Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
1	UI	JEST	@profile def main(): energy_manager = EnergyManager(1.52) root = tk.Tk() root.title("Energy Manager") root.geometry("1920x1080") control_frame = tk.Frame(root) control_frame.pack(fill="both") util_frame = tk.Frame(control_frame) util_frame.pack(side=tk.LEFT, expand=True) chart_frame = tk.Frame(control_frame) chart_frame.pack(side=tk.LEFT) center_frame = tk.Frame(control_frame) center_frame.pack(side=tk.LEFT, expand=True) right_frame = tk.Frame(control_frame) right_frame.pack(side=tk.RIGHT, expand=True) table = ttk.Treeview(root, show="headings") table['columns'] = ('ID', 'Name', 'Power', 'Hours', 'Total Energy', 'Price') table.column('ID', anchor=tk.CENTER, width=100) table.column('Name', anchor=tk.CENTER, width=100) table.column('Power', anchor=tk.CENTER, width=100) table.column('Hours', anchor=tk.CENTER, width=100) table.column('Total Energy', anchor=tk.CENTER, width=100) table.column('Price', anchor=tk.CENTER, width=100) table.heading('ID', text='ID', anchor=tk.CENTER) table.heading('Name', text='Name', anchor=tk.CENTER) table.heading('Power', text='Power (W)', anchor=tk.CENTER) table.heading('Hours', text='Hours', anchor=tk.CENTER) table.heading('Total Energy', text='Total Energy (kWh)', anchor=tk.CENTER) table.heading('Price', text='Price (PLN)', anchor=tk.CENTER) table.tag_configure('oddrow', background='#E8E8E8') table.tag_configure('evenrow', background='#FFFFFF') price_label = tk.Label(right_frame, text="Price per kWh") price_var = tk.StringVar() price_var.set("1.52") price_var.trace_add('write', lambda *args: on_price_change(energy_manager, price_var, table, total_price_label)) price = tk.Entry(right_frame,	☑		

			textvariable=price_var, width=20) total_energy_label = tk.Label(center_frame, text="Total Energy (kWh)") total_energy_label.config(text=f"Total Energy: {energy_manager.calculate_total_energy():.2f} kWh") total_price_label = tk.Label(center_frame, text="Total Price (PLN)") total_price_label.config(text=f"Total Price: {energy_manager.calculate_total_energy():.2f} PLN") add_device_button = ttk.Button(util_frame, text="Add Device", command=lambda: show_add_device_window(energy_manager, table, total_energy_label, total_price_label)) load_file_button = ttk.Button(util_frame, text="Load from CSV", command=lambda: show_load_file_window(energy_manager, table, total_energy_label, total_price_label)) save_file_button = ttk.Button(util_frame, text="Save to CSV", command=lambda: show_save_file_window(energy_manager)) create_energy_chart_button = ttk.Button(chart_frame, text="Create energy chart", command=lambda: show_energy_chart(energy_manager)) create_price_chart_button = ttk.Button(chart_frame, text="Create price chart", command=lambda: show_price_chart(energy_manager)) price_label.pack() price.pack() total_energy_label.pack() total_price_label.pack() add_device_button.pack(pady=5) load_file_button.pack(pady=5) save_file_button.pack(pady=5) create_energy_chart_button.pack(pady=5) create_price_chart_button.pack(pady=5) table.pack(expand=True, fill=tk.BOTH) root.mainloop()			
		Wprowadzanie danych	price = tk.Entry(right_frame, textvariable=price_var, width=20)	☑	2	2
		Wyświetlanie danych	total_energy_label = tk.Label(center_frame, text="Total Energy (kWh)") total_energy_label.config(text=f"Total Energy: {energy_manager.calculate_total_energy():.2f} kWh") total_price_label = tk.Label(center_frame, text="Total Price (PLN)") total_price_label.config(text=f"Total Price: {energy_manager.calculate_total_energy():.2f} PLN")	☑	2	2

		Zmiana danych	def on_price_change(energy_manager, price_var, table, total_price_label):
			  try:
			  energy_manager.price = float(price_var.get())
			  reload_table(energy_manager, table)
			  reload_total_price_label(energy_manager, total_price_label)
			  except ValueError:
			  # Jeśli użytkownik usunie zawartość pola (np. backspacem), próba konwersji pustego ciągu na float spowoduje ValueError.
			  # W takim przypadku nic nie robimy i kończymy funkcję.
			  return None
				☑	2	2

		Wyszukiwanie danych	-	☐		2
		Przedstawienie wyników	total_energy_label = tk.Label(center_frame, text="Total Energy (kWh)") total_energy_label.config(text=f"Total Energy: {energy_manager.calculate_total_energy():.2f} kWh") total_price_label = tk.Label(center_frame, text="Total Price (PLN)") total_price_label.config(text=f"Total Price: {energy_manager.calculate_total_energy():.2f} PLN")	☑	2	2
2	Podstawy	Zmienne	total_energy_label = tk.Label(center_frame, text="Total Energy (kWh)") total_energy_label.config(text=f"Total Energy: {energy_manager.calculate_total_energy():.2f} kWh") total_price_label = tk.Label(center_frame, text="Total Price (PLN)") total_price_label.config(text=f"Total Price: {energy_manager.calculate_total_energy():.2f} PLN")	☑	2	2

		Typy danych	def on_price_change(energy_manager, price_var, table, total_price_label):
			  try:
			  energy_manager.price = float(price_var.get())
			  reload_table(energy_manager, table)
			  reload_total_price_label(energy_manager, total_price_label)
			  except ValueError:
			  # Jeśli użytkownik usunie zawartość pola (np. backspacem), próba konwersji pustego ciągu na float spowoduje ValueError.
			  # W takim przypadku nic nie robimy i kończymy funkcję.
			  return None
				☑	2	2

		Komentarze	def on_price_change(energy_manager, price_var, table, total_price_label):
			  try:
			  energy_manager.price = float(price_var.get())
			  reload_table(energy_manager, table)
			  reload_total_price_label(energy_manager, total_price_label)
			  except ValueError:
			  # Jeśli użytkownik usunie zawartość pola (np. backspacem), próba konwersji pustego ciągu na float spowoduje ValueError.
			  # W takim przypadku nic nie robimy i kończymy
				☑	1	1

			funkcję. return None			

		Operatory	def reload_table(energy_manager, table):
			for item in table.get_children():
			table.delete(item)
			
			devices = energy_manager.devices
			
			for i in range(len(devices)):
			tag = 'evenrow' if i % 2 == 0 else 'oddrow'
			table.insert(parent='', index='end', values=(
			i + 1,
			devices[i].name,
			devices[i].power,
			devices[i].hours_per_day,
			devices[i].calculate_energy(),
			
			devices[i].calculate_price(energy_manager.price)
			), tags=(tag,))
				☑	1.5	1.5

		Instrukcje warunkowe (if, elif, else)	def reload_table(energy_manager, table):
			for item in table.get_children():
			table.delete(item)
			
			devices = energy_manager.devices
			
			for i in range(len(devices)):
			tag = 'evenrow' if i % 2 == 0 else 'oddrow'
			table.insert(parent='', index='end', values=(
			i + 1,
			devices[i].name,
			devices[i].power,
			devices[i].hours_per_day,
			devices[i].calculate_energy(),
			
			devices[i].calculate_price(energy_manager.price)
			), tags=(tag,))
				☑	3	3

		Instrukcje interacyjne	def reload_table(energy_manager, table):
			for item in table.get_children():
			table.delete(item)
			
			devices = energy_manager.devices
			
			for i in range(len(devices)):
			tag = 'evenrow' if i % 2 == 0 else 'oddrow'
			table.insert(parent='', index='end', values=(
			i + 1,
			devices[i].name,
			devices[i].power,
			devices[i].hours_per_day,
			devices[i].calculate_energy(),
			
			devices[i].calculate_price(energy_manager.price)
			), tags=(tag,))
				☑		

		For	def reload_table(energy_manager, table):
			for item in table.get_children():
			table.delete(item)
			
			devices = energy_manager.devices
			
			for i in range(len(devices)):
				☑	2	2

			tag = 'evenrow' if i % 2 == 0 else 'oddrow' table.insert(parent='', index='end', values=(i + 1, devices[i].name, devices[i].power, devices[i].hours_per_day, devices[i].calculate_energy(), devices[i].calculate_price(energy_manager.price)), tags=(tag,))			
		While	-	☐		2
		Operacje wejścia (input)	-	☐		1.5
		Operacje wyjścia (print)	-	☐		1.5
		Funkcje z parametrami i wartościami zwracanymi	def calculate_price(self, price): return round(self.calculate_energy() * price, 2)	☑	2	2
		Funkcje rekurencyjne	-	☐		3
		Funkcje przyjmujące inne funkcje jako argumenty	-	☐		3
		Dekoratory	@profile def main():	☑	1.5	1.5

3	Kontenery	Użycie listy	class EnergyManager:
			def __init__(self, price):
			self.devices = []
			self.price = price
				☑	2	2

		Użycie słownika	-	☐		2
		Użycie zbioru	-	☐		1.5
		Użycie krotki	table = ttk.Treeview(root, show="headings") table['columns'] = ('ID', 'Name', 'Power', 'Hours', 'Total Energy', 'Price')	☑	1.5	1.5

4	Przestrzenie nazw	Zastosowano zmienne lokalne	def load_from_csv(self, filename):
			with open(filename, 'r', newline='') as csvfile:
			reader = csv.DictReader(csvfile, delimiter=',')
			for row in reader:
			self.devices.append(Device(row['name'], row['power'], row['hours_per_day']))
				☑		1.5

		Zastosowano zmienne globalne	-	☐		1.5
		Zastosowano zakresy funkcji	-	☐		1.5
		Zastosowan	-	☐		1.5

		o zakresy klas				
5	Moduły i pakiety	Projekt podzielony na moduły (import, __init__)	from charts.energy_chart import show_energy_chart from charts.price_chart import show_price_chart from models.energy_manager import EnergyManager from gui.add_window import show_add_device_window from gui.utils import on_price_change, show_load_file_window, show_save_file_window	☑	2	2
		Własne pakiety/funkcje pomocnicze w osobnych plikach .py	from charts.energy_chart import show_energy_chart from charts.price_chart import show_price_chart from models.energy_manager import EnergyManager from gui.add_window import show_add_device_window from gui.utils import on_price_change, show_load_file_window, show_save_file_window	☑	2	2

6	Obsługa błędów	Obsługa wyjątków (try, except, finally)	def show_price_chart(energy_manager):
			  try:
			  devices = energy_manager.devices
			
			  if len(devices) == 0:
			  raise Exception("No devices found")
			
			  names = [device.name for device in devices]
			  costs = [device.calculate_energy() * energy_manager.price for device in devices]
			
			  plt.figure(figsize=(8, 8))
			  plt.pie(costs, labels=names, autopct='%1.1f%%', startangle=90)
			  plt.title("Full price")
			  plt.axis('equal')
			  plt.show()
			  except Exception as e:
			  messagebox.showerror("Error", str(e))
				☑	2	2

		Użycie assert do testów i walidacji	class Device:
			  def __init__(self, name, power, hours_per_day):
			  self.name = name
			  self.power = int(power)
			  self.hours_per_day = float(hours_per_day)
			  assert 0 < self.hours_per_day <= 24, "Hours must be a float value between 0 and 24"
			  assert 0 < self.power <= 10000, "Power must be a float value between 0 and 10000"
				☑	1.5	1.5

7	Łańcuchy znaków	Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie)	total_energy_label = tk.Label(center_frame, text="Total Energy (kWh)") total_energy_label.config(text=f"Total Energy: {energy_manager.calculate_total_energy():.2f} kWh") total_price_label = tk.Label(center_frame, text="Total Price (PLN)") total_price_label.config(text=f"Total Price: {energy_manager.calculate_total_energy():.2f} PLN")	☑	2	2
8	Obsługa plików	Odczyt z plików .txt, .	def load_from_csv(self, filename): with open(filename, 'r', newline='') as csvfile:	☑	2	2

		csv, .json, .xml (min. 1)	reader = csv.DictReader(csvfile, delimiter=',') for row in reader: self.devices.append(Device(row['name'], row['power'], row['hours_per_day']))			

		Zapis do plików .txt, .csv, .json, .xml (min. 1)	def save_to_csv(self, filename):
			with open(filename, 'w', newline='') as csvfile:
			writer = csv.DictWriter(csvfile,
			fieldnames=['id', 'name', 'power', 'hours_per_day', 'total_energy', 'price'])
			writer.writeheader()
			for device in self.devices:
			writer.writerow({
			'id': self.devices.index(device) + 1,
			'name': device.name,
			'power': device.power,
			'hours_per_day': device.hours_per_day,
			'total_energy': device.calculate_energy(),
			'price': device.calculate_price(self.price)
			})
				☑	2	2

9	OOP	Klasy	class EnergyManager:
			def __init__(self, price):
			self.devices = []
			self.price = price
			
			def add_device(self, name, power, hours_per_day):
			self.devices.append(Device(name, power, hours_per_day))
			
			def load_from_csv(self, filename):
			with open(filename, 'r', newline='') as csvfile:
			reader = csv.DictReader(csvfile, delimiter=',')
			for row in reader:
			self.devices.append(Device(row['name'], row['power'], row['hours_per_day']))
			
			def save_to_csv(self, filename):
			with open(filename, 'w', newline='') as csvfile:
			writer = csv.DictWriter(csvfile,
			fieldnames=['id', 'name', 'power', 'hours_per_day', 'total_energy', 'price'])
			writer.writeheader()
			for device in self.devices:
			writer.writerow({
			'id': self.devices.index(device) + 1,
			'name': device.name,
			'power': device.power,
			'hours_per_day': device.hours_per_day,
			'total_energy': device.calculate_energy(),
			'price': device.calculate_price(self.price)
			})
			
			def calculate_total_energy(self):
			return reduce(lambda energy, device: energy + device.calculate_energy(), self.devices, 0)
			
			def calculate_total_price(self):
			return reduce(lambda price, device: price + device.calculate_price(self.price), self.devices, 0)
				☑	2	2

		Metody	def calculate_total_energy(self): return reduce(lambda energy, device: energy + device.calculate_energy(), self.devices, 0)	☑	2	2

			def calculate_total_price(self): return reduce(lambda price, device: price + device.calculate_price(self.price), self.devices, 0)			

		Konstruktory	class EnergyManager:
			  def __init__(self, price):
			  self.devices = []
			  self.price = price
				☑	2	2

		Dziedziczenie	class TestDevice(unittest.TestCase):
			  def test_calculate_energy(self):
			  device = Device("Fridge", 100, 24)
			  self.assertEqual(device.calculate_energy(), 2.4)
			
			  def test_calculate_price(self):
			  device = Device("Fridge", 100, 24)
			  self.assertEqual(device.calculate_price(1.52), 3.65)
			
			  def test_invalid_hours(self):
			  self.assertRaises(AssertionError, Device, "Fridge", 100, 25)
			
			  def test_invalid_power(self):
			  self.assertRaises(AssertionError, Device, "Fridge", 10001, 24)
			
			if __name__ == '__main__':
			  unittest.main()
				☑	2	2

10	Programowanie funkcyjne	map	-	☐		1.5
		filter	-	☐		1.5

		lambda	def calculate_total_energy(self):
			  return reduce(lambda energy, device: energy + device.calculate_energy(), self.devices, 0)
			
			def calculate_total_price(self):
			  return reduce(lambda price, device: price + device.calculate_price(self.price), self.devices, 0)
				☑	1.5	1.5

		reduce	def calculate_total_energy(self):
			  return reduce(lambda energy, device: energy + device.calculate_energy(), self.devices, 0)
			
			def calculate_total_price(self):
			  return reduce(lambda price, device: price + device.calculate_price(self.price), self.devices, 0)
				☑	1.5	1.5

11	Wizualizacja danych	Wygenerowano wykres (np. matplotlib, seaborn)	def show_price_chart(energy_manager):
			  try:
			  devices = energy_manager.devices
			
			  if len(devices) == 0:
			  raise Exception("No devices found")
			
			  names = [device.name for device in devices]
			  costs = [device.calculate_energy() * energy_manager.price for device in devices]
			
			  plt.figure(figsize=(8, 8))
			  plt.pie(costs, labels=names, autopct='%1.1f%%', startangle=90)
			  plt.title("Full price")
			  plt.axis('equal')
				☑	2	2

			plt.show() except Exception as e: messagebox.showerror("Error", str(e))			
		Zapisano wykres do pliku graficznego (.png, .jpg)	Zapis do pliku zrealizowany przez matplotlib	☑	1.5	1.5

12	Testowanie	Testy jednostkowe (assert, unittest, pytest)	class TestDevice(unittest.TestCase):
			def test_calculate_energy(self):
			device = Device("Fridge", 100, 24)
			self.assertEqual(device.calculate_energy(), 2.4)
			
			def test_calculate_price(self):
			device = Device("Fridge", 100, 24)
			self.assertEqual(device.calculate_price(1.52), 3.65)
			
			def test_invalid_hours(self):
			self.assertRaises(AssertionError, Device, "Fridge", 100, 25)
			
			def test_invalid_power(self):
			self.assertRaises(AssertionError, Device, "Fridge", 10001, 24)
			
			if __name__ == '__main__':
			unittest.main()
				☑	1.5	1.5

		Testy funkcjonalne	-	☐		1.5
		Testy integracyjne	-	☐		1.5

		Testy graniczne / błędne dane	def test_invalid_hours(self):
			self.assertRaises(AssertionError, Device, "Fridge", 100, 25)
			
			def test_invalid_power(self):
			self.assertRaises(AssertionError, Device, "Fridge", 10001, 24)
				☑	1.5	1.5

		Testy wydajności (np. czas wykonania, timeit)	-	☐		1.5
		Testy pamięci memory_profiler	~/Projects/energy-manager main ❯ python -m memory_profiler main.py Filename: main.py Line # Mem usage Increment Occurrences Line Contents === 12 74.3 MiB 74.3 MiB 1 @profile 13 def main(): 14 74.3 MiB 0.0 MiB 1 energy_manager = EnergyManager(1.52)	☑	1.5	1.5
		Test jakości kodu	~/Projects/energy-manager main ❯ flake8 *	☑	1.5	1.5

		(flake8, pylint)				
13	Wersjonowanie	Repozytorium GIT	https://github.com/mikolaj-pacierz-psk/energy-manager	☑	1	1
		Historia commitów	commit c8d6b75c3e3ba647793b90bf5a1ec1035b67ad85 (HEAD -> main , origin/main, origin/HEAD) Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 19:44:43 2025 +0200 misc: code refactoring commit 63a9802102c6d18b87d1e24f38e4593c032a4709 Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 19:25:39 2025 +0200 misc: modified README commit 56921c79f936b1b4853f4e9d660168253ca5400b Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 15:39:07 2025 +0200 misc: modified README commit 6d2144ec8bcf399a5188d91278f7e73230d3892e Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 15:09:02 2025 +0200 misc: fixed executable file commit af63ddf8c46a71b410532fd37c63257bcc0c7d5b Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 12:09:35 2025 +0200 misc: refactoring	☑	1	1
		Link do GitHub	https://github.com/mikolaj-pacierz-psk/energy-manager	☑	1	1
		Opis commitów	commit c8d6b75c3e3ba647793b90bf5a1ec1035b67ad85 (HEAD -> main , origin/main, origin/HEAD) Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 19:44:43 2025 +0200 misc: code refactoring commit 63a9802102c6d18b87d1e24f38e4593c032a4709 Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 19:25:39 2025 +0200 misc: modified README	☑	1	1

			commit 56921c79f936b1b4853f4e9d660168253ca5400b Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 15:39:07 2025 +0200 misc: modified README commit 6d2144ec8bcf399a5188d91278f7e73230d3892e Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 15:09:02 2025 +0200 misc: fixed executable file commit af63ddf8c46a71b410532fd37c63257bcc0c7d5b Author: Mikołaj Pacierz <mikolajpacierz18@gmail.com> Date: Wed Jun 25 12:09:35 2025 +0200 misc: refactoring			
14	Dokumentacja	Plik README.md (cel, autorzy, uruchamianie)	# Energy Manager ## Cel aplikacji - Celem aplikacji była możliwość zarządzania zużyciem energii w domu, poprzez obliczanie zużycia energii przez urządzenia i obliczanie ceny każdego urządzenia ## Technologie - Python 3.11 - pip 25.0.1 ## Uruchomienie aplikacji - Windows ``` git clone https://github.com/mikolaj-pacierz-psk/energy-manager.git cd energy-manager python -m venv .venv venv\Scripts\activate pip install -r requirements.txt python main.py ``` - Linux ``` git clone https://github.com/mikolaj-pacierz-psk/energy-manager.git cd energy-manager python -m venv .venv source .venv/bin/activate pip install -r requirements.txt python main.py ``` *## Autorzy:* - Mikołaj Pacierz	☑	1.5	1.5

		Przykładowe dane wejściowe i wyjściowe	*Zawartość pliku wejściowego devices.csv:* id,name,power,hours_per_day 1,Refrigerator,150,24 2,Television,100,5 3,Laptop,65,8 4,Washing Machine,500,1 5,Microwave,1200,0.5 6,Ceiling Fan,75,10 7,Air Conditioner,1500,6 8,Light Bulb,10,6 9,Toaster,800,0.25 10,Water Heater,3000,1 *Zawartość pliku wyjściowego devices_output.csv:* id,name,power,hours_per_day,total_energy,price 1,Refrigerator,150,24.0,3.6,5.47 2,Television,100,5.0,0.5,0.76 3,Laptop,65,8.0,0.52,0.79 4,Washing Machine,500,1.0,0.5,0.76 5,Microwave,1200,0.5,0.6,0.91 6,Ceiling Fan,75,10.0,0.75,1.14 7,Air Conditioner,1500,6.0,9.0,13.68 8,Light Bulb,10,6.0,0.06,0.09 9,Toaster,800,0.25,0.2,0.3 10,Water Heater,3000,1.0,3.0,4.56	☑	2	2
		Diagram klas lub struktura modułów	energy_manager_app/ ├── main.py ├── models/ │ ├── device.py │ ├── energy_manager.py │ └── __init__.py ├── charts/ │ ├── energy_chart.py │ ├── price_chart.py │ └── __init__.py ├── gui/ │ ├── add_window.py │ ├── utils.py │ └── __init__.py ├── data/ │ └── devices.csv ├── output/ │ └── devices_output.csv ├── requirements.txt ├── setup.cfg └── README.md	☑	2	2
				SUMA		70	100