MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EE374
FUNDAMENTALS OF POWER SYSTEMS AND ELECTRICAL EQUIPMENT

Term Project Report

Fatih Kaan Öz, 2375434

Group Number: 7

Content :

1. Introduction

This report explains how the Power UI Application was designed and built. The main purpose of this application is to help users choose the right power cable by entering some basic information such as load specification, environment details. Based on the input, the program calculates things like line losses, voltage regulation and economical aspects of the selected cable.

The report covers design philosophy, project folder structure, GUI design, database design, core backend functionalities, and user experience. To make the application easier to maintain and improve in the future, the code was divided into many folders. Each folder has a specific purpose, such as handling the user interface, doing calculations, or managing the database. This structure helps keep the code clean and organized. It also makes it easier to fix bugs or add new features later. Even though the application is not very large, following this approach makes it more scalable and professional. All of these steps are explained in this report.

While sections 2, 3, 4, 5, and 6 focus on the design and internal structure of the application, section 7 presents the application from the user's point of view.

2. Design Philosophy

I believe that following a consistent mental model during system design leads to clearer and more justifiable decisions. Even though the Power UI application does not have complex requirements, the considerations outlined below have been followed:

1) Take it seriously. (As Prof. Göl mentioned when the project was announced.)
2) Believe that designing with layered and scalable functional units helps simplify both development and maintenance.
3) The fewer a functional unit knows, the better and less error prone.
4) Follow established web development patterns, even though this is not a web application.

A modified version of the Model-View-Controller (MVC) pattern was used in architecting the Power UI Application. The Controller logic primarily resides in the Business folder, while the View component is implemented in the GUI folder. However, the Model aspect of the pattern is somewhat distributed across the entire backend logic. As a result, the "contract" between the GUI and backend -typically represented by the Model in a classic MVC architecture- is not strictly defined in this implementation.

3. Folder Structure

To display all the relevant folders (directories) in the project, the following command was used, and its output is shown below:

```
tree -d -L 10 -I '__pycache__' --prune
```

```
├── app
│   ├── backend
│   │   ├── business
│   │   │   └── data
│   │   └── database
│   └── gui
│       ├── components
│       │   ├── inputs
│       │   ├── result
│       │   ├── steps
│       │   ├── tabs
│       │   └── toaster
│       ├── constants
│       ├── pages
│       │   └── tabs
│       └── style
│           ├── fonts
│           │   ├── exo
│           │   ├── quicksand
│           │   │   └── static
│           │   └── tektur
│           │       └── static
│           └── toaster
└── checkpoints

25 directories.
```

app:	All project related code resides here.
backend:	Business and Database logic resides here.
gui:	Stylesheet and components reside here.
checkpoints:	Some screenshots are put here from time to time.

Also the gui folder has lots of subfolders for components, fonts, stylesheets.

4. GUI Design

Dozens of Python files reside in the app/gui/components folder. All the classes within these files inherit from the QWidget class provided by PySide, effectively serving as wrappers around QWidget. Most of these classes also define custom signal fields. These signals are connected to specific functions within the same wrapper classes. This design enables reactive behavior. When a signal is emitted, the connected function is triggered, allowing the widget's state to be updated dynamically. In Power UI Application context, when user input changes these functions are triggered.

Styling is primarily handled through external CSS files. However, when dynamic styling is required at runtime, inline style sheets which are implemented as plain strings are used.

Organizing separate CSS files for each component has significantly improved maintainability and simplified bug fixing. This approach also ensures a clear separation between GUI logic and styling, leading to cleaner and more modular code. To import CSS files, a helper function named load_stylesheet is used. This function is imported and utilized by almost all GUI components.

In this way, the GUI development closely mimics modern web development practices, particularly those used in well-known and successful JavaScript frameworks such as React and NextJS.
Exo, Quicksand, and Tektur font files were downloaded from Google Fonts and are embedded into the executable at runtime. The build specification file located at the root of the project explicitly includes these fonts in the final executable. All three fonts are licensed under the SIL Open Font License, which permits free use for non-commercial purposes.

5. Database

At this scale, storing all cable details in memory is entirely feasible. However, design philosophy (2) was a key consideration from the very beginning. Therefore, instead of hardcoding the data, all cable details are extracted from the Excel file and stored in a SQLite database to ensure better scalability and maintainability. This embedded database likely introduces a slight overhead on the order of milliseconds, primarily due to the SQLite engine, which itself is embedded as a binary component.

5.1 Fields

Except for the voltage column in the Excel file, all other columns in the database match the columns in the Excel file. The voltage column in the Excel file is divided into two separate fields in the database -namely, line-to-neutral and line-to-line voltages- to simplify querying the database. Also, NULL is also allowed in the database to eliminate some cables for calculation phase.

5.2 Queries

For connecting to the database, **connect_database** function in app/backend/database/db.py is used. It creates a singleton to connect to the database. All queries sit in the app/backend/database/queries.py file. While some of them are more generic and construct queries using function parameters, others are more specific to particular use cases. These functions are agnostic to user input. The business layer separates the user input received from the GUI from the underlying database logic. It acts as a barrier between the GUI and the database, ensuring modularity and a clean separation. Both the function names and their parameters are highly self-explanatory therefore, the internal details of the functions will not be explained here.

6. Backend & Functions at Business Logic

The core calculation logic resides here. There are 14 functions in app/backend/business/calculate.py file.

Find_Proper_Cables: This function is part of the **smart listing algorithm**. They are called from components at the GUI folder. Based on user input, it calculates the required voltage and current levels while taking into account temperature and trench reduction factors. Next, with the help of the select_from_db function, the database query is executed. The if-else block within this function operates similarly to a decision tree, guiding the selection logic based on user input.
Note: Voltage levels are quantized using a helper function located in the queries.py file to ensure compatibility with predefined database entries.

Calculate_Active_Losses: This function utilizes the capacitance field of the selected cable. If the capacitance value is NULL, the **Short Line Model** is used to calculate the corresponding losses. If the capacitance value is present (not NULL), the **Medium Line Model** is applied instead. So, model selection is based on selected cable.

Calculate_Reactive_Losses: This function always uses the **Short Line Model** for calculations.

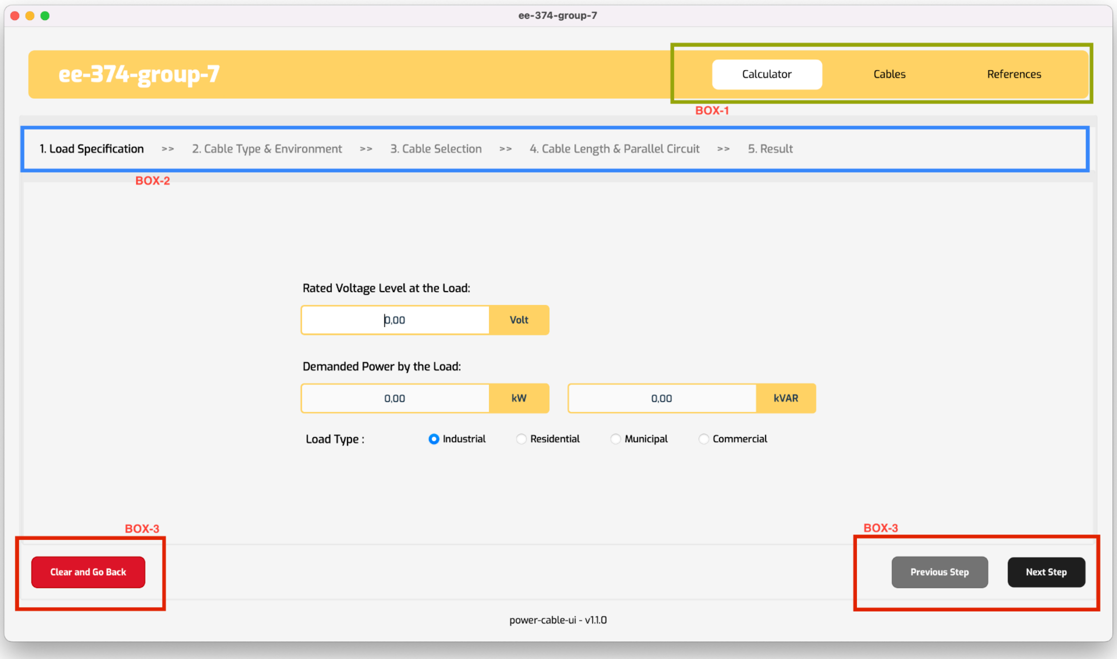
Calculate_Voltage_Regulation: In this function, after performing necessary unit conversions, the total impedance and apparent power are calculated. Using these values, the voltage drop along the line is determined. Finally, the basic voltage regulation formula is applied to compute the regulation value. The **Short Line Model** is used for voltage regulation calculation.

Note: The economic aspect section is calculated using either the Short Line Model or the Medium Line Model to improve the accuracy of the calculation.

All the functions explained above are invoked by components located in the GUI folder. Therefore, the overall calculation flow can be summarized as follows:

Input changes trigger the emission of signals,
Emitted signals lead to the invocation of corresponding business logic functions,
Return values from these functions update the state of the relevant GUI components.

7. User Experience



Rated Voltage Level at the Load: 0.00 Volt

Demanded Power by the Load: 0.00 kW 0.00 kVAR

Load Type : [✓ Industrial] Residential Municipal Commercial

Figure-7.1

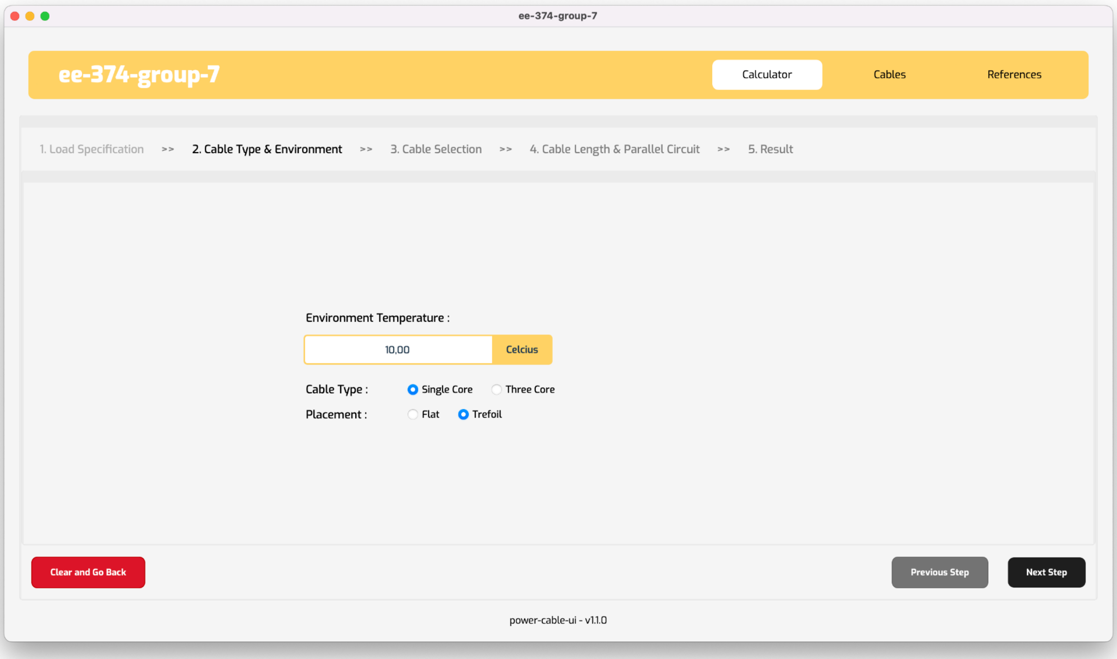
The box labeled Box-2 illustrates the different **pages** of the application. Navigating between these pages using the buttons in this box does not change the state of the pages. For example, if the user is on Step 4 and wants to view all the cables in the database, they can switch to the Cables page and then return to the Calculator page without losing any previously entered input.

The box labeled Box-2 illustrates the main workflow from the user's perspective. Step by step, the user is required to provide various inputs, such as load specifications, environmental conditions, and cable length details. When the user passes from the second step to the third, the smart listing algorithm is triggered, which filters and lists the appropriate cables based on the provided inputs. **Steps** in GUI design correspond to this box.

The box labeled Box-3 illustrates the buttons that allow the user to navigate between steps or clear all inputs. If the input provided at a given step is invalid or incomplete, a toaster message is displayed when the user attempts to proceed. These messages are implemented without using any additional libraries; instead, a separate layout is dynamically stacked and animated to display the messages, providing a responsive user experience.

All numeric input fields are implemented using the **QSpinBox** class. Each field has defined lower and upper bounds to prevent users from entering invalid values. For decimal separation, a **comma** (,) should be used instead of a period.

Load specification inputs such as rated voltage, active/reactive power, and load type are collected in the first step.


ee-374-group-7
ee-374-group-7
Calculator
Cables
References
1. Load Specification >> 2. Cable Type & Environment >> 3. Cable Selection >> 4. Cable Length & Parallel Circuit >> 5. Result
Clear and Go Back
Previous Step
Next Step
power-cable-ui - v1.1.0

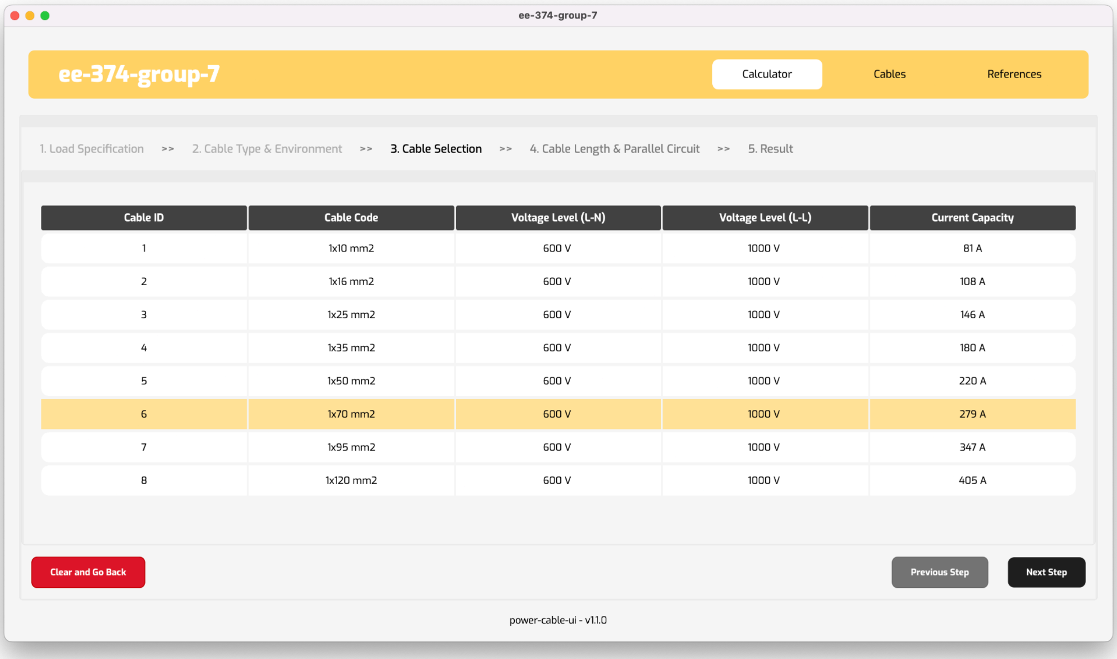
Environment Temperature : 10,00 Celcius

Cable Type : [✓ Single Core] Three Core

Placement : Flat [✓ Trefoil]

Figure-7.2

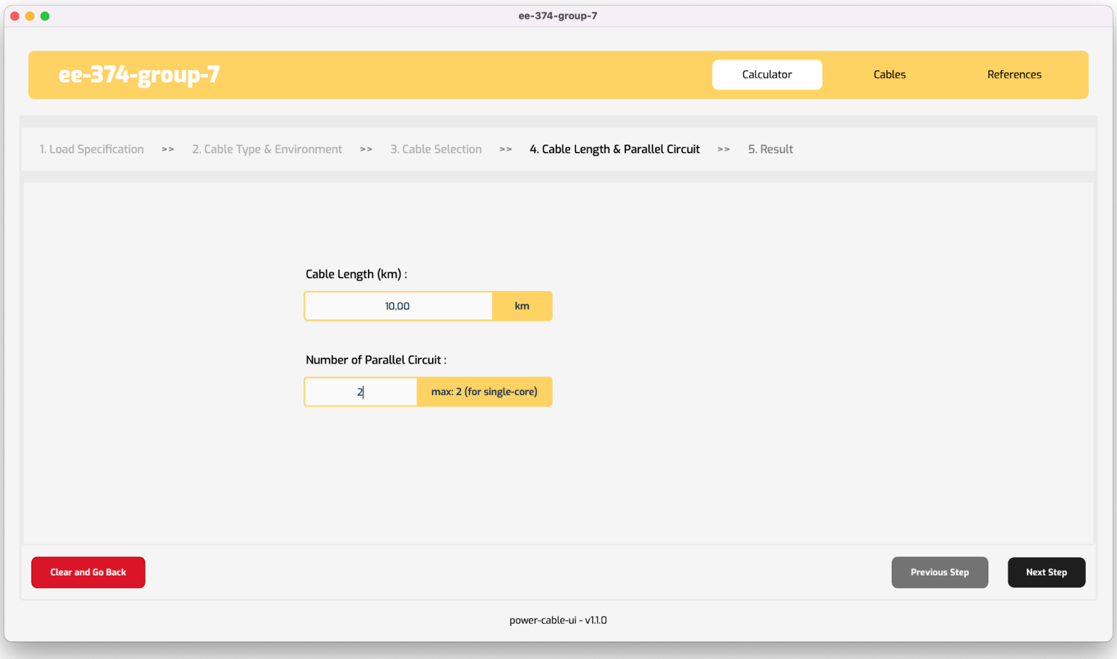
In the second step, the user is required to provide the cable type, ambient temperature, and cable placement. If the selected cable type is **Three Core**, the placement selection is hidden, as it becomes irrelevant in that context.


ee-374-group-7
ee-374-group-7
Calculator
Cables
References
1. Load Specification >> 2. Cable Type & Environment >> 3. Cable Selection >> 4. Cable Length & Parallel Circuit >> 5. Result
Clear and Go Back
Previous Step
Next Step
power-cable-ui - v1.1.0

Cable ID	Cable Code	Voltage Level (L-N)	Voltage Level (L-L)	Current Capacity
1	1x10 mm2	600 V	1000 V	81 A
2	1x16 mm2	600 V	1000 V	108 A
3	1x25 mm2	600 V	1000 V	146 A
4	1x35 mm2	600 V	1000 V	180 A
5	1x50 mm2	600 V	1000 V	220 A
6	1x70 mm2	600 V	1000 V	279 A
7	1x95 mm2	600 V	1000 V	347 A
8	1x120 mm2	600 V	1000 V	405 A

Figure-7.3

At the third step, the user must select one of the cables from the filtered list to proceed. When the user hovers over a cable option, the background color of the corresponding item changes to indicate interactivity. Clicking on a cable selects it, allowing the user to move on to the next step. So, at this step, the user–application interaction becomes more intuitive rather than relying on strict form-based input.

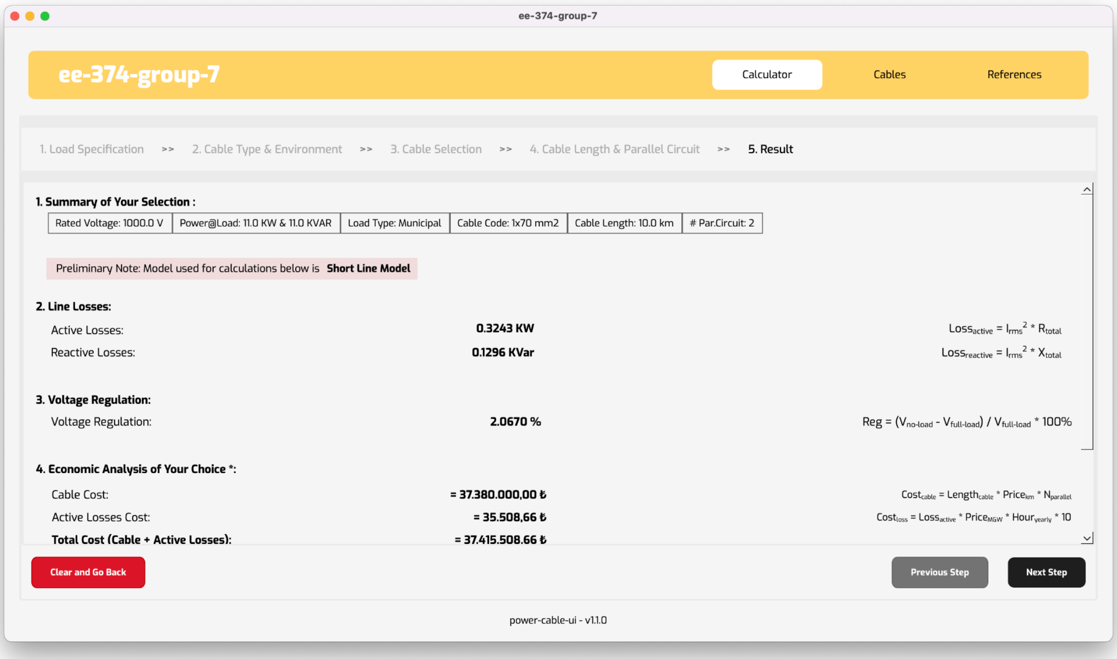

ee-374-group-7
ee-374-group-7
Calculator
Cables
References
1. Load Specification >> 2. Cable Type & Environment >> 3. Cable Selection >> 4. Cable Length & Parallel Circuit >> 5. Result
Clear and Go Back
Previous Step
Next Step
power-cable-ui - v1.1.0

Cable Length (km) : 10,00 km

Number of Parallel Circuit : 2 max: 2 (for single-core)

Figure-7.4

In the fourth step, the user provides inputs for cable length and the number of parallel circuits. The number of parallel circuits has an upper bound that is dynamically updated based on the cable type selected in the second step. If the user selects Single Core in the second step, the maximum allowed number of parallel circuits is limited to 2.
A small informational text appears next to the input field to inform the user of this constraint.


ee-374-group-7
ee-374-group-7
Calculator
Cables
References
1. Load Specification >> 2. Cable Type & Environment >> 3. Cable Selection >> 4. Cable Length & Parallel Circuit >> 5. Result
Clear and Go Back
Previous Step
Next Step
power-cable-ui - v1.1.0

1. Summary of Your Selection :

Rated Voltage: 1000.0 V	Power@Load: 11.0 KW & 11.0 KVAR	Load Type: Municipal	Cable Code: 1x70 mm2	Cable Length: 10.0 km	# Par.Circuit: 2

Preliminary Note: Model used for calculations below is **Short Line Model**

2. Line Losses:

Active Losses:	**0.3243 KW**	$Loss_{active} = I_{rms}^2 * R_{total}$
Reactive Losses:	**0.1296 KVar**	$Loss_{reactive} = I_{rms}^2 * X_{total}$

3. Voltage Regulation:

Voltage Regulation:	**2.0670 %**	$Reg = (V_{no\text{-}load} - V_{full\text{-}load}) / V_{full\text{-}load} * 100\%$

4. Economic Analysis of Your Choice *:

Cable Cost:	**= 37.380.000,00 ₺**	$Cost_{cable} = Length_{cable} * Price_{km} * N_{parallel}$
Active Losses Cost:	**= 35.508,66 ₺**	$Cost_{loss} = Loss_{active} * Price_{MGW} * Hour_{yearly} * 10$
Total Cost (Cable + Active Losses):	**= 37.415.508.66 ₺**	

Figure-7.5

The results of the Power UI Application are displayed in the fifth step. A brief summary of the user's inputs is shown at the top of this step, along with the selected line model either the Short Line or Medium Line model.

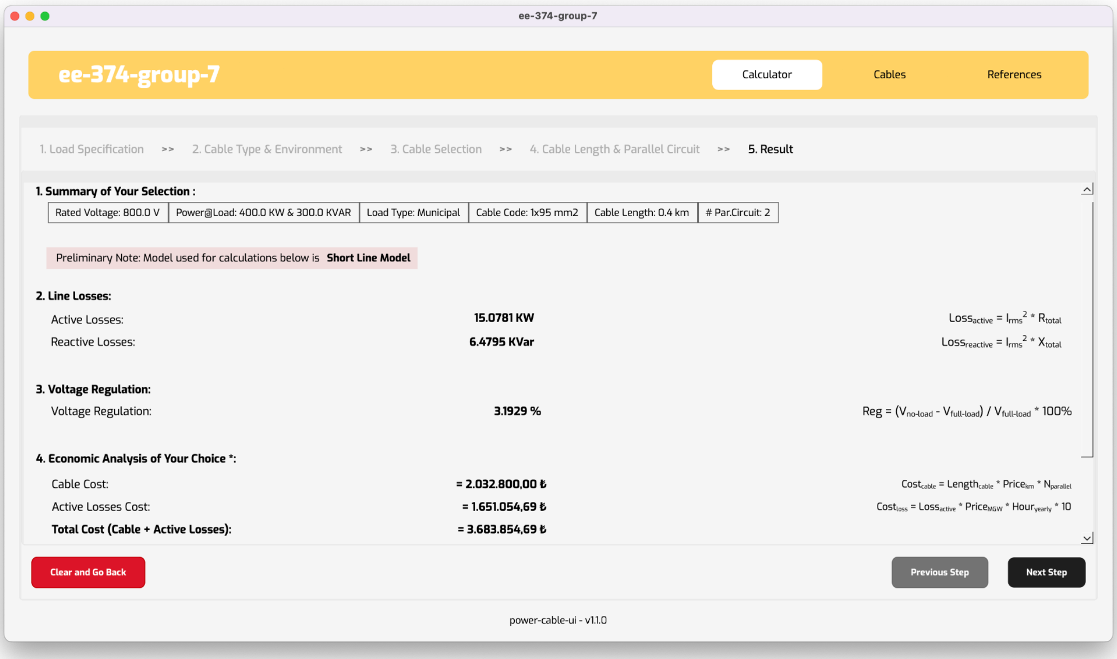
Below the summary, three main result sections are presented: Line Losses, Voltage Regulation, and Economic Analysis. Instead of using traditional tables with borders, the results and formulas are aligned using the alignment properties of QLabel, providing a cleaner and more visually appealing layout.

8. Test Results

In this section, the example inputs provided on ODTUClass are used to test the Power UI Application. For each example, a screenshot is taken. The summary shown at the result step reflects the corresponding user inputs.

Example 1:

Rated Voltage : 800 Volt;	Load Type : Municipal
Active Power : 400 KW;	Reactive Power : 300 KVar
Cable Length : 0.4 km;	Number of Parallel Circuit : 2
Cable ID : 7 (1 x 95mm);	Single Core, Flat


ee-374-group-7
ee-374-group-7
Calculator
Cables
References
1. Load Specification >> 2. Cable Type & Environment >> 3. Cable Selection >> 4. Cable Length & Parallel Circuit >> 5. Result
Clear and Go Back
Previous Step
Next Step
power-cable-ui - v1.1.0

1. Summary of Your Selection :

Rated Voltage: 800.0 V	Power@Load: 400.0 KW & 300.0 KVAR	Load Type: Municipal	Cable Code: 1x95 mm2	Cable Length: 0.4 km	# Par.Circuit: 2

Preliminary Note: Model used for calculations below is **Short Line Model**

2. Line Losses:

Active Losses:	**15.0781 KW**	$Loss_{active} = I_{rms}^2 * R_{total}$
Reactive Losses:	**6.4795 KVar**	$Loss_{reactive} = I_{rms}^2 * X_{total}$

3. Voltage Regulation:

Voltage Regulation:	**3.1929 %**	$Reg = (V_{no\text{-}load} - V_{full\text{-}load}) / V_{full\text{-}load} * 100\%$

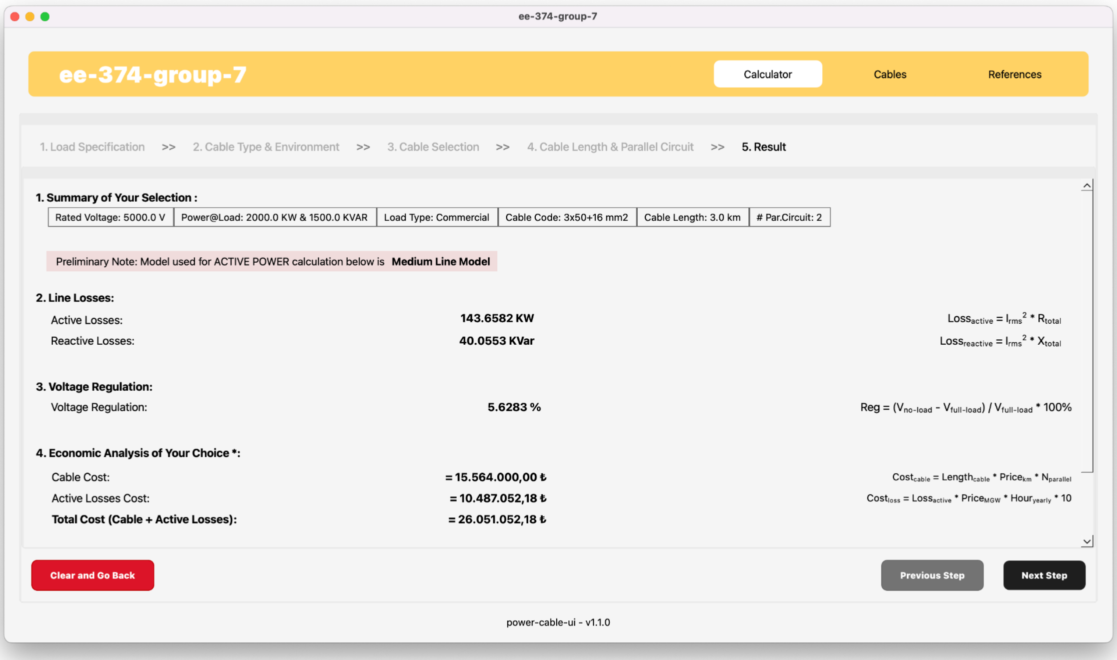
4. Economic Analysis of Your Choice *:

Cable Cost:	**= 2.032.800,00 ₺**	$Cost_{cable} = Length_{cable} * Price_{km} * N_{parallel}$
Active Losses Cost:	**= 1.651.054,69 ₺**	$Cost_{loss} = Loss_{active} * Price_{MGW} * Hour_{yearly} * 10$
Total Cost (Cable + Active Losses):	**= 3.683.854,69 ₺**	

Figure 8-1

Example 2:

Rated Voltage : 5000 Volt;	Load Type : Commercial
Active Power : 2000 KW;	Reactive Power : 1500 KVar
Cable Length : 3 km;	Number of Parallel Circuit : 2
Cable ID : 27 (3x50 + 16mm);	Three Core


ee-374-group-7
ee-374-group-7
Calculator
Cables
References
1. Load Specification >> 2. Cable Type & Environment >> 3. Cable Selection >> 4. Cable Length & Parallel Circuit >> 5. Result
Clear and Go Back
Previous Step
Next Step
power-cable-ui - v1.1.0

1. Summary of Your Selection :

Rated Voltage: 5000.0 V	Power@Load: 2000.0 KW & 1500.0 KVAR	Load Type: Commercial	Cable Code: 3x50+16 mm2	Cable Length: 3.0 km	# Par.Circuit: 2

Preliminary Note: Model used for ACTIVE POWER calculation below is **Medium Line Model**

2. Line Losses:

Active Losses:	**143.6582 KW**	$Loss_{active} = I_{rms}^2 * R_{total}$
Reactive Losses:	**40.0553 KVar**	$Loss_{reactive} = I_{rms}^2 * X_{total}$

3. Voltage Regulation:

Voltage Regulation:	**5.6283 %**	$Reg = (V_{no\text{-}load} - V_{full\text{-}load}) / V_{full\text{-}load} * 100\%$

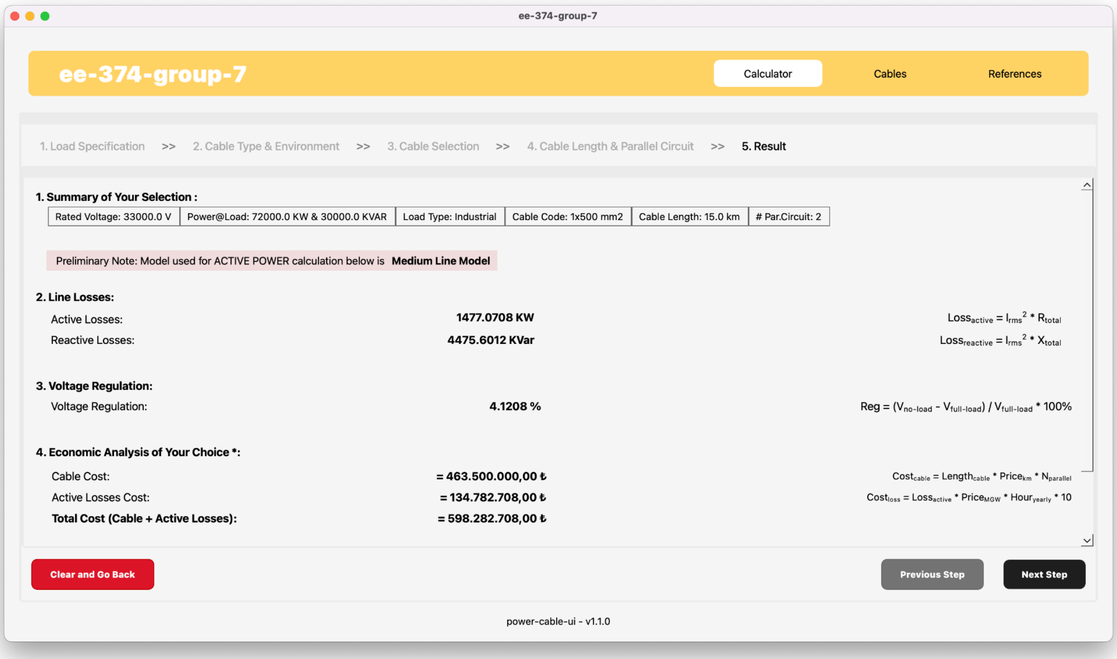
4. Economic Analysis of Your Choice *:

Cable Cost:	**= 15.564.000,00 ₺**	$Cost_{cable} = Length_{cable} * Price_{km} * N_{parallel}$
Active Losses Cost:	**= 10.487.052,18 ₺**	$Cost_{loss} = Loss_{active} * Price_{MGW} * Hour_{yearly} * 10$
Total Cost (Cable + Active Losses):	**= 26.051.052,18 ₺**	

Figure 8-2

Example 3:

Rated Voltage : 33000 Volt;	Load Type : Industrial
Active Power : 72000 KW;	Reactive Power : 30000 KVar
Cable Length : 15 km;	Number of Parallel Circuit : 2
Cable ID : 59 (1x500 mm);	Single Core; Trefoil


ee-374-group-7
ee-374-group-7
Calculator
Cables
References
1. Load Specification >> 2. Cable Type & Environment >> 3. Cable Selection >> 4. Cable Length & Parallel Circuit >> 5. Result
Clear and Go Back
Previous Step
Next Step
power-cable-ui - v1.1.0

1. Summary of Your Selection :

Rated Voltage: 33000.0 V	Power@Load: 72000.0 KW & 30000.0 KVAR	Load Type: Industrial	Cable Code: 1x500 mm2	Cable Length: 15.0 km	# Par.Circuit: 2

Preliminary Note: Model used for ACTIVE POWER calculation below is **Medium Line Model**

2. Line Losses:

Active Losses:	**1477.0708 KW**	$Loss_{active} = I_{rms}^2 * R_{total}$
Reactive Losses:	**4475.6012 KVar**	$Loss_{reactive} = I_{rms}^2 * X_{total}$

3. Voltage Regulation:

Voltage Regulation:	**4.1208 %**	$Reg = (V_{no\text{-}load} - V_{full\text{-}load}) / V_{full\text{-}load} * 100\%$

4. Economic Analysis of Your Choice *:

Cable Cost:	**= 463.500.000,00 ₺**	$Cost_{cable} = Length_{cable} * Price_{km} * N_{parallel}$
Active Losses Cost:	**= 134.782.708,00 ₺**	$Cost_{loss} = Loss_{active} * Price_{MGW} * Hour_{yearly} * 10$
Total Cost (Cable + Active Losses):	**= 598.282.708,00 ₺**	

Figure 8-3

9. Discussion

There is no doubt that electrical energy is one of the most vital parts of modern life. Millions of lives directly depend on its reliable transmission, while billions of dollars and tremendous effort are spent on transmitting it. A significant portion of these efforts focuses on making electricity generation and especially transmission as cost-efficient as possible.

The infrastructure of both transmission and distribution lines typically lasts for decades. This makes long term predictions for energy demand and transmission losses are more and more important. Once these lines are installed, making changes or upgrades can be costly, challenging, and power outage prone. These are not acceptable economically and in terms of public safety.

At first, cable installation costs might appear to be the most important factor in planning transmission infrastructure. However, just focusing on these initial expenses might be misleading. Over the long term, transmission line losses can have a much greater impact on the total cost. Therefore, while minimizing installation costs is important, it must be balanced with long term efficiency.

The Power UI application is designed to provide users with some insights into the costs associated with power transmission. It allows users to compare different cable types under various load conditions. However, its results are highly mechanical, meaning they do not account for the dynamic energy demand at the load side. This limitation can be addressed by integrating a deep learning model, which can be trained using data from SCADA systems. (Actually, I prefer this kind of comprehensive project work, as it offers deeper learning opportunities and practical insights beyond theoretical knowledge.)

In summary, realistic predictions of energy demand, accurate modeling of losses, and realistic estimation of the operational lifetime of infrastructure should all be considered at the first place. These factors are essential to ensure efficient, cost-effective, and reliable power systems.

10. Conclusion

This report elaborated on the design process of the Power UI Application, covering both the design philosophy and implementation details, as well as the usage of the application. Additionally, it explained the theoretical aspects of the development process and the transmission line models used.

Sections 3 through 6 provided insights into the application's development and detailed the folder structure. Section 7 focused on the user-facing side of the application, describing the user interface and its pages, accompanied by relevant screenshots.

Furthermore, the report discussed the importance of electrical energy transmission and highlighted critical considerations in designing transmission line infrastructure.

A1. References

Fitzpatrick, Martin. Create GUI Applications with Python & Qt6 (PySide6 Edition): The Hands-on Guide to Making Apps with Python. Independently published, 2021.

Qt for Python Documentation. The Qt Company Ltd., 2025, Accessed 9 June 2025 https://doc.qt.io/qtforpython-6/

Fitzpatrick, Martin. PySide6 Tutorial 2025: Create Python GUIs with Qt. Python GUIs, Accessed 9 June 2025, https://www.pythonguis.com/pyside6-tutorial/