```
import os
import time
import random
from datetime import datetime
from playwright.sync_api import sync_playwright, TimeoutError as PlaywrightTimeou

COUNTRY        = os.getenv("SPORTYBET_COUNTRY", "ng")
MAX_SELECTIONS = int(os.getenv("MAX_SELECTIONS", "5"))
MIN_ODDS       = float(os.getenv("MIN_ODDS", "1.30"))
MAX_ODDS       = float(os.getenv("MAX_ODDS", "5.00"))
SELECTION_MODE = os.getenv("SELECTION_MODE", "favourite")

COUNTRY_URLS = {
    "ng": "https://www.sportybet.com/ng",
    "gh": "https://www.sportybet.com/gh",
    "ke": "https://www.sportybet.com/ke",
    "tz": "https://www.sportybet.com/tz",
    "ug": "https://www.sportybet.com/ug",
    "zm": "https://www.sportybet.com/zm",
}


def log(msg):
    print("[" + datetime.now().strftime("%H:%M:%S") + "] " + str(msg), flush=True


def run_full_generation(country=None, max_sel=None, min_odds=None, max_odds=None,
    country  = country  or COUNTRY
    max_sel  = max_sel  or MAX_SELECTIONS
    min_odds = min_odds or MIN_ODDS
    max_odds = max_odds or MAX_ODDS
    mode     = mode     or SELECTION_MODE

    base_url = COUNTRY_URLS.get(country, COUNTRY_URLS["ng"])
    log("Starting Playwright session for " + country.upper())

    with sync_playwright() as p:
        browser = p.chromium.launch(
            headless=True,
            args=[
                "--no-sandbox",
                "--disable-setuid-sandbox",
                "--disable-dev-shm-usage",
                "--disable-gpu",
                "--no-first-run",
                "--no-zygote",
                "--single-process",
            ]
        )

        context = browser.new_context(
            user_agent="Mozilla/5.0 (Linux; Android 12; SM-G991B) AppleWebKit/537
            viewport={"width": 390, "height": 844},
            locale="en-GB",
        )

        page = context.new_page()

        try:
            highlights_url = base_url + "/m/football/highlights"
            log("Navigating to " + highlights_url)
            page.goto(highlights_url, wait_until="domcontentloaded", timeout=3000
            time.sleep(3)

            for selector in ["button:has-text(\"Accept\")", "button:has-text(\"OK
                try:
                    btn = page.locator(selector).first
                    if btn.is_visible(timeout=2000):
                        btn.click()
                        time.sleep(1)
                except Exception:
                    pass

            log("Scanning for matches...")
            added_matches = []

            odds_rows = page.locator("[class*=\"event-item\"], [class*=\"match-it
            row_count = odds_rows.count()
            log("Found " + str(row_count) + " match rows")

            clicked = 0
            attempted_rows = 0

            while clicked < max_sel and attempted_rows < min(row_count, 30):
                try:
                    row = odds_rows.nth(attempted_rows)
                    attempted_rows += 1

                    try:
                        match_name = row.locator("[class*=\"team\"], [class*=\"na
                    except Exception:
```

```
                match_name = "Match " + str(attempted_rows)

            btns = row.locator("button, [class*=\"odds\"], [class*=\"odd\
            btn_count = btns.count()
            if btn_count == 0:
                continue

            valid_btns = []
            for i in range(btn_count):
                try:
                    btn = btns.nth(i)
                    text = btn.inner_text(timeout=1000).strip()
                    odds_val = float(text)
                    if min_odds <= odds_val <= max_odds:
                        valid_btns.append((i, odds_val, btn))
                except Exception:
                    continue

            if not valid_btns:
                continue

            if mode == "favourite":
                chosen = min(valid_btns, key=lambda x: x[1])
            elif mode == "random":
                chosen = random.choice(valid_btns)
            else:
                chosen = valid_btns[0]

            chosen[2].click()
            time.sleep(0.8)
            clicked += 1
            pick_labels = ["1", "X", "2"]
            pick = pick_labels[chosen[0]] if chosen[0] < 3 else "?"
            added_matches.append({
                "match": match_name.strip(),
                "odds": chosen[1],
                "pick": pick
            })
            log("Added: " + match_name + " @ " + str(chosen[1]))

        except Exception as e:
            log("Row " + str(attempted_rows) + " error: " + str(e))
            continue

    if clicked == 0:
        browser.close()
        return {"error": "Could not add any selections. SportyBet page st
```

```
log("Added " + str(clicked) + " selections. Looking for share button.
time.sleep(2)

share_selectors = [
    "button:has-text(\"Share\")",
    "button:has-text(\"Book\")",
    "button:has-text(\"Code\")",
    "[class*=\"share\"]",
    "[class*=\"book-code\"]",
    "[class*=\"booking\"]",
]

for sel in share_selectors:
    try:
        btn = page.locator(sel).first
        if btn.is_visible(timeout=3000):
            btn.click()
            log("Clicked: " + sel)
            time.sleep(3)
            break
    except Exception:
        continue

share_code = None

current_url = page.url
if "shareCode=" in current_url or "code=" in current_url:
    from urllib.parse import urlparse, parse_qs
    params = parse_qs(urlparse(current_url).query)
    codes = params.get("shareCode", params.get("code", [None]))
    share_code = codes[0] if codes else None

if not share_code:
    for sel in ["[class*=\"share-code\"]", "[class*=\"booking-code\"]
        try:
            el = page.locator(sel).first
            if el.is_visible(timeout=2000):
                text = el.inner_text(timeout=2000).strip()
                if text and 3 < len(text) < 30:
                    share_code = text
                    break
        except Exception:
            continue

if not share_code:
    try:
```

```
                    inputs = page.locator("input")
                    for i in range(inputs.count()):
                        val = inputs.nth(i).get_attribute("value", timeout=1000)
                        if val and 4 <= len(val) <= 20:
                            share_code = val
                            break
                except Exception:
                    pass

            browser.close()

            if not share_code:
                return {
                    "error": "Selections added but could not extract booking code
                    "partial": {"selections": added_matches, "count": clicked}
                }

            return {
                "success":     True,
                "shareCode":   share_code,
                "deepLink":    "https://www.sportybet.com/" + country + "/?shareC
                "country":     country.upper(),
                "selections":  added_matches,
                "generatedAt": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            }

        except PlaywrightTimeout:
            browser.close()
            return {"error": "Page timed out. Try again."}
        except Exception as e:
            browser.close()
            return {"error": "Error: " + str(e)}


def load_code(share_code, country=None):
    country = country or COUNTRY
    return {
        "shareCode": share_code,
        "deepLink": "https://www.sportybet.com/" + country + "/?shareCode=" + sha
    }
```