$ I

Self-healing system for UI tests using Machine Learning

Andressa Cabistani
Software Quality Engineer

GitHub: https://github.com/andressadotpy
LinkedIn: https://www.linkedin.com/in/andressacabistani/

$ whoami

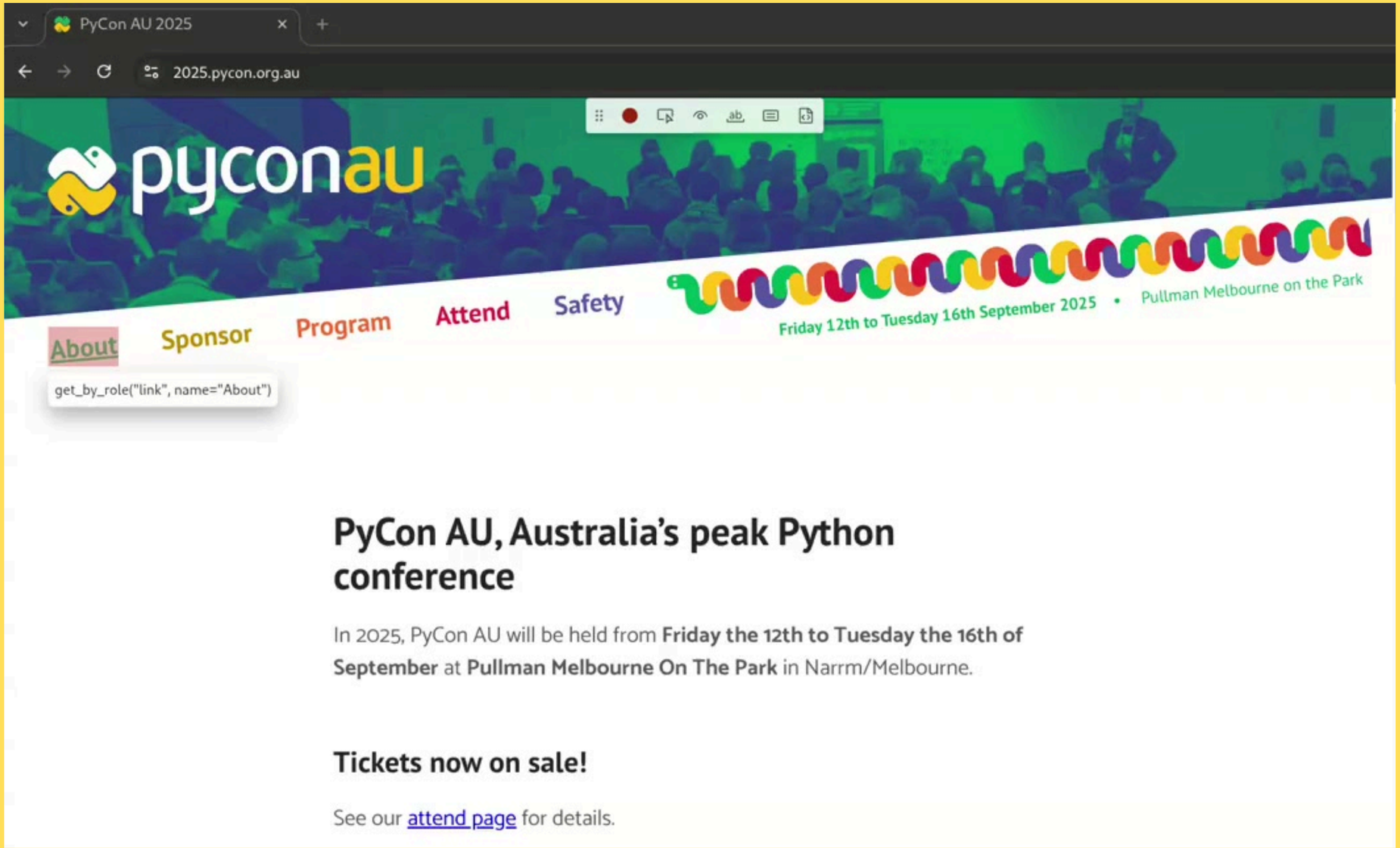
Brazilian
Software Quality Engineer at Red Hat
I love to learn
Mom
Star Wars fan

$ cat “How a UI test works”.md

We use selectors to identify elements in the UI
and perform actions

E.g. Get the CSS selector
for the About link in PyCon AU 2025 website
#header > div > nav > ul > li:nth-child(1) > a
and use a function from an automation framework to
click()

$ I


PyCon AU 2025
2025.pycon.org.au
pyconau
About
Sponsor
Program
Attend
Safety
Friday 12th to Tuesday 16th September 2025 • Pullman Melbourne on the Park
get_by_role("link", name="About")
PyCon AU, Australia's peak Python conference
In 2025, PyCon AU will be held from Friday the 12th to Tuesday the 16th of September at Pullman Melbourne On The Park in Narrm/Melbourne.
Tickets now on sale!
See our attend page for details.

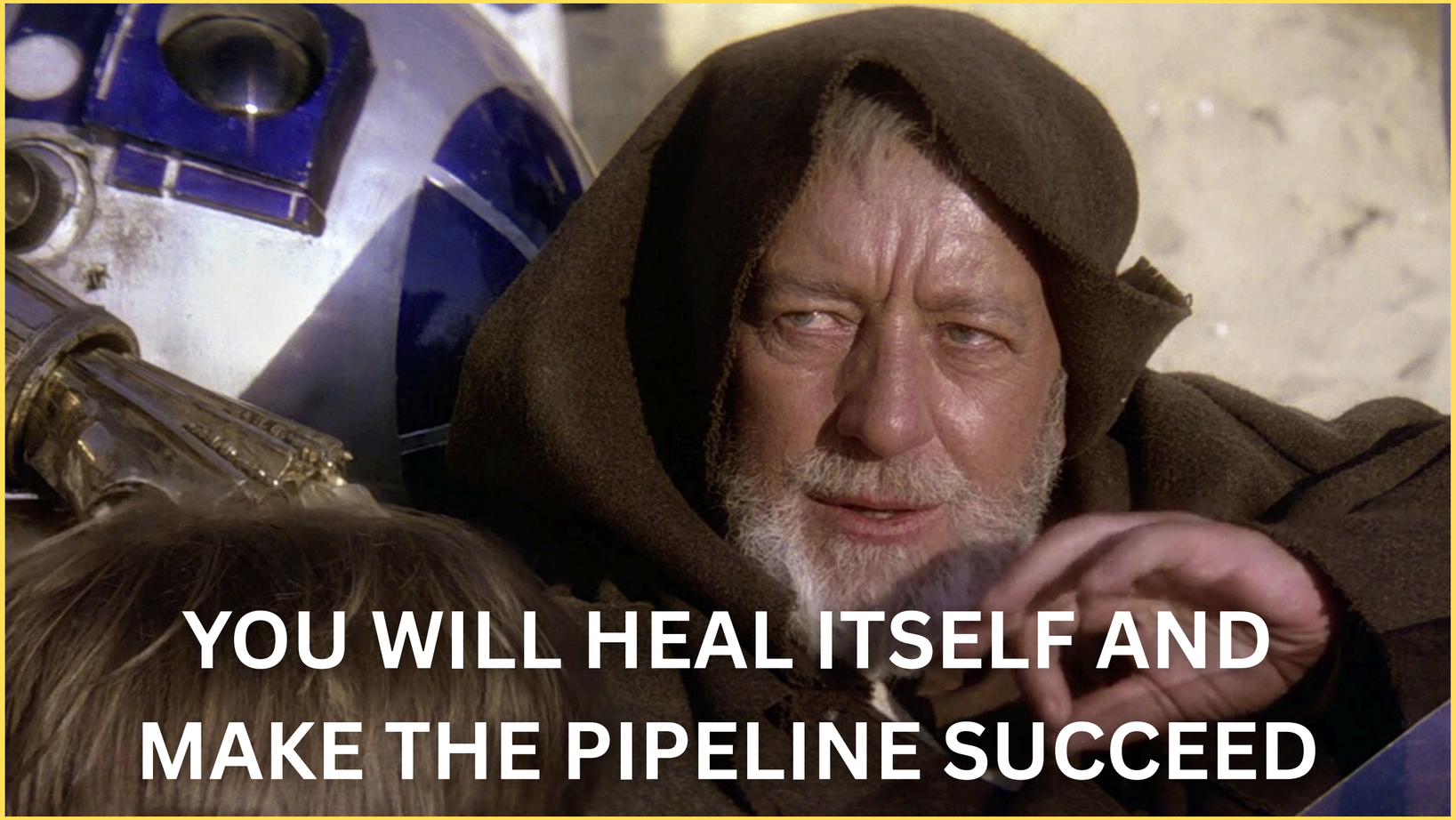
$ cat “Regression tests and CI/CD”.md

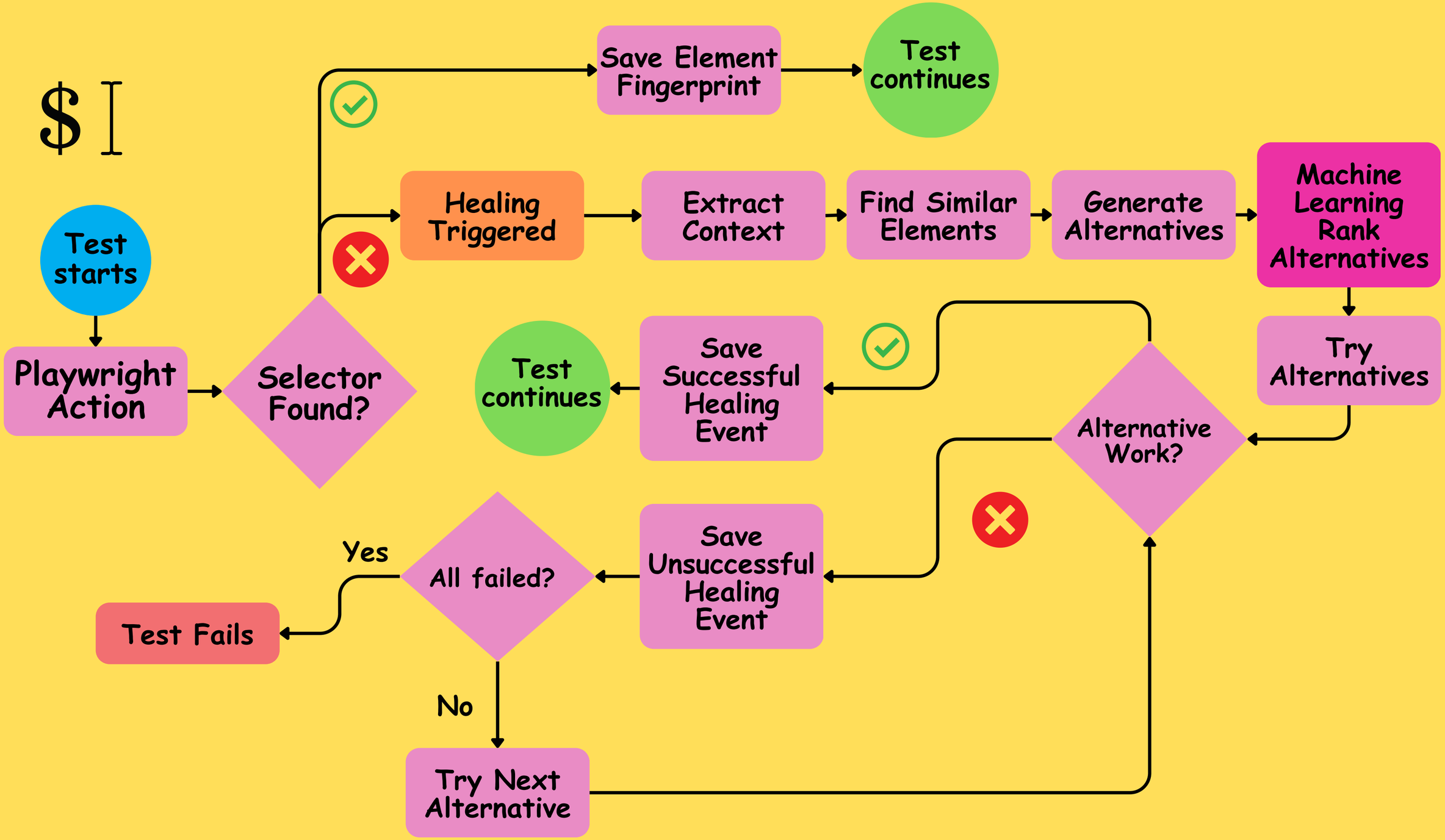
Regression tests run in a CI/CD pipeline to guarantee the changes won’t affect working parts of a Software.

$ cat “UI test problem”.md

Minor front-end changes to attributes (e.g id, class, data-*) break the selectors causing failures during a release process that aren’t real issues, because the functionality is still working.

$ cat “Self-healing system for UI test failures”.md




Test starts
Playwright Action
Selector Found?
Save Element Fingerprint
Test continues
Healing Triggered
Extract Context
Find Similar Elements
Generate Alternatives
Machine Learning Rank Alternatives
Try Alternatives
Alternative Work?
Save Successful Healing Event
Test continues
Save Unsuccessful Healing Event
All failed?
Yes
Test Fails
No
Try Next Alternative

$ I

SelfHealingPage.<locator_method>() →
SelfHealingLocator →
.<action_method>()

$ cat "Differences in how to use SelfHealingPage and Page".md

```python
@pytest.fixture
def pyconau_page(page):
    return PyConAUPage(page)
```

```python
@pytest.fixture(scope="session")
def healing_db():
    return SelfHealingDB("postgresql://postgres:postgres@localhost:5432/healing_db")


@pytest.fixture
def pyconau_page(page, healing_db):
    self_healing_page = SelfHealingPage(page, healing_db)
    return PyConAUPage(self_healing_page)
```

$ I

```
pyconau_page.page.get_by_role("link", name="About").click()
```

$ cat “get_by_role() method”.md

```python
def get_by_role(self, role: str, **kwargs) -> SelfHealingLocator:
    """
    Override get_by_role to return a self-healing locator.

    Returns a SelfHealingLocator that can capture fingerprints and attempt healing.
    """
    locator = self._page.get_by_role(role, **kwargs)
    selector = f"role:{role}"
    if "name" in kwargs:
        selector += f"[name={kwargs['name']}]"
    return SelfHealingLocator(locator, selector, "get_by_role", self, self._healing_service)
```

$ cat "Store fingerprints when action is successful ".md

```python
def _save_successful_interaction(self, action: str) -> None:
    """
    Save fingerprint for successful interaction.

    Args:
        action: The action that was performed (click, fill, etc.)
    """
    if self._healing_page._database is not None:
        try:
            fingerprint = self._healing_page._extract_element_fingerprint(self._selector, self._selector_type)
            if fingerprint:
                fingerprint_id = self._healing_page._database.insert_fingerprint(fingerprint)
                if fingerprint_id:
                    logger.info(
                        f"Saved new fingerprint (ID: {fingerprint_id}) for successful {action} on locator: {self._selector}"
                    )
                else:
                    logger.debug(f"Fingerprint already exists for locator: {self._selector} - skipped duplicate")
        except Exception as e:
            logger.warning(f"Failed to save fingerprint for {action} on {self._selector}: {e}")
```

$ cat “Fingerprints that are stored”.md

	Row #1
id	194
tag_name	a
inner_text	About
text_content	About
text_length	5
attributes	{"href": "/about", "data-astro-cid-pux6a34n": ""}
dom_position	body:nth-child(2) > nav > ul:nth-child(2) > li > a
is_visible	[v]
parent_info	{"id": null, "tag": "li", "class": null}
accessibility	{"role": null, "aria-label": null, "aria-labelledby": null}
surrounding_text	About
key_styles	{"color": "rgb(0, 171, 86)", "width": "auto", "height": "auto", "display": "inline", "visibility": "visible", "background-color": "rgba(0, 0, 0, 0)"}
original_selector	role:link[name=About]
page_url	https://2025.pycon.org.au/about
selector_type	get_by_role
created_at	2025-09-03 22:47:20.536 GMT-03:00
updated_at	2025-09-03 22:47:20.536 GMT-03:00

$ I

```
pyconau_page.page.get_by_role("link", name="Abt").click()
```

$ cat “Handling TimeoutError inside the action method”.md

```python
except TimeoutError:
    if self._healing_service:
        logger.warning(
            f"SelfHealingLocator: Click TIMEOUT on selector '{self._selector}', attempting healing..."
        )

        def click_action(healed_selector, *args, **kwargs):
            healing_kwargs = kwargs.copy()
            healing_kwargs["timeout"] = min(kwargs.get("timeout", 2000), 2000)
            return self._get_healed_locator(healed_selector).click(*args, **healing_kwargs)

        return self._healing_service.attempt_healing(
            page=self._healing_page.original_page,
            selector=self._selector,
            action_callable=click_action,
            modifiers=modifiers,
            position=position,
            delay=delay,
            button=button,
            click_count=click_count,
            timeout=timeout,
            force=force,
            no_wait_after=no_wait_after,
            trial=trial,
        )
    else:
        logger.error(
            f"SelfHealingLocator: Click TIMEOUT on selector '{self._selector}' and no healing service available"
        )
        raise
```

$ cat "Extracting information from the failing selector".md

```python
def extract_element_context(self, page: Page, selector: str) -> Dict:
    try:
        context = {
            "tag_name": "div",
            "inner_text": "",
            "attributes": {},
            "dom_position": "",
            "is_visible": False,
            "parent_info": {},
            "accessibility": {},
            "surrounding_text": "",
            "key_styles": {},
        }

        try:
            page_url = page.url
            context["page_url"] = page_url
        except Exception:
            pass

        try:
            body_elements = page.locator("body *").count()
            if isinstance(body_elements, int):
                context["page_complexity"] = body_elements
        except Exception:
            pass

        context.update(self._infer_context_from_selector(selector))

        return context

    except Exception as e:
        logger.debug(f"Could not extract element context: {e}")
        return {"tag_name": "div", "inner_text": "", "attributes": {}, "dom_position": "", "is_visible": False}
```

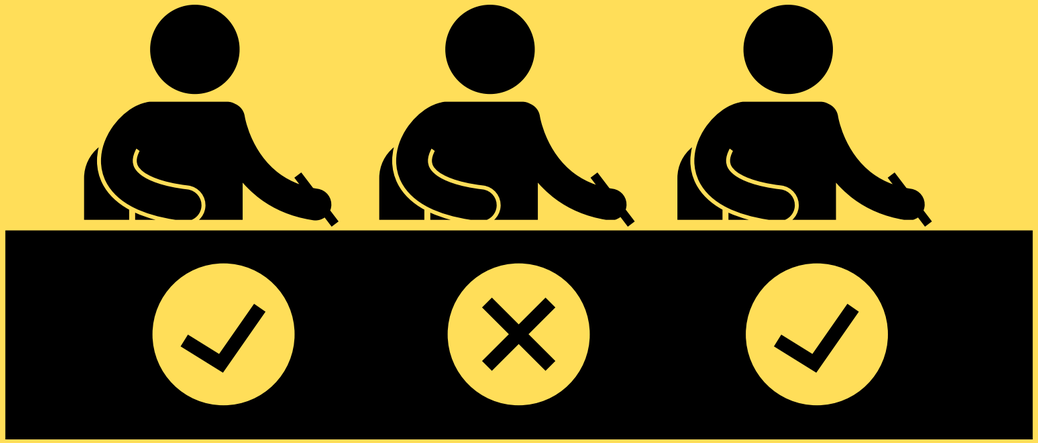
$ cat "Generating alternatives from similar fingerprints".md

```python
def generate_all_alternatives(
    self,
    failed_selector: str,
    page_url: str,
    current_dom_context: Dict,
) -> List[str]:
    logger.info(f"Generating alternatives for failed selector: {failed_selector}")

    fingerprints = self._find_similar_fingerprints(page_url, current_dom_context)

    if not fingerprints:
        logger.warning(f"No similar fingerprints found for {failed_selector} on {page_url}")
        return []

    logger.info(f"Found {len(fingerprints)} similar fingerprints for alternative generation")

    all_alternatives = []
    for strategy in self.strategies:
        strategy_alternatives = strategy.generate_all(fingerprints)
        all_alternatives.extend(strategy_alternatives)
        logger.debug(f"{strategy.__class__.__name__} generated {len(strategy_alternatives)} alternatives")

    unique_alternatives = list(dict.fromkeys(all_alternatives))

    logger.info(f"Generated {len(unique_alternatives)} unique alternative selectors")
    return unique_alternatives
```

$ I


```
{
  "predictions": [
    {
      "selector": "get_by_role(\"link\", name=\"About\", exact=False)",
      "confidence": 0.92
    },
    {
      "selector": "//a[@href=\"/about/culture\"]",
      "confidence": 0.91
    },
    {
      "selector": "get_by_role(\"link\", name=\"About\")",
      "confidence": 0.91
    },
    {
      "selector": "//a[@href=\"/about\"]",
      "confidence": 0.85
    },
    {
      "selector": "ul:nth-child(2) > li.current:nth-child(2) > a",
      "confidence": 0.33
    },
    {
      "selector": "div.scroll-container:nth-child(2) > div.section-nav > ul:nth-child(2) > li.current:nth-child(2) > a",
      "confidence": 0.32
    },
    {
      "selector": "li.current:nth-child(2) > a",
      "confidence": 0.29
    }
  ],
  "ranked_alternatives": [
    "get_by_role(\"link\", name=\"About\", exact=False)",
    "//a[@href=\"/about/culture\"]",
    "get_by_role(\"link\", name=\"About\")",
    "//a[@href=\"/about\"]",
    "ul:nth-child(2) > li.current:nth-child(2) > a",
    "div.scroll-container:nth-child(2) > div.section-nav > ul:nth-child(2) > li.current:nth-child(2) > a",
    "li.current:nth-child(2) > a"
  ],
  "model_version": "1.0.0"
}
```

$ cat “How Scikit-learn RandomForestClassifier works”.md



$ cat “Transforming healing events to numbers”.md

Selector Features

#submit-btn - Complexity score 1
div.form > ul:nth-child(2) > li[data-active="true"] - Complexity score 9
(fragile and complex)

$ cat “Transforming healing events to numbers”.md

Similarity Features

Intersection (unique common characters):
{'g', 'e', 't', '_', 'b', 'y', 'r', 'o', 'l', '(', '"', 'i', 'n', 'k', ',', ' ', 'm', 'a', '=', ')'}
Size = 20

Union (all unique characters):
{'g', 'e', 't', '_', 'b', 'y', 'r', 'o', 'l', '(', '"', 'i', 'n', 'k', ',', ' ', 'm', 'a', '=', ')', 'u'}
Size = 21

Jaccard Similarity = 20/21 = 0.952 - Very similar

$ cat “Transforming healing events to numbers”.md

Context Features

https://2025.pycon.org.au/
url_length=29, domain_length=19, path_length=1, path_depth=0, has_query_params=0;
https://2025.pycon.org.au/about/
url_length=35, domain_length=19, path_length=7, path_depth=1, has_query_params=0;

$ cat "Transforming healing events to numbers".md

Reliability Features

Converting best practices for "good" and "bad" selectors into numerical scores.

$ cat “Transforming healing events to numbers”.md

DOM Features

body > div.main-content > nav.header > ul.menu > li:nth-child(2) > a

dom_position_length = 61 (total characters in the position string)

dom_depth = 5 (number of structural levels from body to the target element)

$ cat “Transforming healing events to numbers”.md

Text Features

get_by_role(“link”, name=”About”)
has_inner_text = 1
inner_text_length = 5
text_content_length = 5
text_word_count = 1
text_is_numeric = 0
text_has_spaces = 0

$ I

acabista@acabista-thinkpadp16vgen1:~/Documents/automations/pyconau

```
(pycon) acabista@acabista-thinkpadp16vgen1:~/Documents/automations/
pyconau$ pytest --headed test_without_healing.py
========================= test session starts =========================
platform linux -- Python 3.12.10, pytest-8.4.1, pluggy-1.6.0
rootdir: /home/acabista/Documents/automations/pyconau
plugins: rhsm-0.1.0, base-url-2.1.0, playwright-0.7.0
collected 1 item

test_without_healing.py
```

test_without_healing.py

```
test

_page import PyConAUPage


ixture
au_page(page):
 PyConAUPage(page)


avigation_without_healing(pyconau_page):
au_page.navigate()
au_page.page.get_by_role("link", name="Abt").click()
au_page.page.get_by_role("link", name="the culture of PyCon AU", exa
au_page.page.get_by_role("link", name="Sponsor").click()
au_page.page.get_by_role("listitem").filter(has_text="Attend").click(
au_page.page.get_by_role("link", name="Get your tickets now!").click(
au_page.page.get_by_text("Sept. 12th - until 16th, 2025 Begin: 09:00
au_page.page.get_by_role("heading", name="PyCon AU 2025 Sept. 12th").
au_page.page.get_by_label("Professional - Early Bird").get_by_role("b
au_page.page.get_by_role("button", name="Hide variants").click()
```

PyCon AU 2025

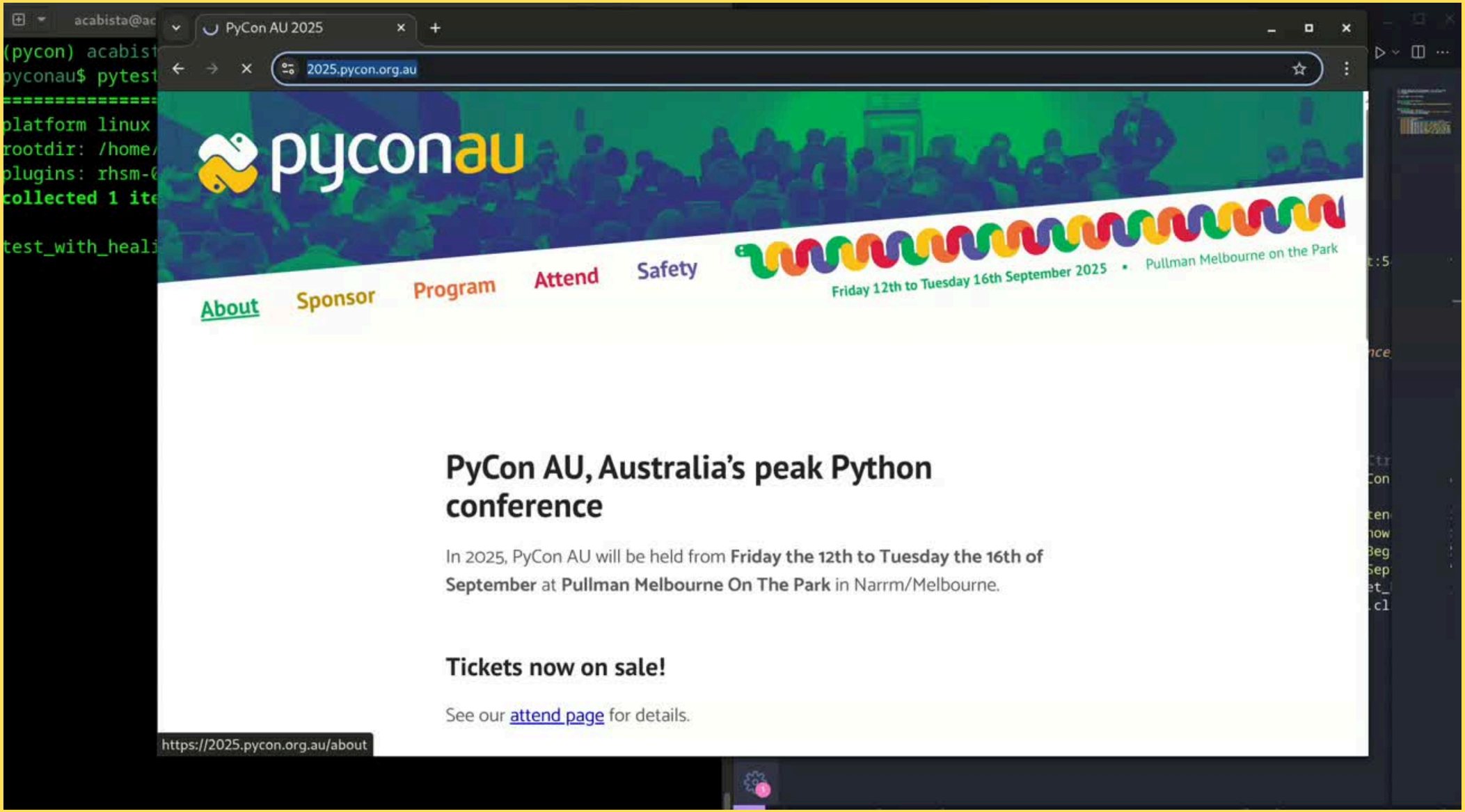
2025.pycon.org.au

pyconau

About Sponsor Program Attend Safety

Friday 12th to Tuesday 16th September 2025 • Pullman Melb

$ I


acabista@ac
(pycon) acabist
pyconau$ pytest
platform linux
rootdir: /home/
plugins: rhsm-6
collected 1 ite
test_with_heali
PyCon AU 2025
2025.pycon.org.au
pyconau
About
Sponsor
Program
Attend
Safety
Friday 12th to Tuesday 16th September 2025 • Pullman Melbourne on the Park
PyCon AU, Australia's peak Python conference
In 2025, PyCon AU will be held from Friday the 12th to Tuesday the 16th of September at Pullman Melbourne On The Park in Narrm/Melbourne.
Tickets now on sale!
See our attend page for details.
https://2025.pycon.org.au/about

$ cat “Future improvements and final considerations”.md

More healing events, more data and a more refined model

Notification system

```
$echo "Thank you"
```