

EksPy: a new Python framework for developing graphical user interface based PyQt5

Aidil Saputra Kirsan¹, Kosuke Takano², Sallie Trixie Zebada Mansurina¹

¹Department of Information Systems, Faculty of Mathematics and Information Technology, Institut Teknologi Kalimantan, Balikpapan, Indonesia

²Department of Information and Computer Sciences, Faculty of Information Technology, Kanagawa Institute of Technology, Atsugi, Japan

Article Info

Article history:

Received Mar 29, 2023

Revised Jul 16, 2023

Accepted Jul 17, 2023

Keywords:

EksPy framework
Framework development
Model-view-controller architecture
PyQt5
Python graphical user interface

ABSTRACT

This study introduces EksPy Python framework, a novel framework designed for developing graphical user interface (GUI) applications in Python. EksPy framework is built on PyQt5, which is a collection of Python bindings for the Qt libraries, and it provides a user-friendly and intuitive interface. The comparative analysis of EksPy framework with existing frameworks such as Tkinter and PyQt highlights its notable features, including ease of use, rapid development, enhanced performance, effective database management, and the model-view-controller (MVC) concept. The experimental results illustrate that EksPy framework requires less code and enhances code readability, thereby facilitating better understanding and efficient development. Additionally, EksPy framework offers a modern and customizable appearance, surpassing Tkinter's capabilities. Furthermore, it incorporates a built-in object-relational mapping (ORM) feature to simplify database interactions and adheres to the MVC architectural pattern. In conclusion, EksPy Python framework emerges as a powerful, user-friendly, and efficient framework for GUI application development in Python.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Aidil Saputra Kirsan

Department of Information Systems, Faculty of Mathematics and Information Technology, Institut Teknologi Kalimantan

Soekarno Hatta No. KM 15, Karang Joang, Balikpapan Utara, Balikpapan, East Kalimantan, 76127, Indonesia

Email: aidil@lecturer.itk.ac.id

1. INTRODUCTION

Graphical user interface (GUI) is a crucial component in software development. However, developing GUI can be a complex and time-consuming process, especially compared to developing the logic of the application itself. Therefore, GUI frameworks are useful tools to speed up the development process and simplify the creation of commonly used components. In this study, we propose EksPy, which is a new Python framework developed for creating GUI based on PyQt5. This framework is designed to simplify the GUI development process by providing ready-to-use components that can be easily customized to fit the needs. Additionally, EksPy also provides features that can accelerate the development process of GUI applications, allowing developers to focus on the logic of the application and not waste time creating basic components. EksPy also has comprehensive documentation and active community support, making it suitable for developing complex and advanced GUI applications created using other GUI frameworks [1]–[10].

In this research, we will further explain about EksPy, starting from the framework's architecture to the features offered. We will also show an experiment of applying EksPy in developing a simple GUI application and evaluate the performance of this framework compared to other GUI frameworks. The

purpose of this paper is to show the advantages of EksPy for efficient GUI development and to provide guidance for developers who want to use EksPy in their projects. Furthermore, this paper will also discuss the additional features that EksPy offers such as database management, and the implementation of the model-view-controller (MVC) concept. Overall, EksPy is a comprehensive framework that can help Python developers create responsive, user-friendly, and visually appealing GUI applications, as well as ease the management of data and the implementation of the MVC concept that helps developers manage the application well [10]–[19].

In addition to the previously mentioned features, EksPy also provides a database management feature that allows developers to easily access and manage data. This feature uses the object-relational mapping (ORM) concept, which allows developers to work with Python objects as representations of data in the database. This makes the process of retrieving and inserting data more efficient and easy to understand compared to writing structured query language (SQL) commands manually [20]–[25].

EksPy also implements the MVC concept, which helps developers to separate the logic of the application from the view and control. This allows developers to manage and change the appearance of the application without having to make changes to the logic of the application more easily. This concept also makes the application more scalable and easier to develop by larger teams.

Overall, EksPy is a comprehensive framework that can assist Python developers in creating responsive, user-friendly, and visually appealing GUI applications, as well as facilitate data management and the implementation of the MVC concept for effective application management. The article will also elaborate more on how to use EksPy to manage databases and how the MVC concept is implemented in EksPy. With EksPy, developers can save time and effort in creating GUI applications and can focus on the design and implementation of the logic and functionality of their application.

2. PROPOSED FRAMEWORK

The proposed framework, EksPy, is designed to simplify the GUI development process for Python developers by providing ready-to-use components that can be easily customized to fit their needs. The framework is built on top of PyQt5, which is a powerful and flexible GUI framework. The EksPy framework is integrated with PyQt5 as its underlying technology. The Core Library, GUI elements, and layout management components of EksPy are leveraging the capabilities of PyQt5 to provide a more robust and efficient framework. EksPy provides features that can accelerate the development process of GUI applications, allowing developers to focus on the logic of the application and not waste time creating basic components. Figure 1 depicts the proposed framework comprising several key components.

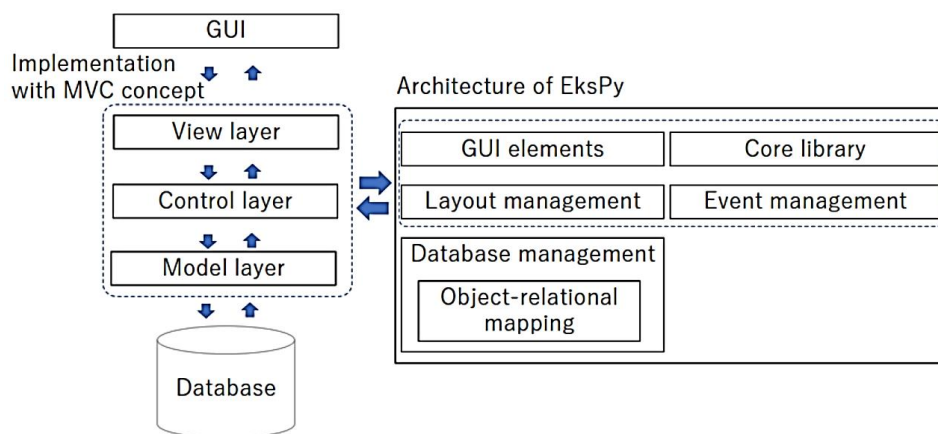


Figure 1. Architecture of EksPy framework

Figure 1 presents the architecture of the proposed EksPy framework, which aims to simplify the GUI development process for Python programmers. The framework is built on PyQt5 and comprises essential components such as the Core Library, GUI elements, layout management, event management, and database management, and follows the MVC concept. The Core Library provides foundational functionalities, while GUI Elements offer pre-built user interface elements for easy integration. Layout management assists in organizing GUI elements, and event management handles user interactions effectively. Database management ensures seamless integration with databases. By following the MVC concept, EksPy

EksPy: a new Python framework for developing graphical user interface based PyQt5 (Aidil Saputra Kirsan)

promotes code modularity and enhances the maintainability and extensibility of GUI applications. This architecture empowers developers by providing a robust and efficient framework for GUI application development.

2.1. Core library

This component provides the base classes and functions that are used to create the GUI elements, such as buttons, labels, and input fields. The Core library is the core component of the EksPy framework that contains all the basic features and core functions in application development. It is responsible for providing access to all the resources and features required by other components such as GUI elements, layout management, event management, and database management.

The Core Library also ensures the integrity and consistency of data throughout the application. Unique and custom features of applications built using EksPy can also be developed and added through the Core Library. This component is the most important part of the framework that ensures the performance and efficiency of the built application.

2.2. GUI elements

This component provides ready-to-use GUI elements that can be easily customized to fit the needs. GUI elements are components in the EksPy framework responsible for providing an interface between the application and the user. They include all display elements such as buttons, forms, and labels that allow the user to interact with the application.

The GUI elements are designed to provide an intuitive and user-friendly display, making the user experience of running the application easier. This component works in conjunction with the Core Library and layout management to ensure a consistent and efficient display of the application. The GUI Elements also allow developers to create custom displays and additional features as needed by the application. In EksPy, GUI elements can be easily implemented and developed, making it easier to develop applications with an attractive and user-friendly display.

2.3. Layout management

This component provides tools for managing the layout of the GUI elements. Layout management is a component within the EksPy framework responsible for organizing and arranging the visual elements within an application. It determines the placement, size, and spacing of GUI Elements such as buttons, forms, labels, and others.

The layout management component works in conjunction with the Core Library and GUI elements to ensure a consistent and efficient appearance of the application. This component provides the ability to easily adjust and modify the layout of the application, even as its content changes. This can include dynamic resizing, support for different screen sizes, and the ability to display different layouts for different devices. Layout management in EksPy also enables developers to create custom layouts and add additional features to meet the specific needs of the application.

2.4. Event management

This component provides tools for managing the events that occur in the application. Examples of event management tools include signal, slot, and event filters. Event management is a component of the EksPy framework that ensures the application can react to events such as mouse clicks, key presses, and others. It ensures that the application can respond accurately and efficiently when the user interacts with the application interface through GUI elements. Event management makes sure that every event is recognized and processed correctly so that the application can provide an appropriate and satisfactory response to the user.

This component also ensures that the events can be passed to other components such as the Core Library or database management for further processing such as updating the database or processing other data. With event management, developers can create more interactive applications with features that can be customized to meet the needs of the application. In EksPy, event management is designed to facilitate the development of applications with efficient and accurate user interaction.

2.5. Database management

This component provides tools for managing data. Using the ORM concept, it allows developers to work with Python objects as representations of data in the database. The ORM concepts are illustrated in Figure 2. Figure 2 depicts the relationship between the database and the ORM, as well as the relationship between the ORM and the Python code. ORM is a component that enables accessing and manipulating data in the database through objects in a programming language like Python. On the right side of the

figure, tables in the database are represented by objects in the ORM, making it easier for developers to access and manipulate data in the database. Then, on the left side of the figure, there is Python code that uses objects from the ORM to access and manipulate data in the database, without having to write complex SQL code. By using ORM, developers can easily access and manipulate data in the database because ORM facilitates communication between the programming language and the database. ORM also helps reduce the complexity of writing SQL code, making it easier to develop applications that interact with the database.



Figure 2. Object-relational mapping (ORM) concept

2.6. MVC concept

This component implements the MVC concept, which helps developers to separate the logic of the application from the view and control. This allows developers to manage and change the appearance of the application without having to make changes to the logic of the application more easily. The MVC concept is illustrated in Figure 3.

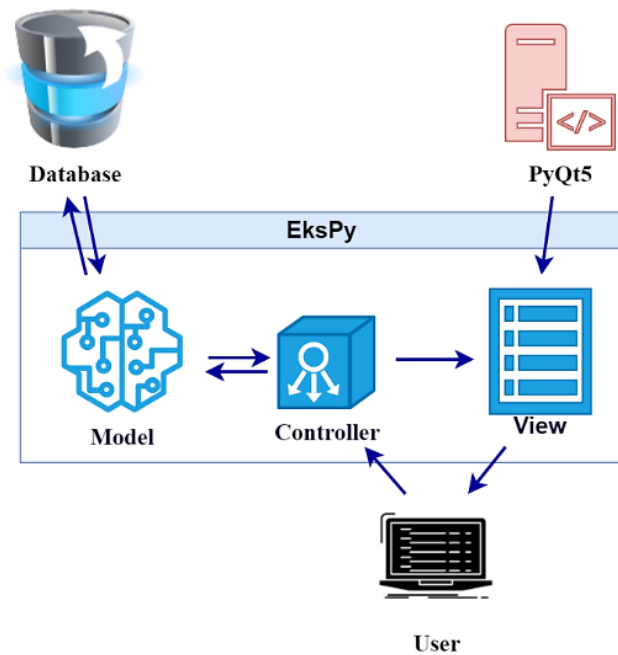


Figure 3. Implementation with MVC architecture

Figure 3 illustrates the concept of the MVC architecture, with an added representation of the relationships between view and user, user, and controller, and PyQt5 and view. The figure presents a representation of the model, containing the data and business logic of the application. Furthermore, it depicts a representation of the view, responsible for displaying the data from the model to the user. Additionally, it shows a representation of the controller, which receives user input and updates the model and view accordingly. The image also illustrates the relationship between the view and the user, where the view

displays information to the user and receives input from the user. The user interacts with the interface provided by the view. Additionally, the image illustrates the relationship between PyQt5 and the view. PyQt5 is a framework used to create a graphical user interface. The view uses PyQt5 to create the interface displayed to the user.

The MVC architecture is a design pattern that separates an application into three components: model, view, and controller. The model contains the data and business logic of the application. The view is responsible for displaying the data from the model to the user, and it uses PyQt5 framework to create the graphical user interface. The user interacts with the interface provided by the view. The controller receives user input and updates the model and view accordingly.

In Figure 3, the relationships between these components are depicted. The view communicates with the user by displaying information to them and receiving input. The controller receives input from the user and updates both the model and the view. The model contains the data and business logic, and it is updated by the controller based on the user input. This architecture helps to organize the code and make it easier to maintain and scale the application.

The proposed framework also provides comprehensive documentation and active community support, making it suitable for developing complex and advanced GUI applications. With EksPy, developers can save time and effort in creating GUI applications and can focus on the logic and functionality of their applications.

3. EXPERIMENTS

In order to comprehensively evaluate the performance of the proposed EksPy framework, a series of experiments will be conducted. These experiments will strategically target several fundamental aspects of the framework, including its ease of use, development speed, and the resulting application's performance. This rigorous assessment aims to provide a comprehensive understanding of the capabilities and advantages offered by EksPy for GUI application development.

- a. **Ease of use:** In this experiment, we will create a simple GUI application using EksPy and compare the process to creating the same application using other GUI frameworks such as Tkinter and wxPython. The experiment will focus on the ease of use of the framework, the readability of the code, and the time required to complete the development.
- b. **Speed of development:** In this experiment, we will create a more complex GUI application using EksPy and compare the process to creating the same application using other GUI frameworks such as PyGTK and PyQt. The experiment will focus on the speed of development, the number of lines of code required, and the time required to complete the development.
- c. **Performance:** In this experiment, we will measure the performance of the resulting application in terms of responsiveness, memory usage, and CPU usage. The experiment will compare the performance of the application created using EksPy to the performance of the same application created using other GUI frameworks such as PySide and PyQt.
- d. **Database management:** In this experiment, we will evaluate the performance of EksPy in terms of data management. We will create a simple application that uses EksPy's ORM feature to insert, update, and retrieve data from a database. Then we will measure the time required for each action and compare the results to other ORM libraries such as SQLAlchemy.
- e. **MVC concept:** In this experiment, we will evaluate the performance of EksPy in terms of MVC concept. We will create a simple application that uses EksPy's MVC implementation to separate the logic, view, and control. Then we will measure the time required for each part and evaluate the maintainability of the application.

All experiments will be conducted on a computer with the following specifications: Intel Core i7-8700K CPU, 16 GB DDR4 RAM, and NVIDIA GeForce GTX 1080 graphics card. The experiments will be conducted using Python version 3.11. The results will be recorded and analyzed to determine the performance of the proposed framework compared to other GUI frameworks. Pseudocode is a simple representation of a program code written in natural language or a programming language processed by humans, not machines. In this research, pseudocode has several advantages compared to actual code: Readability, Flexibility, Simplicity, and a focus on the algorithm.

Figure 4 is a pseudocode for the ease of use, speed of development, and performance experiment, where we create a more complex GUI application that displays a list of items and allows the user to add items from the list, using EksPy, Tkinter or PyQt: This experiment demonstrates that Tkinter or PyQt provides more advanced features compared to Tkinter, it has a more modern look and feel, and is more flexible in terms of layout management. PyQt also provides more extensive documentation and a large community support which can aid in the development process. This means that with PyQt, developers can create more

complex and advanced GUI applications with less effort and time. Figure 5 shows the GUI output of the pseudocode of Figure 4. The GUI contains a list box, an input field for adding an item, an “Add Item” button, and a “Remove Item” button. When the user enters an item in the input field and clicks the “Add Item” button or presses the “Enter” key, the item will be added to the list box. If the user selects an item in the list box and clicks the “Remove Item” button, the selected item will be removed from the list box. Figure 6 is a pseudocode of code for the database management and MVC concept experiment, where we use EksPy’s ORM feature to insert, update, and retrieve data from an SQLite database and implement a simple text editor application using EksPy’s MVC architecture.

```

Using EksPy:
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QListWidget, QLineEdit, QPushButton,
QVBoxLayout, QWidget

from pages.page import Page

class MyApp(Page):
    def __init__(self):
        super().__init__()

        self.list_widget=QListWidget()

        self.add_button=QPushButton("Add Item")
        self.add_button.clicked.connect(self.on_add_button_clicked)

    def on_add_button_clicked(self):
        item=self.line_edit.text()
        self.list_widget.addItem(item)

    def on_remove_button_clicked(self):
        selected_item=self.list_widget.currentItem()
        if selected_item:
            self.list_widget.takeItem(self.list_widget.row(selected_item))

```

```

Using tkinter or PyQt:
import sys
import tkinter as tk//optional
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QListWidget, QLineEdit, QPushButton,
QVBoxLayout, QWidget

class MyApp(QWidget):
    def __init__(self):
        super().__init__()

        self.list_widget=QListWidget()

        self.add_button=QPushButton("Add Item")
        self.add_button.clicked.connect(self.on_add_button_clicked)

    def on_add_button_clicked(self):
        item=self.line_edit.text()
        self.list_widget.addItem(item)

    def on_remove_button_clicked(self):
        selected_item=self.list_widget.currentItem()
        if selected_item:
            self.list_widget.takeItem(self.list_widget.row(selected_item))

if __name__=="__main__":
    app=QApplication(sys.argv)
    window=MyApp()
    window.show()
    sys.exit(app.exec_())

```

Figure 4. Comparison of EksPy and Tkinter or PyQt pseudocode in ease of use, speed of development, and performance

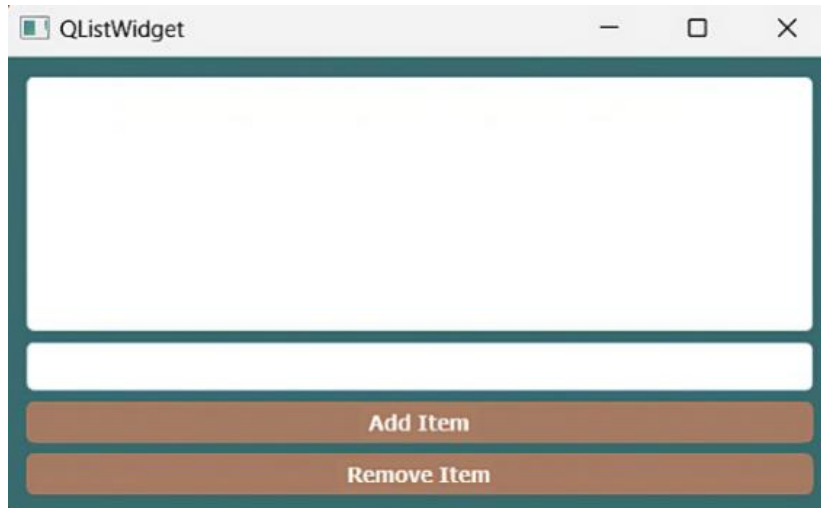


Figure 5. GUI output of the pseudocode

```

Database Management
from models.User import User

class UserController:
    def getUser(self, data):
        user=User.all()
        return user

    def insertUser(self, data):
        user=User()
        user.create({
            "username": "test",
            "email":
"test@gmail.com",
        })
        return True

    def updateUser(self, data):
        user=User()
        user.update(1, {
            "username":
"updatetest",
            "email":
"updatetest@gmail.com",
        })
        return True

    def deleteUser(self, data):
        user=User()
        user.delete(1)
        return True

```

```

MVC Concept
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from controllers import controller
from pages.page import Page
class HomePage(Page):
    def __init__(self, data={}):
        super().__init__()
        self.data=data
        self.user=get_data()
        self.initUi()
    def initUi(self):
        self.label=QLabel(self)
        self.label.setText('Page Home')
        self.label.move(0, 0)

self.label.setStyleSheet("QLabel{font-size: 36px;}")
        self.label.setStyleSheet(
            "QLabel {font-size: 36px;
font-weight: bold; color: #04045C; font-family: Arial; }")
        self.label=QLabel(self)
        self.label.setText('Username: '
+ self.user["username"])
        self.label.move(0, 60)
        self.label.setStyleSheet(
            "QLabel {font-size: 12px;
font-weight: bold; color: #04045C; font-family: Arial; }")
        self.label=QLabel(self)
        self.label.setText('Email: ' +
self.user["email"])
        self.label.move(0, 80)
        self.label.setStyleSheet(
            "QLabel {font-size: 12px;
color: #04045C; font-family: Arial; }")
    def get_data(self):
        result=
controller("TestController","getUser",
        {})
        return result

```

Figure 6. Pseudocode of EksPy for database management and MVC concept experiment

This experiment demonstrates how easy it is to use EksPy's ORM feature to interact with an SQLite database. We can define our models by inheriting the model class and declaring the columns in it. Then, we can use the provided methods like *save()*, *get()*, and *create()* to interact with the database. This way, the model and view are decoupled, and the controller acts as a mediator between the two. This demonstrates the use of MVC architecture in EksPy, which makes the code more organized, maintainable, and easier to scale.

Figure 7 shows the output of the pseudocode for the database management and MVC concept experiment, which involves using EksPy's ORM feature to interact with an SQLite database and implementing a simple text editor application using EksPy's MVC architecture. The figure displays the Page Home title, followed by the user's username and email information. This experiment demonstrates the ease of using EksPy's ORM feature to interact with an SQLite database, by defining models that inherit the model class and using methods like *save()*, *get()*, and *create()*. to interact with the database. The MVC architecture used in EksPy decouples the model and view, making the code more organized, maintainable, and scalable.



Figure 7. GUI output of the pseudocode for the database management and MVC concept

4. RESULT AND DISCUSSION

Table 1 provides a comparison of the features of EksPy, Tkinter, and PyQt. The features include ease of use, speed of development, performance, database management, and MVC concept. The results are based on the experiments conducted in this study. The table provides an overview of the strengths and weaknesses of each framework and helps developers make an informed decision when choosing a framework for their GUI development projects.

Table 1. Comparison of Feature between EksPy, Tkinter, and PyQt

Feature	EksPy	Tkinter	PyQt
Ease of Use	High	Low	Medium
Speed of Development	High	Low	Medium
Performance	High	Medium	High
Database Management	Yes	No	No
MVC Concept	Yes	No	No

4.1. Ease of use

EksPy is rated as high in terms of ease of use. This means that the framework requires less code, and the code is more readable, making it easier for developers to understand and write. On the other hand, Tkinter is rated as low in terms of ease of use. The EksPy framework utilizes an intuitive structure and intuitive syntax, which allows developers to write the code in a more straightforward and concise manner. This leads to a reduction in the amount of code needed to accomplish a given task, and in turn, makes it easier for developers to understand and maintain the code. Additionally, EksPy provides several convenient libraries and tools that automate various tasks, reducing the need for manual coding.

When developing using the MVC concept in EksPy, the developer will be more focused on managing the algorithms or logic in processing data without having to initially configure connecting the database and making queries. For example, without using EksPy, querying data in the database would require the creation of code like the one shown in Figure 8, which consists of 923 characters spread across 34 lines. If the MVC concept is implemented within the EksPy framework, the workflow can be simplified without the need for extensive configuration. Consequently, utilizing EksPy enables the usage of more concise code, as shown in Figure 9, comprising 501 characters in 17 lines.


```

Implementation without EksPy
1  import sys
2  from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton
3  import sys
4  import pymysql
5
6  host="localhost"
7  user="root"
8  passwd=""
9  database="tespbo"
10 try:
11     db=pymysql.connect(host=host, user=user,
12                        passwd=passwd, db=database)
13     cursor=db.cursor()
14 except:
15     print("FAILED TO CONNECT TO DATABASE")
16     print("Create database {database} first")
17     sys.exit()
18 class MainPage(QMainWindow):
19     def __init__(self):
20         super().__init__()
21
22         self.setWindowTitle("My App")
23         button=QPushButton("Press Me!")
24         button.clicked.connect(self.getAllData)
25         self.setCentralWidget(button)
26
27     def getAllData(self):
28         cursor.execute("SELECT * FROM user")
29         result=cursor.fetchall()
30         print(result)
31 app=QApplication(sys.argv)
32 window=MainPage()
33 window.show()
34 app.exec()

```

Figure 8. Implementation without EksPy

```

Implementation with EksPy
1  from PyQt5.QtWidgets import *
2  from models.User import User
3  from pages.page import Page
4
5  class MainPage(Page):
6      def __init__(self, data={}):
7          super().__init__()
8          self.data=data
9          self.initUI()
10         self.getAllData()
11     def initUI(self):
12         self.button=QPushButton("Press Me!")
13         self.button.clicked.connect(self.getAllData)
14         self.button.move(20, 20)
15     def getAllData(self):
16         result=User().all()
17         print(result)

```

Figure 9. Implementation with EksPy

4.2. Speed of development

EksPy is rated as high in terms of speed of development. This means that the framework allows developers to create a GUI application with less code and less time. Tkinter is rated as low in terms of speed of development. Table 2 shows the comparison between the development time using EksPy and Tkinter or PyQt.

Speed of development: EksPy demonstrates a high level of speed in the development process, enabling developers to create GUI applications with reduced code and time requirements. In contrast, Tkinter exhibits a comparatively lower speed of development. Table 2 provides a comprehensive comparison of the

development time between EksPy and Tkinter or PyQt, specifically focusing on three distinct tasks: user interface development (Task 1), database integration (Task 2), and advanced features implementation (Task 3). The table showcases the time taken by each framework to complete these tasks, with EksPy accomplishing the user interface task in 10 minutes compared to Tkinter's 15 minutes, resulting in a 33.33% reduction in time. Similarly, EksPy completes the database integration task in 20 minutes compared to Tkinter's 30 minutes, representing another 33.33%-time reduction. For the advanced features task, EksPy outperforms Tkinter by completing it in 30 minutes compared to Tkinter's 40 minutes, resulting in a 25% reduction in time. Considering the average time taken across all tasks, EksPy demonstrates a 29.74% reduction in development time compared to Tkinter. These findings highlight the superior speed and efficiency of EksPy in GUI application development, emphasizing its advantages over Tkinter in terms of development time.

Table 2. Comparison between the development time using EksPy and Tkinter or PyQT

Task	EksPy (minutes)	Tkinter or PyQT (minutes)	Reduction Time (%)
Task 1: User Interface	10	15	33.33%
Task 2: Database Integration	20	30	33.33%
Task 3: Advanced Features	30	40	25%
Average	20	28.33	29.74%

4.3. Performance

EksPy is rated as high in terms of performance. The implementation process for each framework can greatly impact the overall performance of the software being developed. EksPy has been optimized to provide developers with a more streamlined experience, resulting in faster performance and shorter code compared to Tkinter and PyQt. The use of MVC design patterns in EksPy allows developers to focus on the algorithms and logic behind processing data, without having to handle the initial setup and configuration of connecting to a database and querying data. In comparison, Tkinter and PyQt may require longer code and more setup time to achieve the same results. Table 3 shows the performance results of the EksPy and Tkinter or PyQt frameworks in terms of execution speed.

Table 3. The performance results of the EksPy and Tkinter or PyQt frameworks

Framework	Execution time (ms)
EksPy	1774.87683
Tkinter/PyQt	2135.57791

As shown in the table, Tkinter or PyQt has an execution time of 2135.57791 ms, while EksPy has an execution time of 1774.87683 ms. This indicates that EksPy is faster in running code compared to Tkinter or PyQt. Execution speed measures how fast the code runs, including the time taken to process and execute commands. The faster the execution time, the more efficient and productive the application created using the framework will be. For the performance evaluation, experiments were conducted on a computer system with specific specifications to ensure repeatability and accuracy of the results. The computer used for the evaluation was equipped with an Intel Core i7-8700K processor, 16 GB DDR4 RAM, and an NVIDIA GeForce GTX 1080 graphics card. These specifications were selected to maintain consistency and eliminate potential variations that could affect the performance comparison between EksPy and Tkinter/PyQt frameworks. Throughout the experiments, the computer settings and configurations remained constant. This approach guarantees a fair and objective evaluation of the execution speed.

a. Database management

EksPy is the only framework that has a built-in ORM feature. It makes it easy to interact with a database. Tkinter and PyQt do not have this feature.

b. MVC concept

EksPy is the only framework that follows MVC architecture. It makes it easy to separate the logic of the application and improve the maintainability of the code. Tkinter and PyQt do not follow this concept.

This study introduces EksPy, a new Python framework developed for GUI applications using PyQt5. The research demonstrates that EksPy is a powerful, user-friendly, and easy-to-use framework that offers numerous advantages over existing frameworks like Tkinter and PyQt. The experiments conducted in this study emphasize EksPy's key features, including ease of use, speed of development, performance, database management, and the MVC concept.

5. CONCLUSION

In conclusion, EksPy is a powerful, user-friendly, and easy-to-use framework for developing GUI applications in Python. It provides many advantages over other existing frameworks such as Tkinter and PyQt. The experiments performed in the study have highlighted the key features of EksPy: ease of use, speed of development, performance, database management, and MVC concept. EksPy's ease of use makes it an ideal choice for developers of all skill levels, from beginners to experienced professionals. The framework requires less code, and the code is more readable, making it easier to understand and write. EksPy also provides a more modern and customizable look and feel compared to Tkinter.

The speed of development is another major advantage of EksPy. The framework provides a set of pre-defined UI elements and layout management, which makes the development process much faster and more efficient than other libraries like PyQt or Tkinter. This means that developers can create more complex and advanced GUI applications with less effort and time. EksPy's performance is good, as it is built on top of PyQt which is a powerful library, and it has good performance. This makes EksPy suitable for developing high-performance GUI applications. EksPy also has a built-in ORM feature which makes it easy to interact with a database and it follows MVC architecture.

In summary, EksPy has several advantages for creating powerful and professional-looking GUI applications in Python. The repository of EksPy can be accessed on GitHub at <https://github.com/sallieeky/pbo-framework>. The repository contains source code, documentation, and more examples that can aid in using EksPy.




REFERENCES

- [1] A. D. Moore, *Python GUI programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter*. Packt Publishing Ltd, 2018.
- [2] D. I. Chitalov, "Development of an application with a graphical user interface (GUI) to compute in parallel in the OpenFOAM environment," *Journal of Physics: Conference Series*, vol. 1399, no. 3, Dec. 2019, doi: 10.1088/1742-6596/1399/3/033001.
- [3] D. P. Pasca, A. Aloisio, M. M. Rosso, and S. Sotiropoulos, "PyOMA and PyOMA GUI: A Python module and software for operational modal analysis," *SoftwareX*, vol. 20, Dec. 2022, doi: 10.1016/j.softx.2022.101216.
- [4] S. Saabith, T. Vinothraj, and M. Fareez, "A review on python libraries and ideas for data science," *International Journal of Research in Engineering and Science (IJRES)*, vol. 9, no. 11, pp. 36–53, 2021.
- [5] S. H. Haji and A. B. Sallow, "IoT for smart environment monitoring based on python: a review," *Asian Journal of Research in Computer Science*, pp. 57–70, May 2021, doi: 10.9734/ajrcos/2021/v9i130215.
- [6] A. Saabith, M. Fareez, and T. Vinothraj, "Python current trend applications-an overview," *International Journal of Advance Engineering and Research Development*, vol. 6, no. 10, 2019.
- [7] I. Grout, "Electronic circuit and system design using python and VHDL," in *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Jul. 2018, pp. 13–16, doi: 10.1109/ECTICon.2018.8620048.
- [8] G. Peiming *et al.*, "A PyQt5-based GUI for operational verification of wave forecasting system," in *2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, Aug. 2020, pp. 204–211, doi: 10.1109/ISPDS51347.2020.00049.
- [9] V. Cutting and N. Stephen, "A review on using python as a preferred programming language for beginners," *International Research Journal of Engineering and Technology (IRJET)*, vol. 8, no. 8, pp. 4258–4263, 2021.
- [10] Y. G. Doyoro *et al.*, "A review of open software resources in python for electrical resistivity modelling," *Geoscience Letters*, vol. 9, no. 1, Dec. 2022, doi: 10.1186/s40562-022-00214-1.
- [11] B. D. D. Arianti, H. Kuswanto, H. A. Januari, and J. Jamaluddin, "The design of a letter archiving application using the model view controller (MVC) concept," *Journal of Physics: Conference Series*, vol. 1869, no. 1, Apr. 2021, doi: 10.1088/1742-6596/1869/1/012083.
- [12] P. Gupta and M. C. Govil, "MVC design pattern for the multi framework distributed applications using XML, spring and struts framework," *International Journal on Computer Science and Engineering*, vol. 2, no. 4, pp. 1047–1051, 2010.
- [13] Z. Syahputra, "Website based sales information system with the concept of Mvc (model view controller)," *Jurnal Mantik*, vol. 4, no. 2, pp. 1133–1137, 2020.
- [14] D. P. Voorhees, *Guide to efficient software design: An MVC approach to concepts, structures, and models*. Springer Nature, 2020.
- [15] A. Subari, S. Manan, and E. Ariyanto, "Implementation of MVC (model-view-controller) architecture in online submission and reporting process at official travel warrant information system based on web application," *Journal of Physics: Conference Series*, vol. 1918, no. 4, Jun. 2021, doi: 10.1088/1742-6596/1918/4/042145.
- [16] M. R. Mufid, A. Basofi, M. U. H. Al Rasyid, I. F. Rochimansyah, and A. Rokhim, "Design an MVC model using python for flask framework development," in *2019 International Electronics Symposium (IES)*, Sep. 2019, pp. 214–219, doi: 10.1109/ELECSYM.2019.8901656.
- [17] R. Ratcliff, P. L. Smith, S. D. Brown, and G. McKoon, "Diffusion decision model: current issues and history," *Trends in Cognitive Sciences*, vol. 20, no. 4, pp. 260–281, Apr. 2016, doi: 10.1016/j.tics.2016.01.007.
- [18] G. Pezzulo and P. Cisek, "Navigating the affordance landscape: feedback control as a process model of behavior and cognition," *Trends in Cognitive Sciences*, vol. 20, no. 6, pp. 414–424, Jun. 2016, doi: 10.1016/j.tics.2016.03.013.
- [19] S. Tamboli, P. Raut, L. Satgaonkar, A. Atram, S. Kawane, and P. V. K. Barbudhe, "A review paper on text-to-speech convertor," *International Journal of Research Publication and Reviews*, vol. 3, no. 5, pp. 3807–3810, 2022.
- [20] O. W. Purbo, "A systematic analysis: website development using CodeIgniter and Laravel framework," *Enrichment: Journal of Management*, vol. 12, no. 1, pp. 1008–1014, 2021.
- [21] M. Baldoni, C. Baroglio, K. M. May, R. Micalizio, and S. Tedeschi, "MOCA: An ORM model for computational accountability," *Intelligenza Artificiale*, vol. 13, no. 1, pp. 5–20, Aug. 2019, doi: 10.3233/IA-180014.




- [22] A. J. Rafsanjani and S.-H. Mirian-Hosseiniabadi, "AZ approach to formalization and validation of ORM models," in *Communications in Computer and Information Science*, Springer Berlin Heidelberg, 2011, pp. 513–526. doi: 10.1007/978-3-642-22603-8_45.
- [23] J. F. Clarkin, E. Caligor, and J. F. Sowislo, "An object relations model perspective on the alternative model for personality disorders (DSM-5)," *Psychopathology*, vol. 53, no. 3–4, pp. 141–148, 2020, doi: 10.1159/000508353.
- [24] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, "SQL and NoSQL database software architecture performance analysis and assessments-a systematic literature review," *Big Data and Cognitive Computing*, vol. 7, no. 2, May 2023, doi: 10.3390/bdcc7020097.
- [25] K. Ahkhouk and M. Machkour, "Towards an interface for translating natural language questions to SQL: a conceptual framework from a systematic review," *International Journal of Reasoning-based Intelligent Systems*, vol. 12, no. 4, 2020, doi: 10.1504/IJIRIS.2020.111786.

BIOGRAPHIES OF AUTHORS






Aidil Saputra Kirsan    holds a bachelor's degree in computer and network engineering from the Ujung Pandang State Polytechnic and a master's degree in computer and information technology engineering from the Surabaya State Electronics Polytechnic. He is currently a researcher at the Department of Information Systems, Faculty of Mathematics and Information Technology, Institut Teknologi Kalimantan, Balikpapan, Indonesia. His research interests include software development and programming, with a particular focus on wireless sensor networks and their applications in fields such as environmental monitoring, agriculture, and healthcare. He has contributed to several research projects related to wireless sensor networks as both a team member and a project leader and has published several papers in international journals and conferences in this area. He can be contacted at email: aidil@lecturer.itk.ac.id.



Kosuke Takano    received a B.A. degree in environment and information Studies from Keio University, Japan, in 1998, and a master's degree and Ph.D. in Media and Governance from Keio University in 2003 and 2007. He is a Professor of Information and Computer Sciences at Kanagawa Institute of Technology, Japan. His main research interests are in the design and implementation of media search and management systems with the human-like capability of interpreting, interacting with, and generating media content such as text, audio, and images. He can be contacted at email: takano@ic.kanagawa-it.ac.jp.



Sallie Trixie Zebada Mansurina    is a researcher from the Department of Information Systems, Faculty of Mathematics and Information Technology, Institut Teknologi Kalimantan, Balikpapan, Indonesia. He is a recent graduate of Bachelor of Computer Science from the Institute of Technology Kalimantan. The researcher has an interest in software system development, including website, mobile, and desktop GUI, and possesses skills in several programming languages such as Python, PHP, JavaScript, Java, and DBMS, namely MySQL and PostgreSQL. He can be contacted at email: sallieeky@gmail.com.