58  driver.quit()
59
60  # Save to CSV
61  df = pd.DataFrame(data)
62  df.to_csv('tayara_vehicles.csv', index=False, encoding='utf-8-sig')
63  print(" ✅ Scraping completed and data saved to 'tayara_vehicles.csv' ")
64
```

Python Code

Encryption

```
encryter.py > ...
1  from cryptography.fernet import Fernet
2
3  # Generate a key (only run this ONCE and save the key)
4  key = Fernet.generate_key()
5
6  # Save the key securely
7  with open("encryption.key", "wb") as key_file:
8      key_file.write(key)
9
10 # Initialize Fernet with the key
11 fernet = Fernet(key)
12
13 # Read the CSV file to encrypt
14 with open("tayara_vehicles.csv", "rb") as file:
15     original = file.read()
16
17 # Encrypt the data
18 encrypted = fernet.encrypt(original)
19
20 # Save the encrypted data to a file
21 with open("tayara_vehicles_encrypted.csv", "wb") as enc_file:
22     enc_file.write(encrypted)
23
24 print("✅ tayara_vehicles.csv has been encrypted.")
```

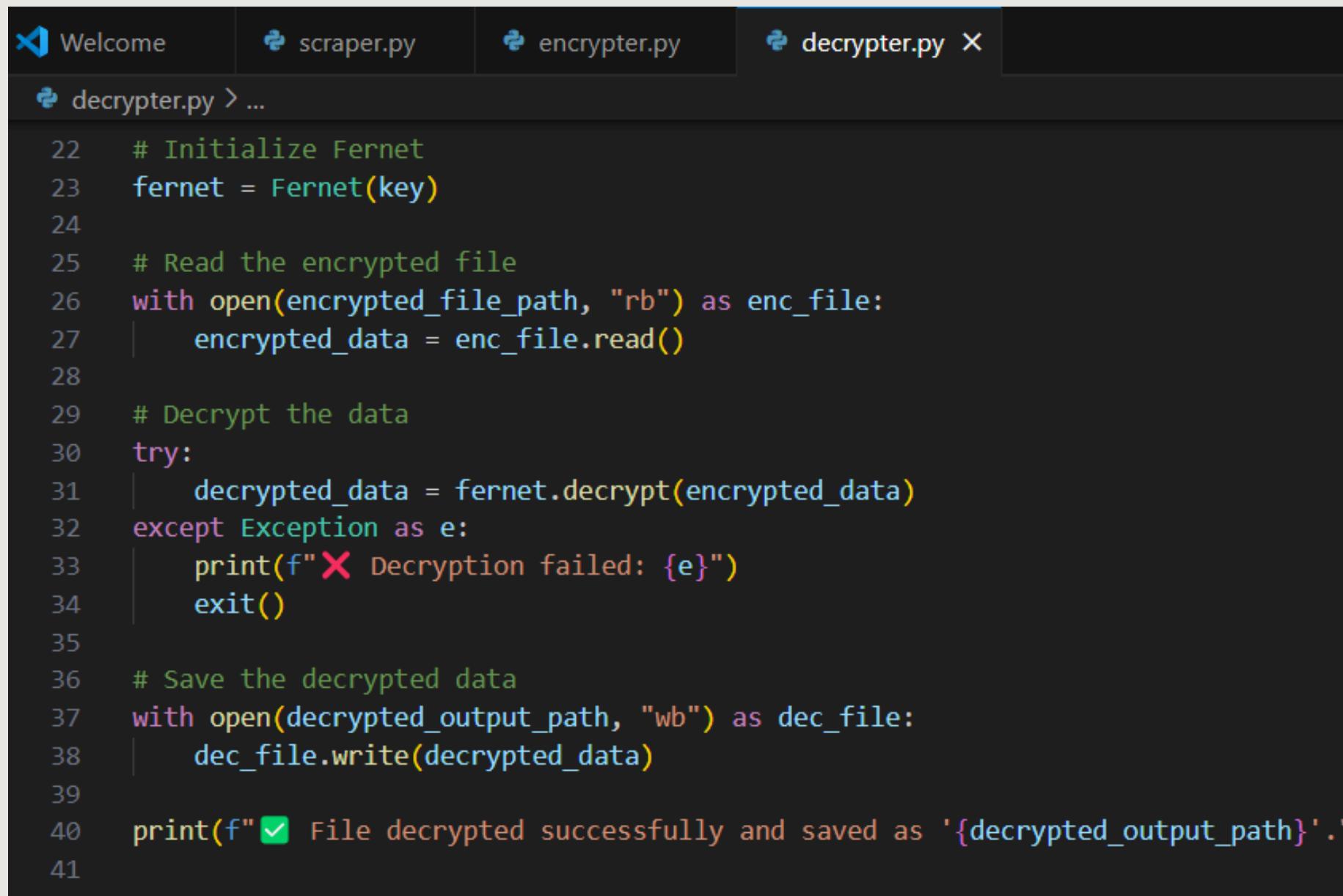
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

✓ tayara_vehicles.csv has been encrypted.

PS C:\Users\MSI\Desktop\JUNIOR\SECURITY\Projet> & C:/Users/MSI/AppData/Local/Programs/Python/Python313/python.exe encryter.py

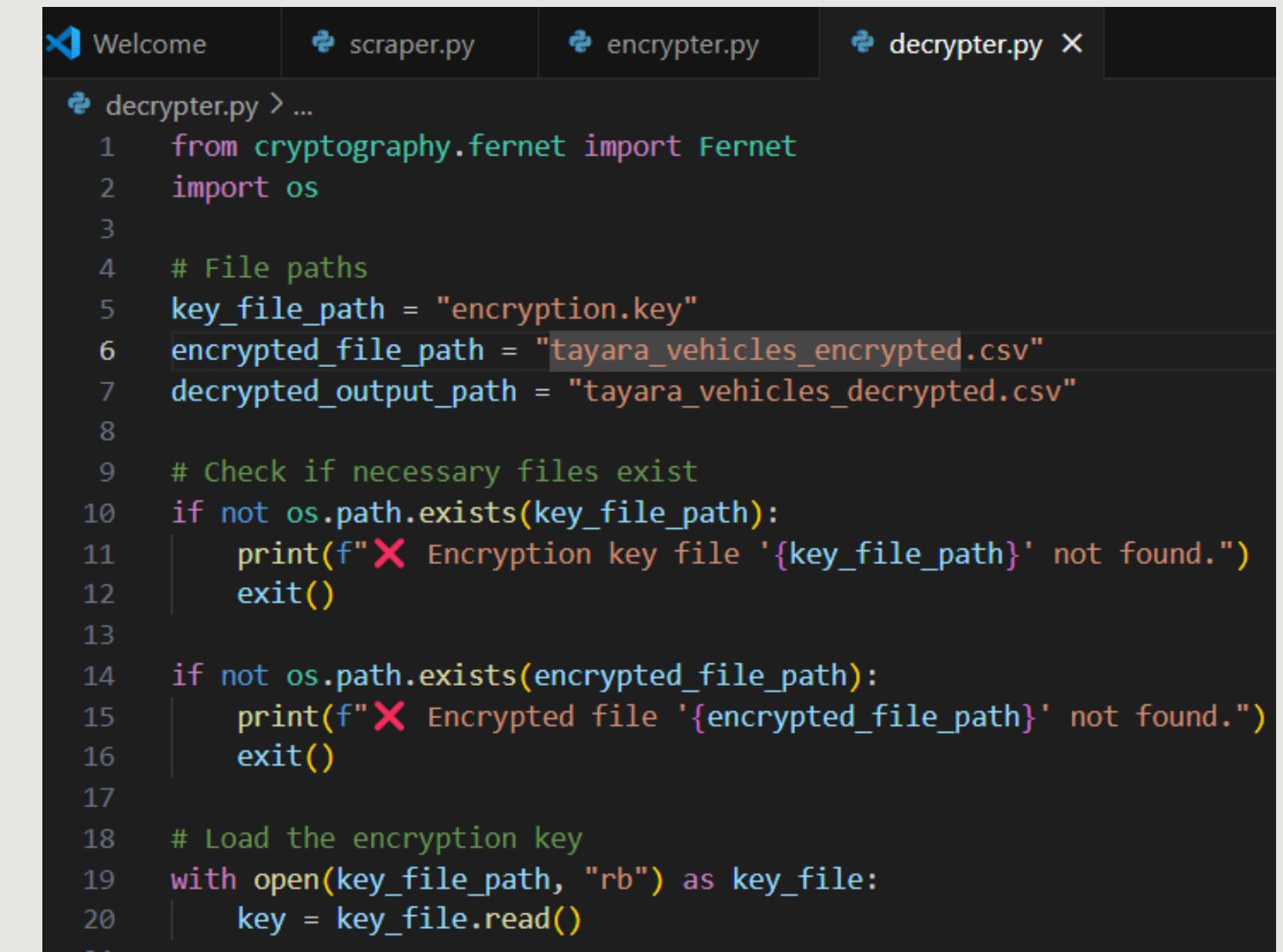
Python Code

Decryption



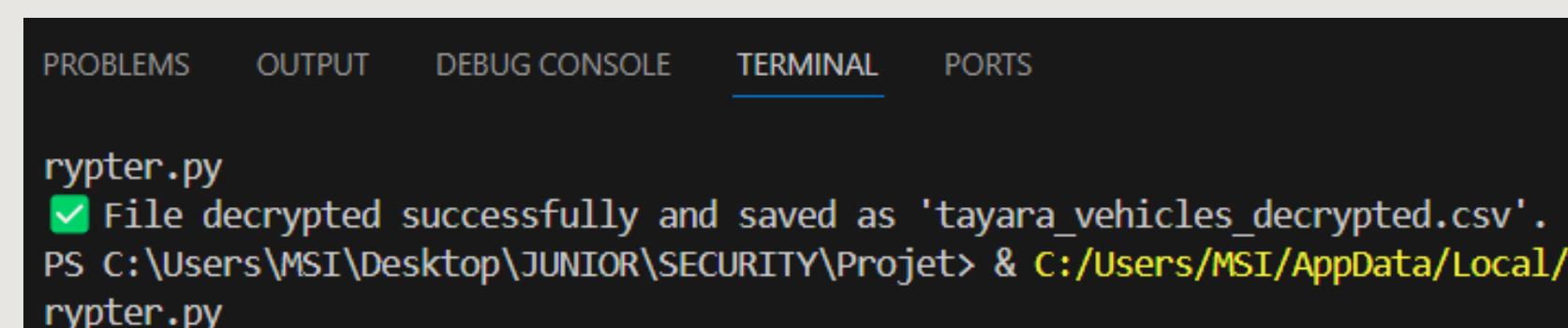
The screenshot shows the VS Code interface with the 'decrypter.py' file open in the center editor tab. The code implements a Fernet cipher to decrypt an encrypted CSV file. It reads the encrypted data from a file, decrypts it, and then writes the decrypted data to a new file. It includes error handling for decryption failures and checks if necessary files exist.

```
1 # Initialize Fernet
2 fernet = Fernet(key)
3
4 # Read the encrypted file
5 with open(encrypted_file_path, "rb") as enc_file:
6     encrypted_data = enc_file.read()
7
8 # Decrypt the data
9 try:
10     decrypted_data = fernet.decrypt(encrypted_data)
11 except Exception as e:
12     print(f"🔴 Decryption failed: {e}")
13     exit()
14
15 # Save the decrypted data
16 with open(decrypted_output_path, "wb") as dec_file:
17     dec_file.write(decrypted_data)
18
19 print(f"✅ File decrypted successfully and saved as '{decrypted_output_path}'.")
```



The screenshot shows the VS Code interface with the 'decrypter.py' file open in the center editor tab. The code uses the 'cryptography.fernet' module to handle the decryption process. It defines file paths for the key and the encrypted file, checks if they exist, and then loads the encryption key from the key file to decrypt the data.

```
1 from cryptography.fernet import Fernet
2 import os
3
4 # File paths
5 key_file_path = "encryption.key"
6 encrypted_file_path = "tayara_vehicles_encrypted.csv"
7 decrypted_output_path = "tayara_vehicles_decrypted.csv"
8
9 # Check if necessary files exist
10 if not os.path.exists(key_file_path):
11     print(f"🔴 Encryption key file '{key_file_path}' not found.")
12     exit()
13
14 if not os.path.exists(encrypted_file_path):
15     print(f"🔴 Encrypted file '{encrypted_file_path}' not found.")
16     exit()
17
18 # Load the encryption key
19 with open(key_file_path, "rb") as key_file:
20     key = key_file.read()
```



The screenshot shows the VS Code terminal window at the bottom. It displays the command 'python decrypter.py' being run, followed by the output message indicating successful decryption and saving of the file. The terminal also shows the current working directory as 'C:/Users/MSI/Desktop/JUNIOR/SECURITY/Projet'.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
rypter.py
✅ File decrypted successfully and saved as 'tayara_vehicles_decrypted.csv'.
PS C:\Users\MSI\Desktop\JUNIOR\SECURITY\Projet & C:/Users/MSI/AppData/Local/rypter.py
```

THANK YOU

For Following Along