```python
import unittest
import time
import selenium.common.exceptions as ex
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from openpyxl import Workbook

class ShortVehicleMaster(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
        cls.driver.maximize_window()
        cls.wait = WebDriverWait(cls.driver, 10)

    def click_element(self, by, value, retry=2):
        for i in range(retry):
            try:
                self.wait.until(EC.element_to_be_clickable((by, value))).click()
                return True
            except (ex.ElementClickInterceptedException, ex.StaleElementReferenceException, ex.TimeoutException):
                time.sleep(1)
        try:
            element = self.driver.find_element(by, value)
            self.driver.execute_script("arguments[0].click();", element)
            return True
        except:
            return False

    def switch_frames(self, element_id):
        driver = self.driver
        driver.switch_to.default_content()
        iframes = driver.find_elements(By.TAG_NAME, "iframe")
        for iframe in iframes:
            driver.switch_to.frame(iframe)
            try:
                if driver.find_element(By.ID, element_id):
                    return True
            except ex.NoSuchElementException:
                driver.switch_to.default_content()
        return False

    def send_keys(self, by, value, text):
        try:
            element = self.wait.until(EC.visibility_of_element_located((by, value)))
            element.clear()
            element.send_keys(text)
            return True
        except ex.NoSuchElementException:
            return False


    def test_short_market_vehicle(self):
        driver = self.driver
        driver.get("")
        self.send_keys(By.ID, "Login", "")
        self.send_keys(By.ID, "Password", "")
        self.click_element(By.ID, "btnLogin")

        self.click_element(By.XPATH, "(//a[normalize-space()='Fleet'])[1]")
        self.click_element(By.XPATH, "(//a[normalize-space()='Fleet Master »'])[1]")
        self.click_element(By.XPATH, "//a[normalize-space()='Vehicle »']")
        self.click_element(By.XPATH, "//a[normalize-space()='Short Market Vehicle']")

        if self.switch_frames("btn_NewRecord"):
            self.click_element(By.ID, "btn_NewRecord")
            time.sleep(2)

            # Vehicle e-KYC Detail
            if self.switch_frames("acaretdowndivEkycDetails"):
                self.click_element(By.ID,"acaretdowndivEkycDetails")
                self.send_keys(By.ID,"ekycVehicleNo","RJ32GE0163")
                if self.switch_frames("btn_SearchVehicleDtl_Referesh"):
                    self.click_element(By.ID, "btn_SearchVehicleDtl_Referesh")
                    time.sleep(2)
```

```
                vehicle_data = {
                    "Vehicle No": driver.find_element(By.ID, "ekycVehicleNo").get_attribute("value"),
                    "RC Status": driver.find_element(By.ID, "ekycVehicleRCStatus").get_attribute("value"),
                    "Blacklist Status": driver.find_element(By.ID, "ekycNonUseStatus").get_attribute("value"),
                    "Registered At": driver.find_element(By.ID, "ekycRegisteredAt").get_attribute("value"),
                    "Issue Date": driver.find_element(By.ID, "ekycIssueDate").get_attribute("value"),
                    "Owner Name": driver.find_element(By.ID, "ekycOwnerName").get_attribute("value"),
                    "Permanent Address": driver.find_element(By.ID, "ekycPermanentAddress").get_attribute("value"),
                    "Engine Number": driver.find_element(By.ID, "ekycVehicleEngineNumber").get_attribute("value"),
                    "Chassis Number": driver.find_element(By.ID, "ekycVehicleChassisNumber").get_attribute("value"),
                    "Gross Weight": driver.find_element(By.ID, "ekycVehicleGrossWeight").get_attribute("value"),
                    "Unladen Weight": driver.find_element(By.ID, "ekycVehicleUnladenWeight").get_attribute("value"),
                    "Maker Model": driver.find_element(By.ID, "ekycVehicleMakerModel").get_attribute("value"),
                    "PUC Expiry Date": driver.find_element(By.ID, "ekycPucExpiryDate").get_attribute("value"),
                    "Fitness Expiry Date": driver.find_element(By.ID, "ekycExpiryDate").get_attribute("value"),
                    "Tax End Date": driver.find_element(By.ID, "ekycTaxEndDate").get_attribute("value"),
                    "Financier": driver.find_element(By.ID, "ekycFinancier").get_attribute("value"),
                    "Permit Number": driver.find_element(By.ID, "ekycPermitNumber").get_attribute("value"),
                    "Permit Expiry Date": driver.find_element(By.ID, "ekycPermitExpiryDate").get_attribute("value"),
                    "Owner Serial No": driver.find_element(By.ID, "ekycOwnerSerialNo").get_attribute("value"),
                    "National Permit Number": driver.find_element(By.ID, "ekycNationalPermitNumber").get_attribute(
                        "value"),
                    "National Permit Expiry Date": driver.find_element(By.ID,
                                                                       "ekycNationalPermitExpiryDate").get_attribute(
                        "value"),
                    "Insurance Company": driver.find_element(By.ID, "ekycInsuranceCompany").get_attribute("value"),
                    "Insurance Policy No": driver.find_element(By.ID, "ekycInsurancePolicyNumber").get_attribute(
                        "value"),
                    "Insurance Expiry Date": driver.find_element(By.ID, "ekycInsuranceExpiryDate").get_attribute(
                        "value"),
                    "Log DateTime": driver.find_element(By.ID, "ekycLogDateTime").get_attribute("value"),
                    "Vehicle Capacity": driver.find_element(By.ID, "ekycVehicleCapacity").get_attribute("value")
                }

                # Save to Excel using openpyxl
                vehicle_number = vehicle_data["Vehicle No"]
                wb = Workbook()
                ws = wb.active
                ws.title = "Vehicle Data"

                # Add headers and values
                ws.append(["Field", "Value"])
                for key, value in vehicle_data.items():
                    ws.append([key, value])

                # Save the file
                file_name = f"{vehicle_number}.xlsx"
                wb.save(file_name)
                print(f"Data saved to {file_name}")

    @classmethod
    def tearDownClass(cls):
        cls.driver.quit()

if __name__ == "__main__":
    unittest.main()
```