

2810ICT Software Projects

Part 1: Initiation and Outline

1.1 Project Charter

Section	Example
Project Name	XPower Household Tariff Analysis Software
Purpose	To assist customers by calculating and comparing costs to achieve the best plan
Project Manager	Fred (Team Leader)
Main Sponsor	XPower
High-Level Requirements	<ul style="list-style-type: none">· Allow data import of electricity consumption· Develop different tariff models· Calculate and compare bill breakdown under differing tariff schemes· Visualize and report electricity usage and bill breakdown· Website UI
Budget	\$10,000
Timeline/Milestones	<p>Project Start: 6 Oct</p> <p>Back-End Services Complete: 17 October</p> <p>Front-End Services Complete: 31 October</p> <p>Testing and Adjustment Complete: 7 November</p> <p>Final Delivery: 10 November</p>

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

High-Level Risks	Incorrect implementation of tariffs, incorrect calculation of bills, delay of UI delivery
Approval	Signed by sponsor: Dr. Smith, XPower CEO

1.2 Identify Stakeholders

XPower

Role: Provides budget and approves the project

Interest: Project delivered on time and within budget, meets requirements set

Influence: Determines budget and time constraints, has final say over product

Development Team

Role: Develop UI and software of project

Interest: Build functional software that meets requirements

Influence: Quality and delivery of software

End Users

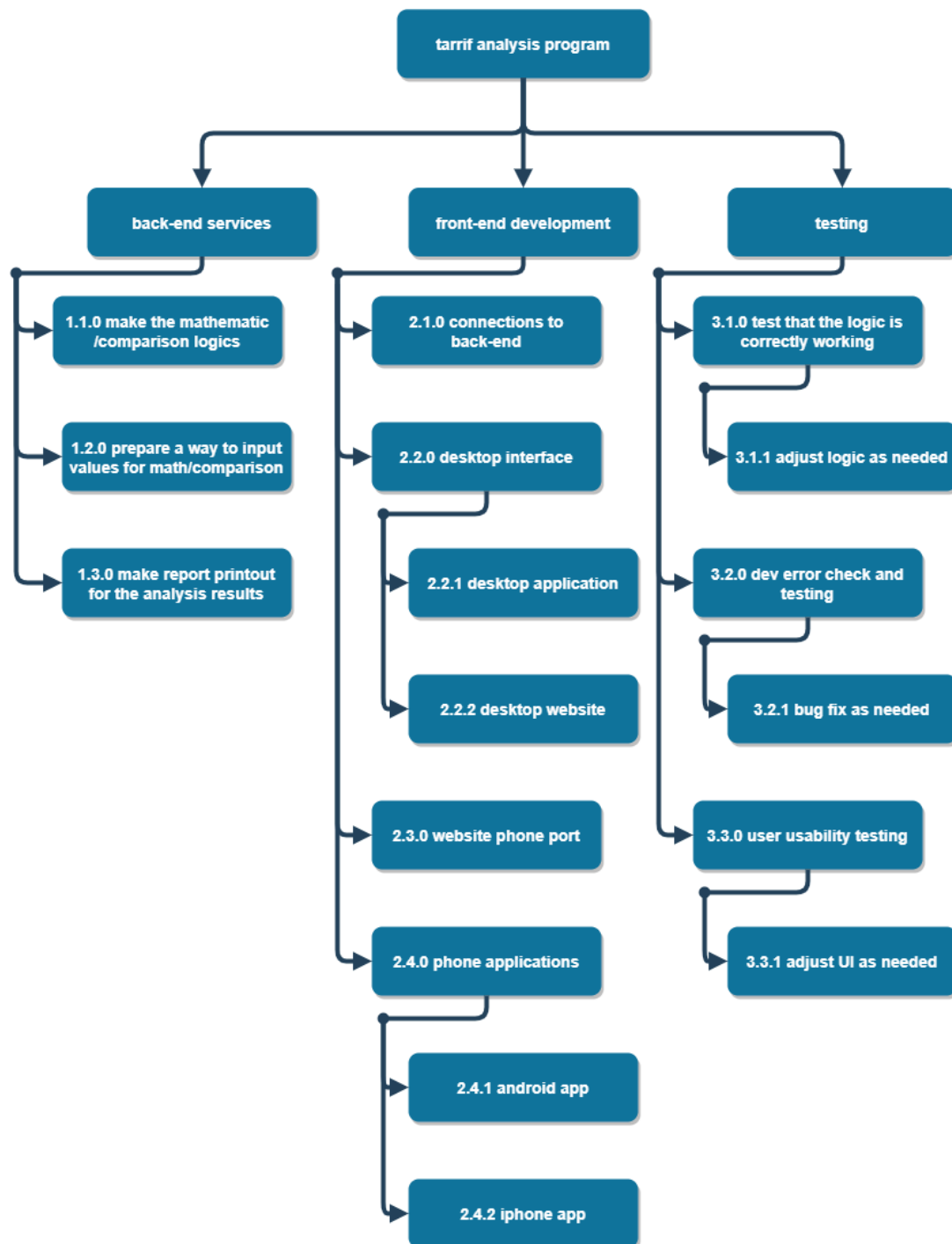
Role: Users of the software

Interest: Want easy-to-use software, with user-friendly UI

Influence: Provide feedback on software from users' perspective

Part 2: planning the work space breakdown, schedule, and metrics

2.1.1 - WBS Diagram



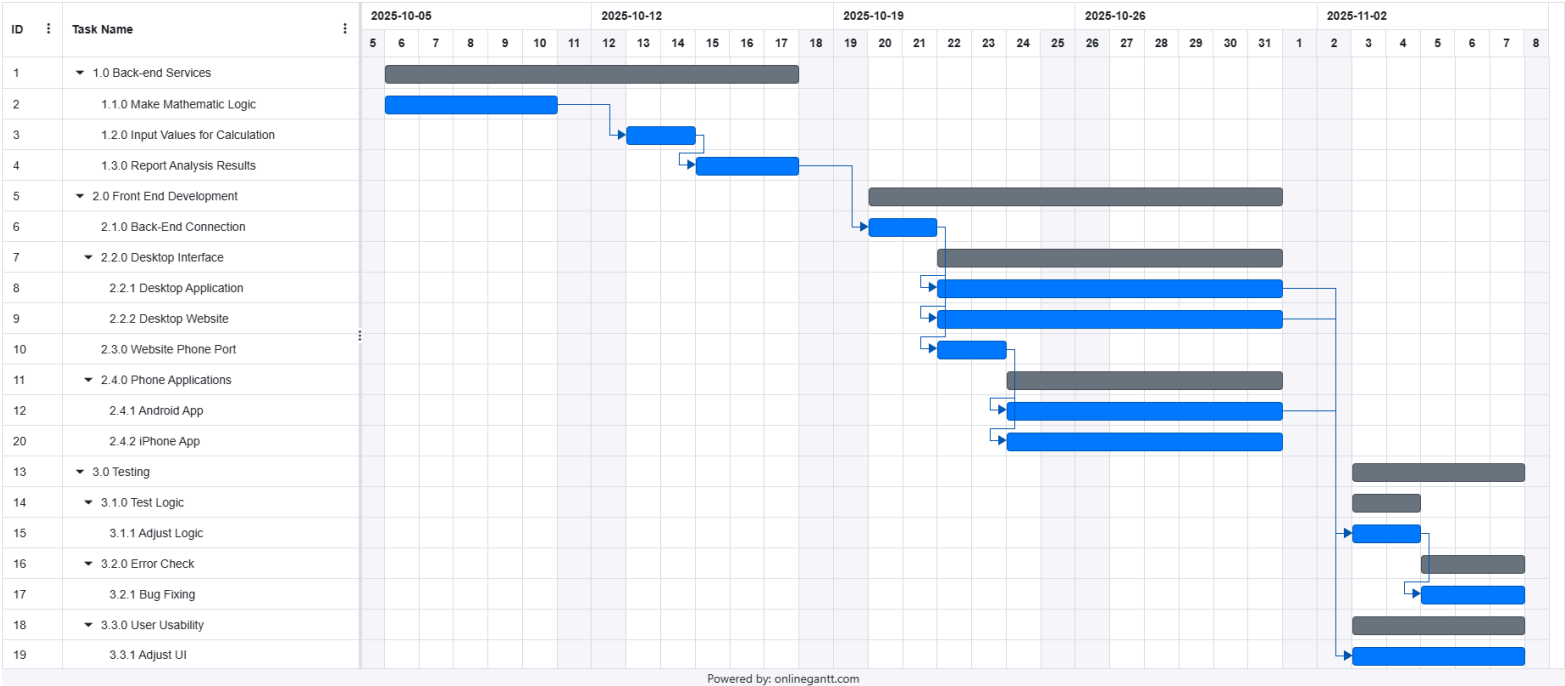
2.1.2 - WBS dictionary

Activity ID	Activity Name	Description	Dependencies	Duration
1.2.0	Prepare a way to input values	Set up a way for the front-end devs to send the user imputed	N/A	2 days

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

		values to the analysis program		
2.2.1	Desktop application	Create the desktop application that will serve as the base/reference version of the program	2.1.0	8 days
2.3.0	Phone website port	Make a resized/rescaled version of the website for mobile users	2.2.2	2 days
3.1.0	Test that the logic works correctly	Use inputs with known outputs to tune the logic and then perform follow-up tests with new values to make sure it works on a variety of data	2.2.1 or 2.2.2	2 days
3.3.0	User suitability testing	Let some people test out the program to see if they can easily navigate it and use it accurately	Any 2.2.x and any 2.4.x	5 days

2.2.1 Project Schedule



2.3.1 - Risk Register

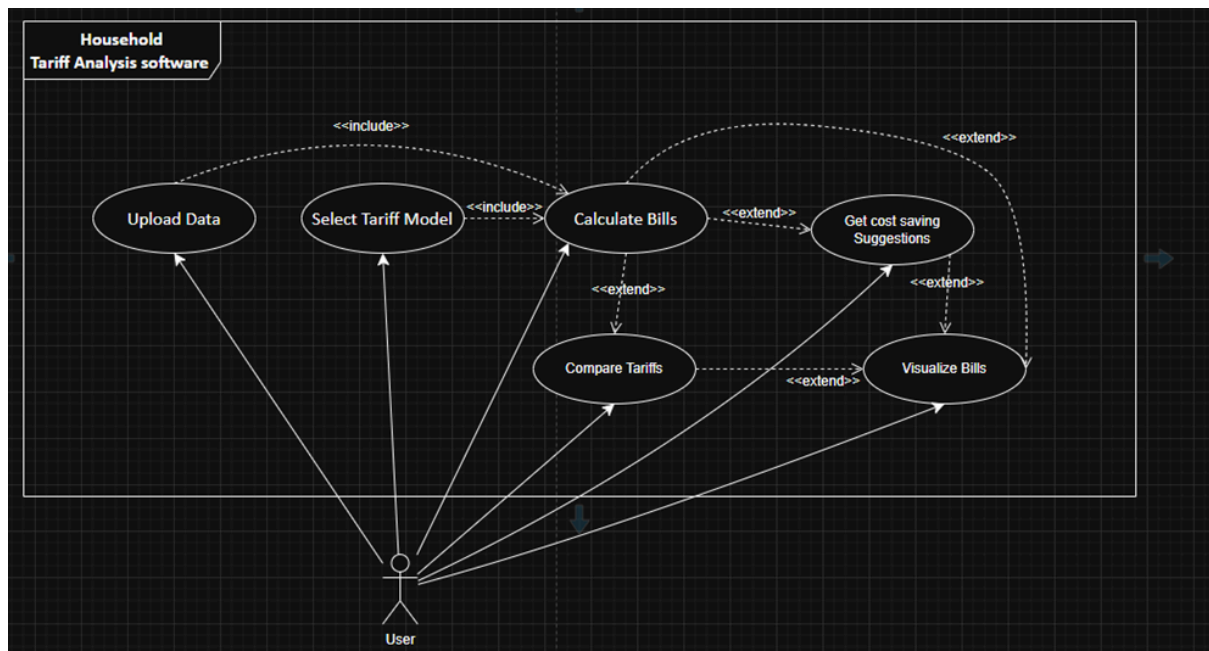
Risk ID	Risk Description	Category	Impact
R-01	Unsupported file uploaded for consumption data	Technical Risk	Software will not run with unsupported file, requiring a change in file type
R-02	Project development exceeds given budget	Cost Risk	Sponsors will need to allocate an increased budget, or stop project
R-03	Project development exceeds time constraints	Schedule Risk	Rollout of software will be delayed, can lead to financial loss
R-04	Project does not meet regulation laws	Compliance Risk	Lawsuit may occur leading to financial loss, or project can stop

2.4.1 - Quality Metrics

Metric Name	Definition	Measurement Method	Acceptance Criteria
UI Layout	How user-friendly and well organizer the UI layout is	Analyzing reviews made on the UI	Above 90% positive reviews on UI layout
File Upload Speed	Measures the average speed to import data	Analyze testing report for average upload speed	Data is uploaded in under 20 seconds
Error Rate	Number of errors occurred during testing	Analyze testing reports for encountered errors	Under 5 errors for every 10,000 operations
Tariff Calculation Accuracy	Tariff calculation software outputs correct result	Manually compare tariff software calculation to expected outputs	All software calculations return the expected output

Part 3: Execution and Coding

3.1.1 - Use Case Diagram



3.1.2 - Use Case Description

Use Case ID	UC 01
Use Case Name	Uploading data
Actors	User
Description	The user uploads their household electricity use
Preconditions	None
Trigger	User selects a file to upload
Flow of events	<ol style="list-style-type: none">1. User selects a data file to be uploaded2. System reads the file data3. System parses the file data4. System uploads the file data

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

Alternate flow	<ol style="list-style-type: none">1. Incorrect file format2. Requests that the user enters the correct format (CSV or EXCEL)
Post conditions	Success message and the user can now use the calculate bill's function
Exceptions	Incorrect file format. Activates Alternate flow

Use Case ID	UC 02
Use Case Name	Calculate Bills
Actors	User
Description	The user can use uploaded data to calculate their new bills
Preconditions	User must have uploaded data
Trigger	User selects the calculate bills function
Flow of events	<ol style="list-style-type: none">1. User selects the tariff model2. Function returns the calculated bills using the tariff model
Alternate flow	None
Post conditions	System displays calculated results
Exceptions	System encounters corrupted data. Displays an error and returns user to the upload data page

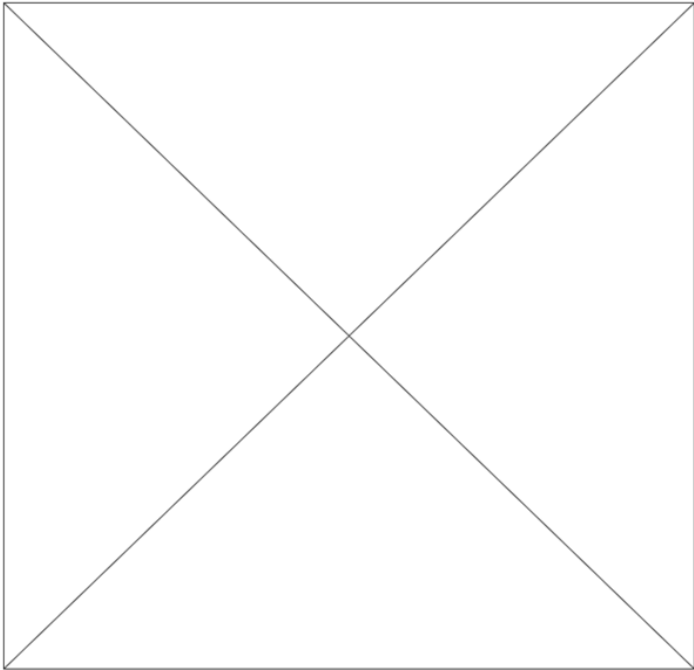
Use Case ID	UC 03
Use Case Name	Compare Tariffs
Actors	User
Description	The user can compare a different tariff scheme to their current one
Preconditions	User must have uploaded data
Trigger	User selects the calculate bills function with multiple tariff options selected
Flow of events	<ol style="list-style-type: none">1. User selects the tariff models to compare2. System calculates and returns the values for each model
Alternate flow	None
Post conditions	System displays the calculated results of each tariff model
Exceptions	System encounters corrupted data. Displays an error and returns user to the upload data page

Use Case ID	UC 04
Use Case Name	Visualize bills
Actors	User
Description	The user can visualize the bills using line chars and bar charts or pie charts
Preconditions	User must have uploaded or calculated data

Trigger	User selects the visualize data option
Flow of events	1. System displays line chat of electricity usage
Alternate flow	1. System displays line chat of electricity usage 2. System displays bar or pie charts for calculated data
Post conditions	System displays charts displaying the electricity usage and calculated bills
Exceptions	No data uploaded. Return an error and take the user to the upload data page

3.1.3 - High-Level Architecture Design:

XPower - Household Tariff Analysis



Select Tariff Option

Calculated Bills

Lorem ipsum dolor sit amet et delectus accommodare his consul copiosae legendos at vix ad putent delectus delicata usu. Vedit dissentiet eos cu

Cost Saving Suggestions

Lorem ipsum dolor sit amet et delectus accommodare his consul copiosae legendos at vix ad putent delectus delicata usu. Vedit dissentiet eos cu eum an brute copiosae

3.1.4 - Coding

Code Function 1 flat rate tariff calculator:

```
import pandas as pd
```

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
df_csv = pd.read_csv('../data/sample_usage_data_month.csv')

def calc_flat_rate_tariff(data, rate, fee):
    try:
        if not (isinstance(rate, (int, float)) and isinstance(fee,
(int, float))):
            raise TypeError

        if rate <= 0 or fee < 0:
            raise ValueError

        total = fee
        for kWh in data['kWh']:
            total += kWh * rate

        return round(total, 4)

    except TypeError:
        return 'Type Error'

    except ValueError:
        return 'Value Error'

print(calc_flat_rate_tariff(df_csv, 0.25, 10.36))
```

Code Function 2 tiered rate tariff calculator:

```
# tierRates[i] = tier, tierRates[i][0] = threshold, tierRates[i][1] =
rate
def calc_tiered_rate_tariff(data, tieredRates, fee):
    try:
        if not (isinstance(data, (int, float)) and isinstance(fee,
(int, float))):
            raise TypeError

        if data <= 0 or fee < 0:
            raise ValueError

        for i in range(len(tieredRates)):

            if not (isinstance(tieredRates[i][0], (int, float)) and
isinstance(tieredRates[i][1], (int, float))):
```

```
        raise TypeError

    if tieredRates[i][0] <= 0 or tieredRates[i][1] <= 0:
        raise ValueError

    # check if the last threshold is inf
    if i == len(tieredRates) - 1 and tieredRates[i][0] !=
float('inf'):
        raise ValueError

    total = fee
    remaining = data
    costBreakdown= dict()

    for tier in tieredRates:
        if remaining > tier[0]:
            consumption = tier[0]
        else:
            consumption = remaining

        costBreakdown[tier[0]] = consumption * tier[1]

        total += costBreakdown[tier[0]]
        remaining -= consumption

        if remaining <= 0:
            break
    costBreakdown['total'] = round(total, 4)
    return costBreakdown

except TypeError:
    return 'Type Error'

except ValueError:
    return 'Value Error'

tieredRates = [ (100, 0.2), (300, 0.30), (float('inf'), 0.40) ]
costBreakdown = calc_tiered_rate_tariff(850, tieredRates, 10)
```

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
print("Tiered Tariff Bill Breakdown:")
print(costBreakdown)
```

Code function 3 time of use tariff calculator:

```
# tierRates[i] = tier, tierRates[i][0] = threshold, tierRates[i][1] =
rate
def calc_tiered_rate_tariff(data, tieredRates, fee):
    try:
        if not (isinstance(data, (int, float)) and isinstance(fee,
(int, float))):
            raise TypeError

        if data <= 0 or fee < 0:
            raise ValueError

        for i in range(len(tieredRates)):

            if not (isinstance(tieredRates[i][0], (int, float)) and
isinstance(tieredRates[i][1], (int, float))):
                raise TypeError

            if tieredRates[i][0] <= 0 or tieredRates[i][1] <= 0:
                raise ValueError

            # check if the last threshold is inf
            if i == len(tieredRates) - 1 and tieredRates[i][0] !=
float('inf'):
                raise ValueError

            total = fee
            remaining = data
            costBreakdown= dict()

            for tier in tieredRates:
                if remaining > tier[0]:
                    consumption = tier[0]
                else:
                    consumption = remaining

                costBreakdown[tier[0]] = consumption * tier[1]

            total += costBreakdown[tier[0]]
```

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
        remaining -= consumption

        if remaining <= 0:
            break
    costBreakdown['total'] = round(total, 4)
    return costBreakdown

except TypeError:
    return 'Type Error'

except ValueError:
    return 'Value Error'

tieredRates = [ (100, 0.2), (300, 0.30), (float('inf'), 0.40) ]
costBreakdown = calc_tiered_rate_tariff(850, tieredRates, 10)

print("Tiered Tariff Bill Breakdown:")
print(costBreakdown)
```

Code Function 4 suggested tariff:

```
from falt_rate_tariff.flat_rate_tariff import calc_flat_rate_tariff
from tiered_tariff.tiered_tariff import calc_tiered_rate_tariff
from time_of_use.time_of_use_tariff import calc_time_of_use_tariff
import pandas as pd

csv = pd.read_csv('./data/sample_usage_data_month.csv')

def suggested_tariff(data, kWh, flatRate, tieredRates, peakRate,
offPeakRate, shoulderRate, peakTime, offPeakTime, fee):
    try:
        flatRate = calc_flat_rate_tariff(data, flatRate, fee)
        tieredRate = calc_tiered_rate_tariff(kWh, tieredRates, fee)
        timeOfUse = calc_time_of_use_tariff(data, peakRate,
offPeakRate, shoulderRate, peakTime, offPeakTime, fee)

        if flatRate == 'Type Error' or tieredRate == 'Type Error' or
timeOfUse == 'Type Error':
            raise TypeError
```

```
        if flatRate == 'Value Error' or tieredRate == 'Value Error' or
timeOfUse == 'Value Error':
            raise ValueError
        lowest = min(flatRate, tieredRate['total'], timeOfUse)

        if lowest == flatRate:
            return 'Flat Rate'
        elif lowest == tieredRate['total']:
            return 'Tiered Rate'
        else:
            return 'Time Of Use'

    except TypeError:
        return 'Type Error'
    except ValueError:
        return 'Value Error'

# flat rate data, rate, fee
# tiered rate int, tierRates, fee
# time of use data, peakRate, offPeakRate, shoulderRate, peakTime,
offPeakTime, fee

kWh = 850
flatRate = 0.2
tieredRates = [ (100, 0.20), (200, 0.30), (float('inf'), 0.40) ]
peakRate = 0.25
offPeakRate = 0.4
shoulderRate = 0.15
peakTime = ('18:00:00', '22:00:00')
offPeakTime = ('22:00:00', '7:00:00')
fee = 10
reccomended = suggested_tariff(csv, kWh, flatRate, tieredRates,
peakRate, offPeakRate, shoulderRate, peakTime, offPeakTime, fee)
print(reccomended)
```

3.2.1 - Unit Testing

Test 1 falt rate tariff:

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
Coverage for flat_rate_tariff.py: 100%
14 statements  14 run  0 missing  0 excluded
« prev  ^ index  » next  coverage.py v7.6.9, created at 2025-09-27 14:06 +1000

1  # import pandas as pd
2  # df_csv = pd.read_csv('../data/sample_usage_data_month.csv')
3
4  def calc_flat_rate_tariff(data, rate, fee):
5      try:
6          if not (isinstance(rate, (int, float)) and isinstance(fee, (int, float))):
7              raise TypeError
8
9          if rate <= 0 or fee < 0:
10             raise ValueError
11
12             total = fee
13             for kWh in data['kWh']:
14                 total += kWh * rate
15
16             return round(total, 4)
17
18     except TypeError:
19         return 'Type Error'
20
21     except ValueError:
22         return 'Value Error'
23
24
25
26 # print(calc_flat_rate_tariff(df_csv, 0.25, 10.36))
```

Test 2 tiered use tariff:

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

Coverage for `tiered_tariff.py`: 100%

31 statements

31 run

0 missing

0 excluded

« prev ^ index » next coverage.py v7.6.9, created at 2025-09-27 14:04 +1000

```
1 # tierRates[i] = tier, tierRates[i][0] = threshold, tierRates[i][1] = rate
2 def calc_tiered_rate_tariff(data, tieredRates, fee):
3     try:
4         if not (isinstance(data, (int, float)) and isinstance(fee, (int, float))):
5             raise TypeError
6
7         if data <= 0 or fee < 0:
8             raise ValueError
9
10        for i in range(len(tieredRates)):
11
12            if not (isinstance(tieredRates[i][0], (int, float)) and isinstance(tieredRates[i][1], (int, float))):
13                raise TypeError
14
15            if tieredRates[i][0] <= 0 or tieredRates[i][1] <= 0:
16                raise ValueError
17
18            # check if the last threshold is inf
19            if i == len(tieredRates) - 1 and tieredRates[i][0] != float('inf'):
20                raise ValueError
21
22        total = fee
23        remaining = data
24        costBreakdown= dict()
25
26        for tier in tieredRates:
27            if remaining > tier[0]:
28                consumption = tier[0]
29            else:
30                consumption = remaining
31
32            costBreakdown[tier[0]] = consumption * tier[1]
33
34            total += costBreakdown[tier[0]]
35            remaining -= consumption
36
37            if remaining <= 0:
38                break
39        costBreakdown['total'] = round(total, 4)
40        return costBreakdown
41
42    except TypeError:
43        return 'Type Error'
44
45    except ValueError:
46        return 'Value Error'
47
48
49
50
51
52 # tieredRates = [ (100, 0.2), (300, 0.30), (float('inf'), 0.40) ]
53 # costBreakdown = calc_tiered_rate_tariff(850, tieredRates, 10)
54
55 # print("Tiered Tariff Bill Breakdown:")
56 # print(costBreakdown)
```

Test 3 time of use tariff:

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

Coverage for `time_of_use_tariff.py`: 100%

49 statements 49 run 0 missing 0 excluded

← prev ^ index → next coverage.py v7.6.9, created at 2025-09-27 15:13 +1000

```
import pandas as pd
from datetime import datetime

# Reading CSV file
df_csv = pd.read_csv('./data/sample_usage_data_month.csv')

def is_time_in_range(timeToCheck, startTime, endTime):

    if startTime > endTime:
        return timeToCheck >= startTime or timeToCheck <= endTime
    else:
        return startTime <= timeToCheck <= endTime

# Forwards are
peakTime[0] = earliest peak time, peakTime[1] = latest peak time
offPeak[0] = earliest offPeak time, offPeak[1] = latest offPeak time
record[0] = time stamp, record[1] = kWh
# timestamp is a tuple of (date, time)
def calc_time_of_use_tariff(data, peakRate, offPeakRate, shoulderRate, peakTime, offPeakTime, fee):
    try:

        if not (isinstance(peakRate, (int, float)) and isinstance(offPeakRate, (int, float)) and isinstance(shoulderRate, (int, float)) and isinstance(fee, (int, float))):
            raise TypeError

        if peakRate <= 0 or offPeakRate <= 0 or shoulderRate <= 0 or fee <= 0:
            raise ValueError

        # error checking for peak time
        for item in peakTime:
            parts = item.split(':')

            if len(parts) != 3:
                raise TypeError

            if not (parts[0].isdigit() and parts[1].isdigit() and parts[2].isdigit()):
                raise TypeError

            if not (0 <= int(parts[0]) <= 23):
                raise ValueError

            if not ((0 <= int(parts[1]) < 60) and (0 <= int(parts[2]) < 60)):
                raise ValueError

        # error checking for off peak time
        for item in offPeakTime:
            parts = item.split(':')

            if len(parts) != 3:
                raise TypeError

            if not (parts[0].isdigit() and parts[1].isdigit() and parts[2].isdigit()):
                raise TypeError

            if not (0 <= int(parts[0]) <= 23):
                raise ValueError

            if not ((0 <= int(parts[1]) < 60) and (0 <= int(parts[2]) < 60)):
                raise ValueError

        total = fee
        peakTime = (
            datetime.strptime(peakTime[0], '%H:%M:%S'),
            datetime.strptime(peakTime[1], '%H:%M:%S')
        )

        offPeakTime = (
            datetime.strptime(offPeakTime[0], '%H:%M:%S'),
            datetime.strptime(offPeakTime[1], '%H:%M:%S')
        )

        timestamp = []
        for i in range(len(data['timestamp'])):
            timestamp.append((datetime.strptime(data['timestamp'][i], '%Y-%m-%d %H:%M:%S'), data['kWh'][i]))

        for record in timestamp:
            if is_time_in_range(record[0].time(), peakTime[0].time(), peakTime[1].time()):
                total += record[1] * peakRate
            elif is_time_in_range(record[0].time(), offPeakTime[0].time(), offPeakTime[1].time()):
                total += record[1] * offPeakRate
            else:
                total += record[1] * shoulderRate
        return round(total, 4)
    except TypeError:
        return 'Type Error'
    except ValueError:
        return 'Value Error'

# total = calc_time_of_use_tariff(df_csv, 0.25, 0.4, 0.15, ('19:00:00', '22:00:00'), ('22:00:00', '7:00:00'), 10)
# print(total)
```

Test 4 suggested tariff:

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
Coverage for suggest_tariff.py: 100%
22 statements 22 run 0 missing 0 excluded
« prev ^ index » next coverage.py v7.6.9, created at 2025-09-27 15:41 +1000

1 from falt_rate_tariff.flat_rate_tariff import calc_flat_rate_tariff
2 from tiered_tariff.tiered_tariff import calc_tiered_rate_tariff
3 from time_of_use.time_of_use_tariff import calc_time_of_use_tariff
4 # import pandas as pd
5
6 # csv = pd.read_csv('./data/sample_usage_data_month.csv')
7
8 def suggested_tariff(data, kWh, flatRate, tieredRates, peakRate, offPeakRate, shoulderRate, peakTime, offPeakTime, fee):
9     try:
10         flatRate = calc_flat_rate_tariff(data, flatRate, fee)
11         tieredRate = calc_tiered_rate_tariff(kWh, tieredRates, fee)
12         timeOfUse = calc_time_of_use_tariff(data, peakRate, offPeakRate, shoulderRate, peakTime, offPeakTime, fee)
13
14         if flatRate == 'Type Error' or tieredRate == 'Type Error' or timeOfUse == 'Type Error':
15             raise TypeError
16
17         if flatRate == 'Value Error' or tieredRate == 'Value Error' or timeOfUse == 'Value Error':
18             raise ValueError
19         lowest = min(flatRate, tieredRate['total'], timeOfUse)
20
21         if lowest == flatRate:
22             return 'Flat Rate'
23         elif lowest == tieredRate['total']:
24             return 'Tiered Rate'
25         else:
26             return 'Time Of Use'
27
28     except TypeError:
29         return 'Type Error'
30     except ValueError:
31         return 'Value Error'
32
33
34
35 # flat rate data, rate, fee
36 # tiered rate int, tierRates, fee
37 # time of use data, peakRate, offPeakRate, shoulderRate, peakTime, offPeakTime, fee
38
39
40 # kWh = 850
41 # flatRate = 0.2
42 # tieredRates = [ (100, 0.20), (200, 0.30), (float('inf'), 0.40) ]
43 # peakRate = 0.25
44 # offPeakRate = 0.4
45 # shoulderRate = 0.15
46 # peakTime = ('18:00:00', '22:00:00')
47 # offPeakTime = ('22:00:00', '7:00:00')
48 # fee = 10
49 # recommended = suggested_tariff(csv, kWh, flatRate, tieredRates, peakRate, offPeakRate, shoulderRate, peakTime, offPeakTime, fee)
50 # print(recommended)
```

3.2.2 - Code for Unit Testing

Code for test 1 flat rate tariff:

```
from flat_rate_tariff import calc_flat_rate_tariff

import pandas as pd
data = pd.read_csv('./data/sample_usage_data_month.csv')

def test_flate_rate_tariff_typical():
    assert calc_flat_rate_tariff(data, 0.25, 10) == 222.6675

def test_flate_rate_tariff_type_error_fee():
```

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
    assert calc_flat_rate_tariff(data, 0.25, '0') == 'Type Error'

def test_flat_rate_tariff_type_error_rate():
    assert calc_flat_rate_tariff(data, '0.25', 0) == 'Type Error'

def test_flat_rate_tariff_value_error_fee():
    assert calc_flat_rate_tariff(data, 0.25, -0.1) == 'Value Error'

def test_flat_rate_tariff_value_error_rate():
    assert calc_flat_rate_tariff(data, -0.25, 10) == 'Value Error'
```

Code for test 2 tiered rate tariff:

```
from tiered_tariff.tiered_tariff import calc_tiered_rate_tariff

def test_flat_rate_tariff_typical():
    assert calc_tiered_rate_tariff(300, [ (100, 0.20), (300, 0.30),
(float('inf'), 0.40) ], 10) == {100: 20.0, 300: 60.0, 'total': 90.0}

def test_flat_rate_tariff_type_error_tier():
    assert calc_tiered_rate_tariff(850.67, [ ('100', 0.20), (300,
0.30), (float('inf'), 0.40) ], 10) == 'Type Error'

def test_flat_rate_tariff_type_error_tier_rate():
    assert calc_tiered_rate_tariff(850.67, [ (100, '0.20'), (300,
0.30), (float('inf'), 0.40) ], 10) == 'Type Error'

def test_flat_rate_tariff_type_error_kWh():
    assert calc_tiered_rate_tariff('850.67', [ (0, 0.2), (300, 0.30),
(float('inf'), 0.40) ], 10) == 'Type Error'

def test_flat_rate_tariff_value_error_tier_rate():
    assert calc_tiered_rate_tariff(850.67, [ (100, 0), (300, 0.30),
(float('inf'), 0.40) ], 10) == 'Value Error'

def test_flat_rate_tariff_value_error_tier():
    assert calc_tiered_rate_tariff(850.67, [ (0, 0.2), (300, 0.30),
(float('inf'), 0.40) ], 10) == 'Value Error'

def test_flat_rate_tariff_value_error_kWh():
    assert calc_tiered_rate_tariff(0, [ (100, 0.2), (300, 0.30),
(float('inf'), 0.40) ], 10) == 'Value Error'
```

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
def test_flat_rate_tariff_value_error_not_inf():
    assert calc_tiered_rate_tariff(300, [ (100, 0.20), (300, 0.30),
(500, 0.40) ], 10) == 'Value Error'
```

Code for test 3 time of use tariff:

```
from time_of_use_tariff import calc_time_of_use_tariff
import pandas as pd

data = pd.read_csv('./data/sample_usage_data_month.csv')

def test_time_of_use_tariff_typical():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('19:00:00',
'22:00:00'), ('22:00:00', '7:00:00'), 10) == 202.1175

def test_time_of_use_tariff_type_error_rate():
    assert calc_time_of_use_tariff(data, '0.25', 0.4, 0.15,
('18:00:00', '22:00:00'), ('22:00:00', '7:00:00'), 10) == 'Type Error'

# 18:00 inseat of 18:00:00
def test_time_of_use_tariff_type_error_start_peak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('18:00',
'22:00:00'), ('22:00:00', '7:00:00'), 10) == 'Type Error'

# 22:00 inseat of 22:00:00
def test_time_of_use_tariff_type_error_start_offpeak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('18:00:00',
'22:00:00'), ('22:00', '7:00:00'), 10) == 'Type Error'

# 22:00:sd inseat of 22:00:00
def test_time_of_use_tariff_type_error_end_peak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('18:00:00',
'22:00:sd'), ('22:00:00', '7:00:00'), 10) == 'Type Error'

# 22:00:sd inseat of 22:00:00
def test_time_of_use_tariff_type_error_end_offpeak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('18:00:00',
'22:00:00'), ('22:00:sd', '7:00:00'), 10) == 'Type Error'

def test_time_of_use_tariff_value_error_base_rate():
    assert calc_time_of_use_tariff(data, 0, 0.4, 0.15, ('18:00:00',
'22:00:00'), ('22:00:00', '7:00:00'), 10) == 'Value Error'
```

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
# 24:00:00 is not a valid time
def test_time_of_use_tariff_value_error_start_peak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('24:00:00',
'22:00:00'), ('22:00:00', '7:00:00'), 10) == 'Value Error'

# 24:00:00 is not a valid time
def test_time_of_use_tariff_value_error_start_offpeak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('18:00:00',
'22:00:00'), ('24:00:00', '7:00:00'), 10) == 'Value Error'

# 22:60:00 is not a valid time
def test_time_of_use_tariff_value_error_end_peak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('18:00:00',
'22:60:00'), ('22:00:00', '7:00:00'), 10) == 'Value Error'

# 7:60:00 is not a valid time
def test_time_of_use_tariff_value_error_end_offpeak_time():
    assert calc_time_of_use_tariff(data, 0.25, 0.4, 0.15, ('18:00:00',
'22:00:00'), ('22:00:00', '7:60:00'), 10) == 'Value Error'
```

Code for test 4 suggest tariff:

```
from suggest_tariff import suggested_tariff

import pandas as pd
data = pd.read_csv('./data/sample_usage_data_month.csv')

def test_suggested_tariff_typical_time_of_use():
    assert suggested_tariff(data, 850, 0.3, [ (100, 0.20), (200, 0.30),
(float('inf'), 0.40) ], 0.25, 0.4, 0.15, ('18:00:00', '22:00:00'),
('22:00:00', '7:00:00'), 10) == 'Time Of Use'

def test_suggested_tariff_typical_flat_rate():
    assert suggested_tariff(data, 850, 0.05, [ (100, 0.20), (200,
0.30), (float('inf'), 0.40) ], 0.25, 0.4, 0.15, ('18:00:00',
'22:00:00'), ('22:00:00', '7:00:00'), 10) == 'Flat Rate'

def test_suggested_tariff_typical_tiered_rate():
    assert suggested_tariff(data, 300, 0.2, [ (100, 0.20), (200, 0.30),
(float('inf'), 0.40) ], 0.25, 0.4, 0.15, ('18:00:00', '22:00:00'),
('22:00:00', '7:00:00'), 10) == 'Tiered Rate'

def test_suggested_tariff_value_error_fee():
```

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

```
assert suggested_tariff(data, 850, 0.2, [ (100, 0.20), (200, 0.30),  
(float('inf'), 0.40) ], 0.25, 0.4, 0.15, ('18:00:00', '22:00:00'),  
(('22:00:00', '7:00:00'), -10)) == 'Value Error'  
  
def test_flate_rate_tariff_type_error_fee():  
    assert suggested_tariff(data, 850, 0.2, [ (100, 0.20), (200, 0.30),  
(float('inf'), 0.40) ], 0.25, 0.4, 0.15, ('18:00:00', '22:00:00'),  
(('22:00:00', '7:00:00'), '10')) == 'Type Error'
```

Part 4: Closing Statements

Caleb Griffin (S5393316):

This project has been an invaluable learning experience, both in terms of the specific task at hand and the broader skills of project management and teamwork. Reflecting on the process, there are several key insights that will inform my approach to future group assignments and professional projects. One of the most important lessons I learned was the value of clearly defining roles and responsibilities from the start. Having a strong understanding of who is responsible for each task ensures that everyone is aligned and can work efficiently towards a common goal. I also realized that effective project management was essential in maintaining momentum throughout the project. While we encountered some challenges with time management, a more structured schedule, with set check-ins and intermediate deadlines, would have helped us stay even more on track. A standout positive aspect of this project was the collaboration among our team. Despite occasional challenges, we kept the lines of communication open, which allowed us to support one another and keep the project moving forward. In summary, this project reinforced the importance of careful planning, clear communication, and adaptability within a team setting. While there were opportunities to improve our time management and increase our collaboration, I am happy with the final result and the way we worked together. The lessons I've learned about effective teamwork and balancing individual and group responsibilities will be invaluable for future projects. I now feel more confident in my ability to manage tasks within a team and contribute effectively to collective efforts.

Koby Hanham (S5393401):

When working on this assessment task, the clear aspect that stood out the most to complete this task successfully was the need for teamwork between all members. From the start, splitting the roles fairly and evenly helped the task to move from the initial blank state as a clear idea was made on exactly what each teammate had to complete. Keeping open and often communication between the members helped in ensuring the task was moving forward and progress continued to be made. Being able to ask questions and opinions from the other team members helped to create a better product due to them having a fresh mind with differing opinions when reviewing each other's work. In terms of my individual experience with this assessment, when completing my sections of the work I was able to focus on sections of the assessment where I felt more strongly about my ability to complete the work well. This also allowed my team members to focus on parts they are confident in, resulting in each of us combining to create a strong product. Also, for this task, I needed to learn how to use Wireframe to complete the High-Level Architectural Diagram. This experience allowed me to expand my knowledge by creating a more informative project report and can assist me in future projects through being able to create a mockup design for UI. For future possible situations where work with a team is needed, I am confident I will be able to work collaboratively with them to create the strongest possible product.

Caleb Griffin (S5393316), Koby Hanham (S5393401), Alex Henry (S5398949)

Alex Henry (S5398949):

From the start of this assignment we had divided the work to as evenly as possible along with being able to focus on what we were each best at, for me that was primarily categorising the tasks into smaller parts and then explaining the different parts of the project in a way that would be understandable to the hypothetical stakeholders. Although we probably could've been better organised in terms of working alongside the assignment together more, we were able to maintain communication when we needed help or had constructive criticism about the work each other did and to keep each other accountable about doing our parts of the assignment when we would get busy/distracted. So I am happy with the work put in by my other team members, especially Caleb as he took over what I'd say is the most important part of the assignment and did it punctually and very well. I however definitely was a bit late doing my parts of the assignment due to poor time management and related distractions, so I thank my teammates for giving me the time I needed to get to working on this assignment. This experience gave me more experience being able to partition parts of a project and explaining my ideas/plans to others which will be very useful in future for group assignments or other projects and teamwork activities, along with further highlighting my need to work on my time management/planning so as to not impact future teammates in other projects I am a part of later on.