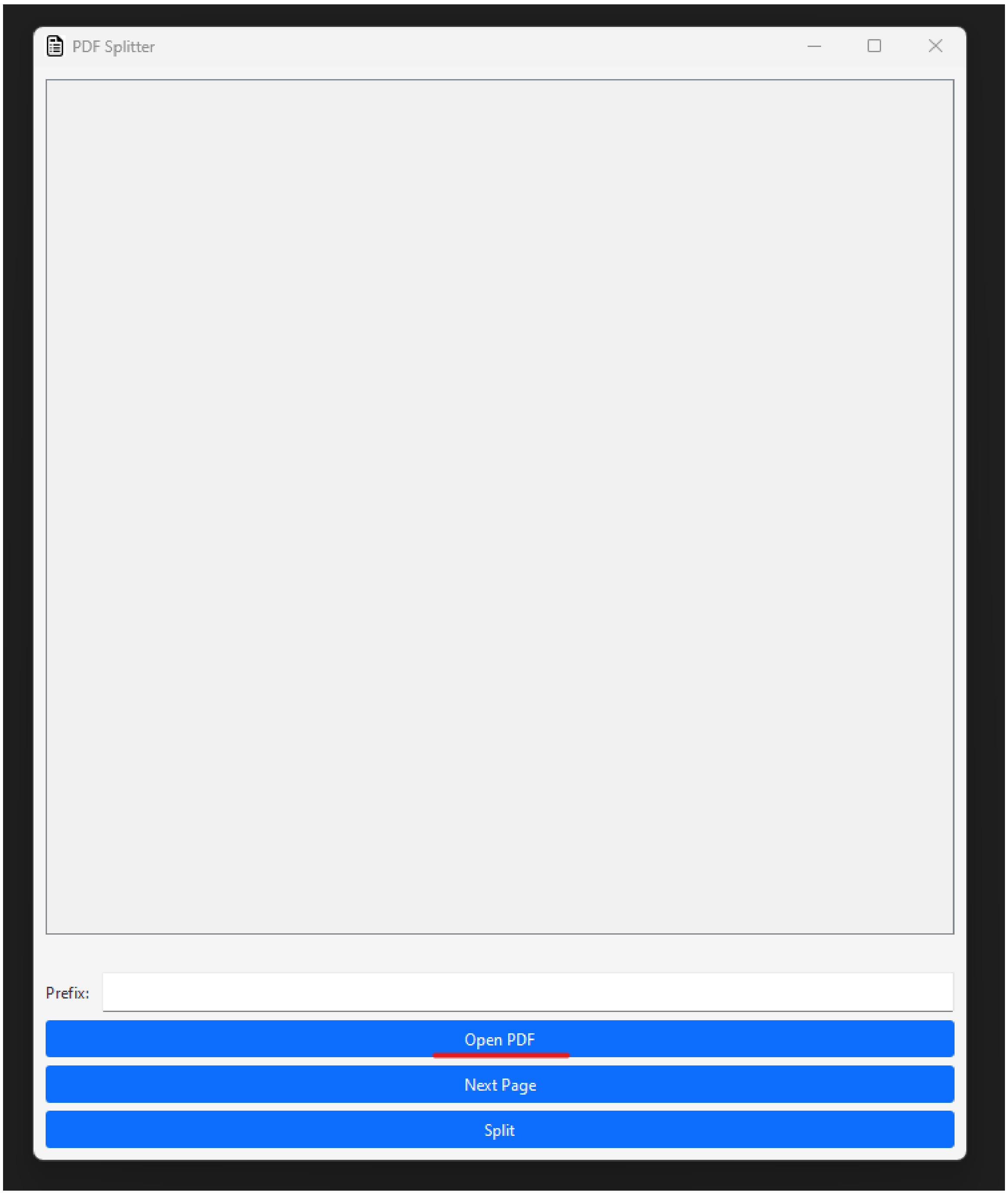
Documentation on **PDF SPLITTER**

Summary

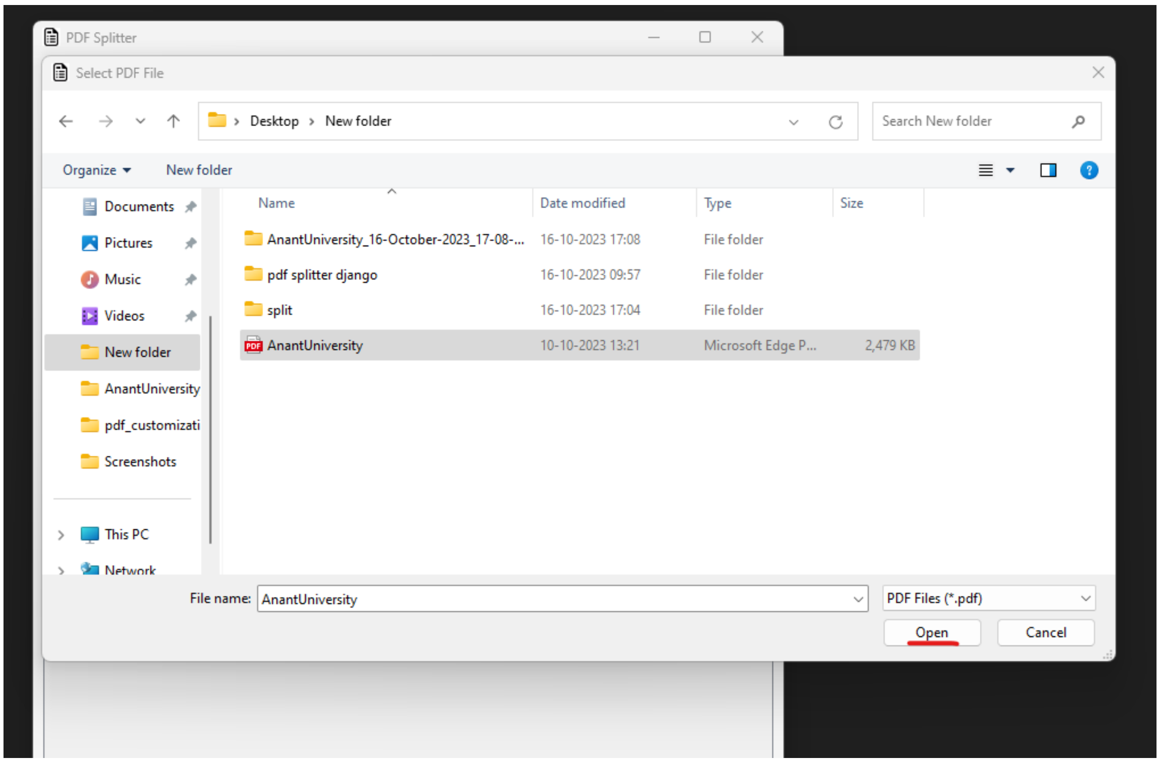
PDF Splitter operates by taking a multi-page PDF, splitting each page into its own PDF file, and naming these files according to the text coordinates set for each page.

This application opens a PDF, also let's the user to select a page, and then allows the user to define coordinates by dragging the mouse cursor over unique text on every page. Once the coordinates are set, it uses the unique text to rename the split PDF files.

Below Steps Define how to use this Application



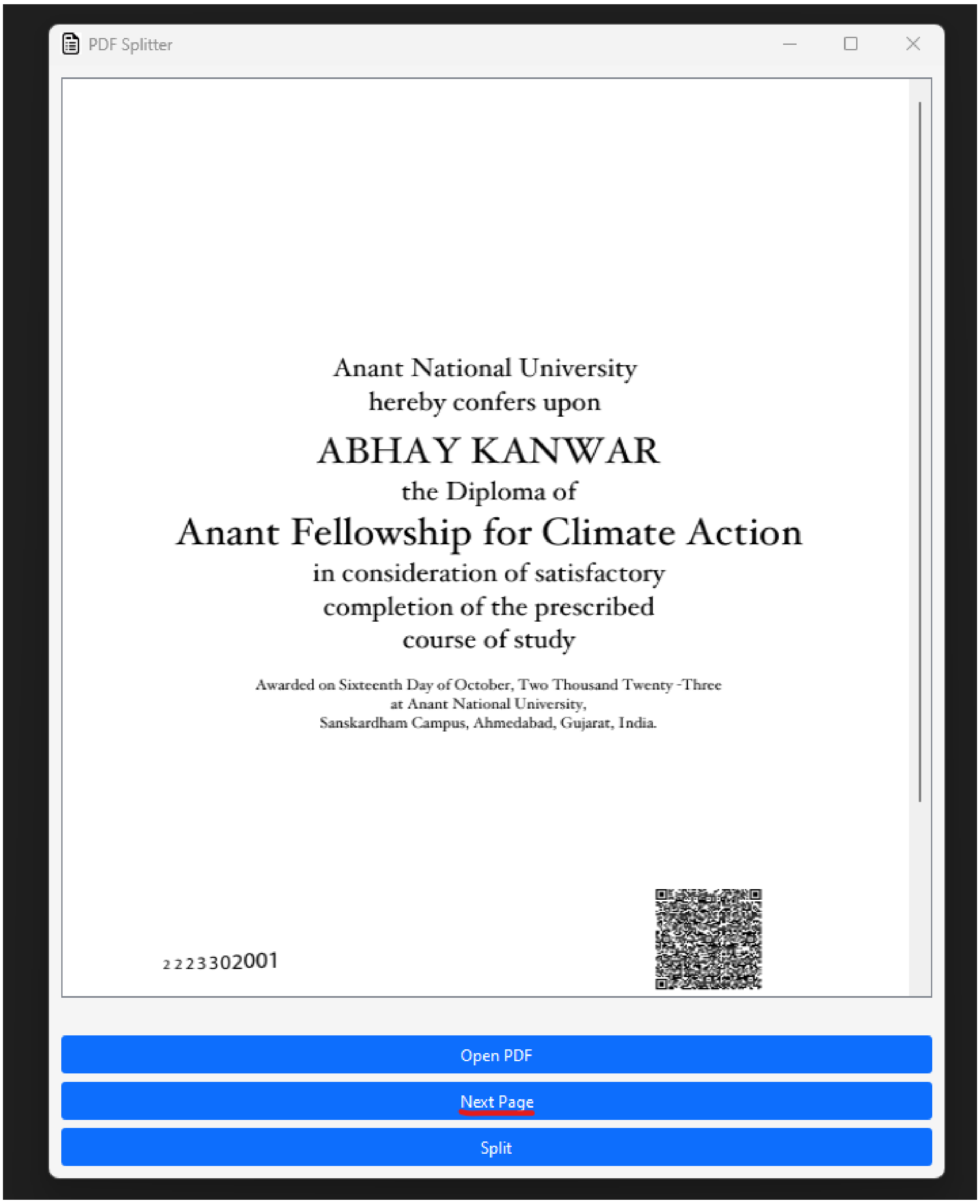
1. When you Clicked on the application, This window get's open. Then clicked on " Open PDF " button.



Name	Date modified	Type	Size
AnantUniversity_16-October-2023_17-08-...	16-10-2023 17:08	File folder	
pdf splitter django	16-10-2023 09:57	File folder	
split	16-10-2023 17:04	File folder	
AnantUniversity	10-10-2023 13:21	Microsoft Edge P...	2,479 KB

File name: AnantUniversity
PDF Files (*.pdf)

2. Then a File Dailog will get open, where you can select a pdf file only. Then Choose the pdf file on which you want to split the pdf and then select open.

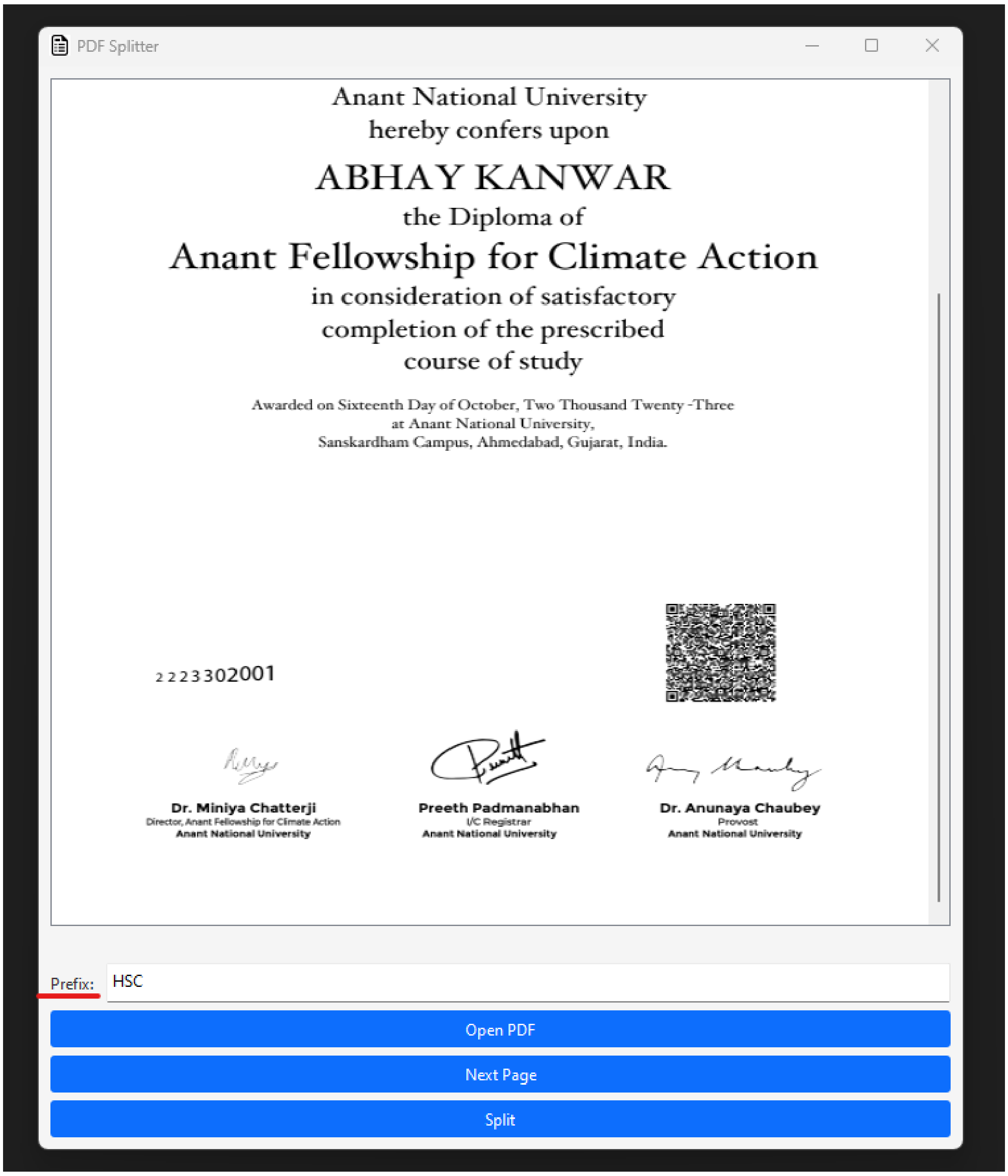


Anant National University
hereby confers upon
ABHAY KANWAR
the Diploma of
Anant Fellowship for Climate Action
in consideration of satisfactory
completion of the prescribed
course of study

Awarded on Sixteenth Day of October, Two Thousand Twenty -Three
at Anant National University,
Sanskardham Campus, Ahmedabad, Gujarat, India.

2223302001

3. Then First page of pdf will get Open, you can also change the page by clicking on Next Page.



Anant National University
hereby confers upon
ABHAY KANWAR
the Diploma of
Anant Fellowship for Climate Action
in consideration of satisfactory
completion of the prescribed
course of study

Awarded on Sixteenth Day of October, Two Thousand Twenty -Three
at Anant National University,
Sanskardham Campus, Ahmedabad, Gujarat, India.

2223302001

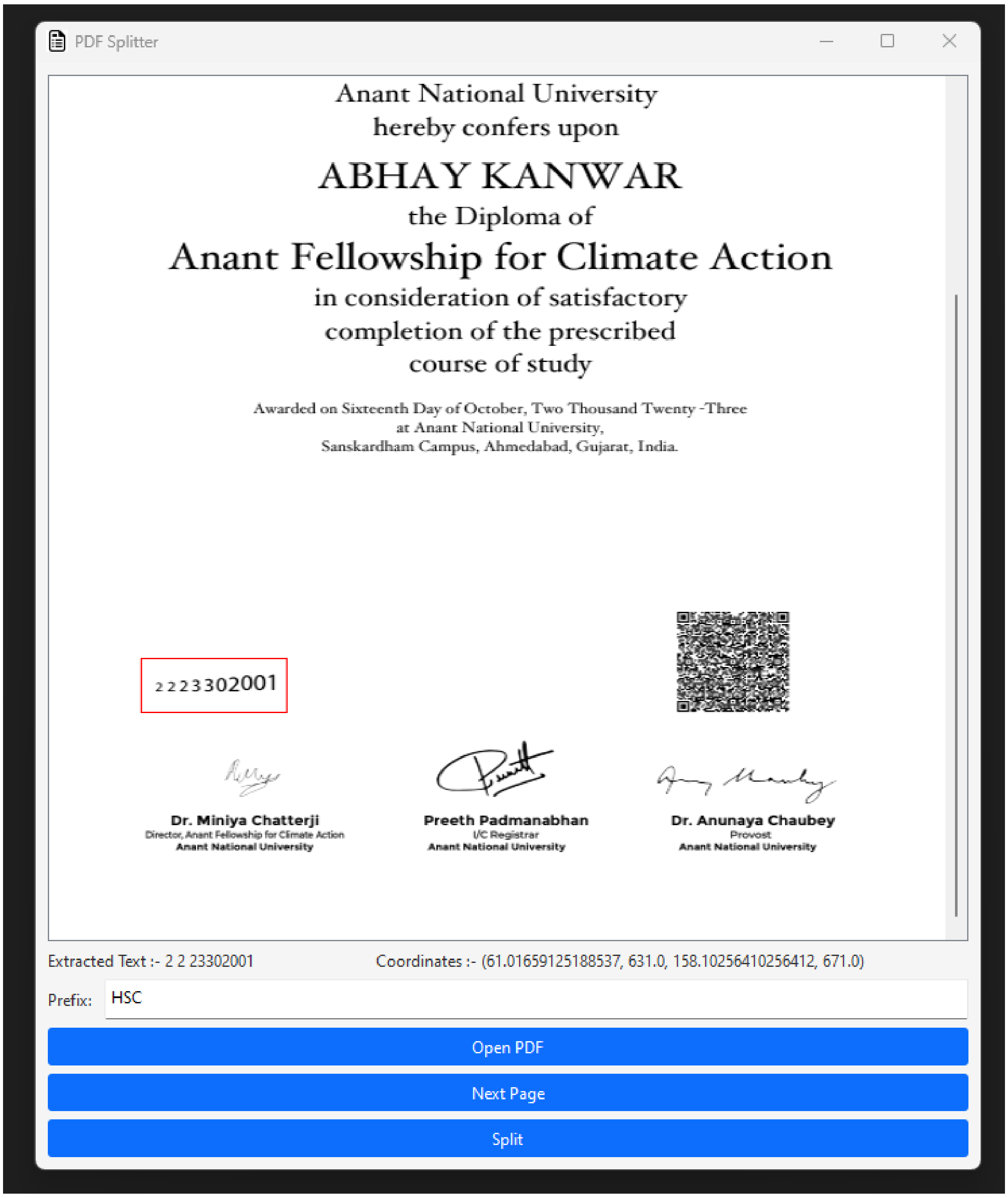
Dr. Miniya Chatterji
Director, Anant Fellowship for Climate Action
Anant National University

Preeth Padmanabhan
I/C Registrar
Anant National University

Dr. Anunaya Chaubey
Provost
Anant National University

Prefix: HSC

4. If you want to add prefix and want to save your split pdf file as the text in the prefix and the text from the set coordinates you should enter in prefix input.
5. And even if you not enter anything it will just save the file name based on the text extracted from the set coordinates.



Anant National University
hereby confers upon
ABHAY KANWAR
the Diploma of
Anant Fellowship for Climate Action
in consideration of satisfactory
completion of the prescribed
course of study

Awarded on Sixteenth Day of October, Two Thousand Twenty -Three
at Anant National University,
Sanskardham Campus, Ahmedabad, Gujarat, India.

2223302001

Dr. Miniya Chatterji
Director, Anant Fellowship for Climate Action
Anant National University

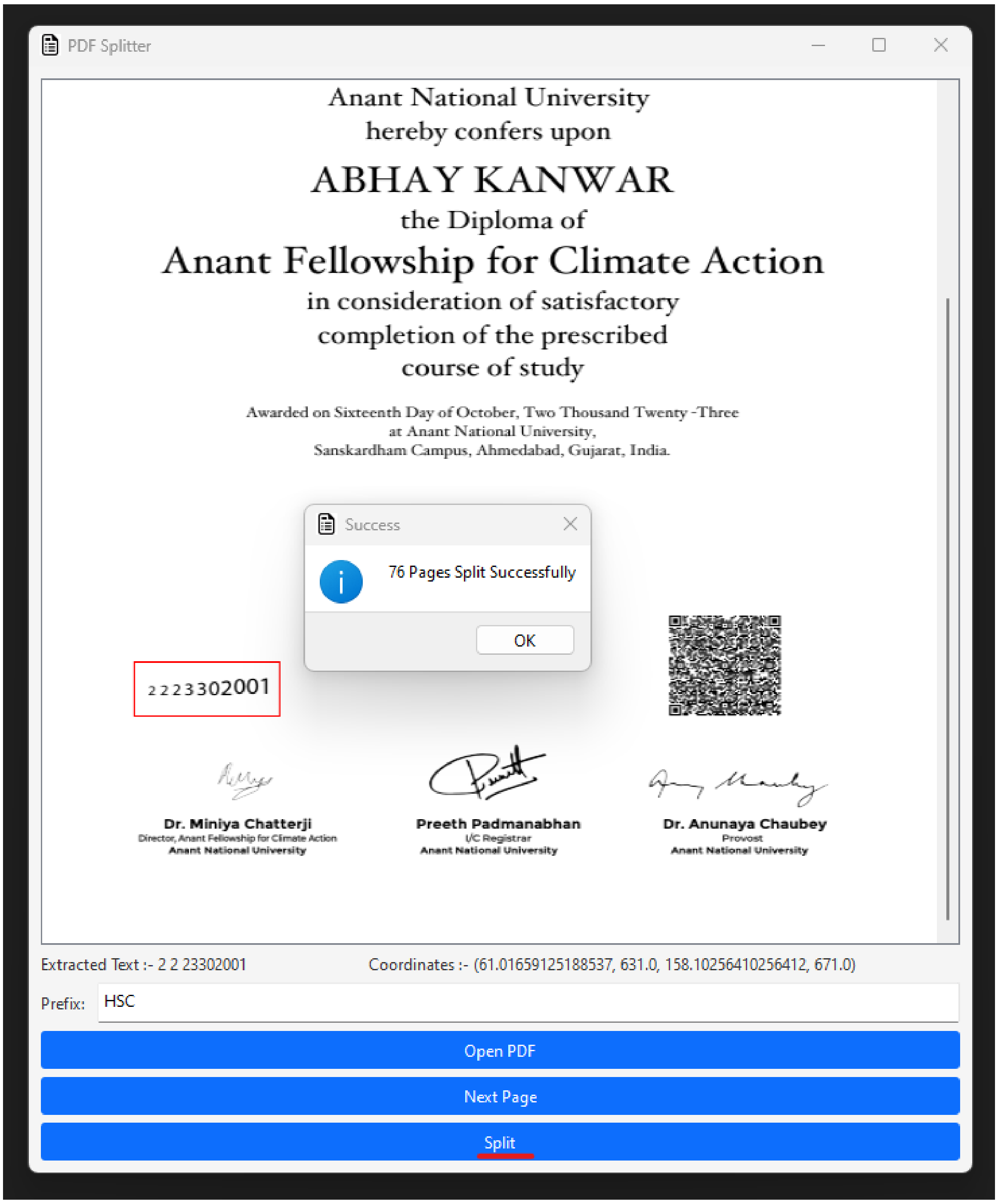
Preeth Padmanabhan
I/C Registrar
Anant National University

Dr. Anunaya Chaubey
Provost
Anant National University

Extracted Text :- 2 2 23302001 Coordinates :- (61.01659125188537, 631.0, 158.10256410256412, 671.0)

Prefix: HSC

6. Then Drag your mouse over the unique text which will be there at same position on every page. Once your Drag is complete, You can see the extracted text and cordinates on above the prefix input.
7. You just cross verify the text, if you dont get the exact text then drag your mouse over the text.



Anant National University
hereby confers upon
ABHAY KANWAR
the Diploma of
Anant Fellowship for Climate Action
in consideration of satisfactory
completion of the prescribed
course of study

Awarded on Sixteenth Day of October, Two Thousand Twenty -Three
at Anant National University,
Sanskardham Campus, Ahmedabad, Gujarat, India.

2223302001

Success

76 Pages Split Successfully

OK

Dr. Miniya Chatterji
Director, Anant Fellowship for Climate Action
Anant National University

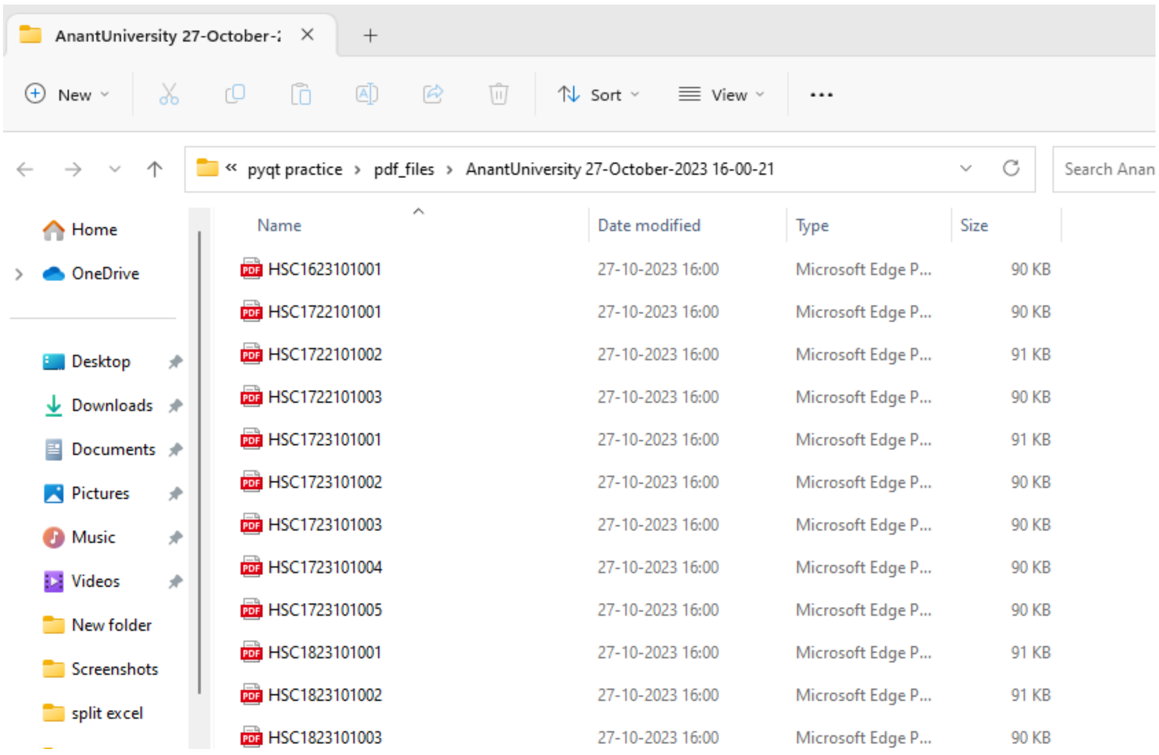
Preeth Padmanabhan
I/C Registrar
Anant National University

Dr. Anunaya Chaubey
Provost
Anant National University

Extracted Text :- 2 2 23302001 Coordinates :- (61.01659125188537, 631.0, 158.10256410256412, 671.0)

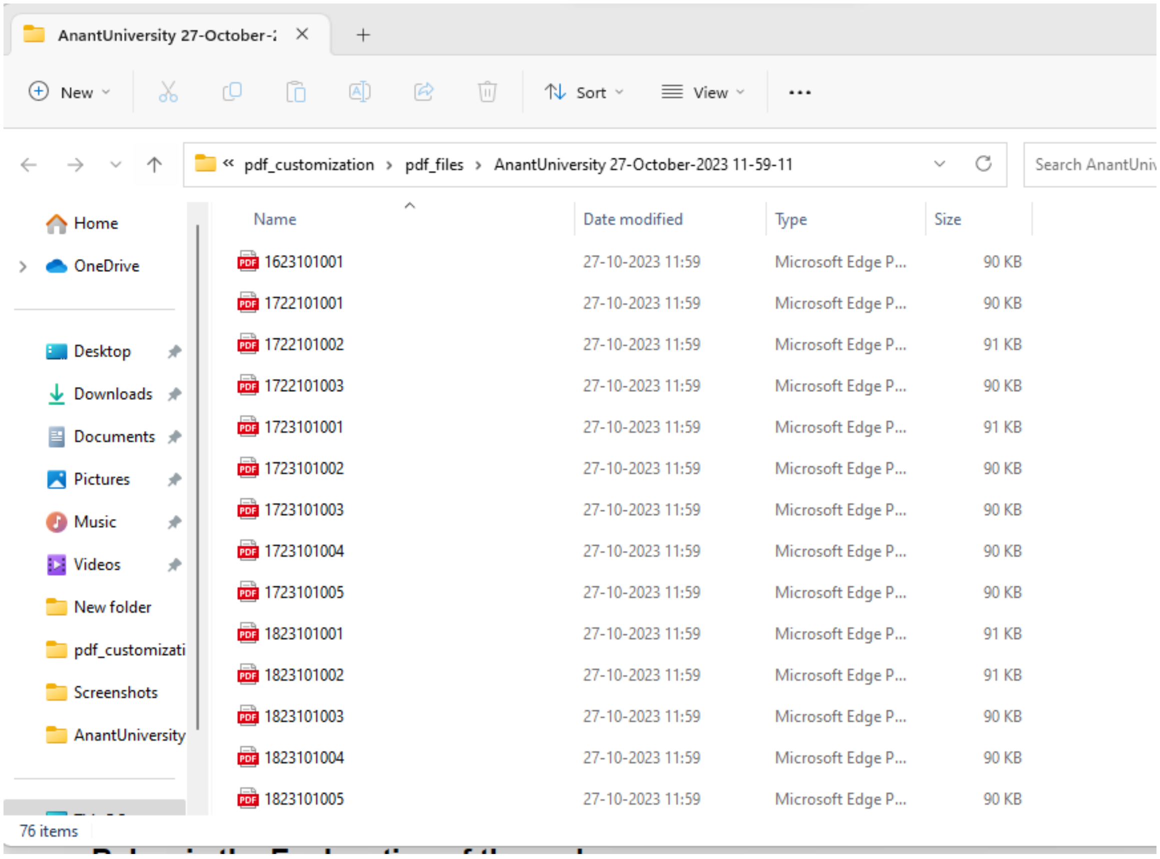
Prefix: HSC

8. Click on Split button. Then a Process of Splitting PDF will start and you will get to see a spinner rolling on your screen. Once the Process get's completed you will get a success message.



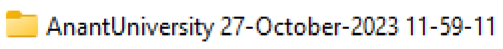
Name	Date modified	Type	Size
HSC1623101001	27-10-2023 16:00	Microsoft Edge P...	90 KB
HSC1722101001	27-10-2023 16:00	Microsoft Edge P...	90 KB
HSC1722101002	27-10-2023 16:00	Microsoft Edge P...	91 KB
HSC1722101003	27-10-2023 16:00	Microsoft Edge P...	90 KB
HSC1723101001	27-10-2023 16:00	Microsoft Edge P...	91 KB
HSC1723101002	27-10-2023 16:00	Microsoft Edge P...	90 KB
HSC1723101003	27-10-2023 16:00	Microsoft Edge P...	90 KB
HSC1723101004	27-10-2023 16:00	Microsoft Edge P...	90 KB
HSC1723101005	27-10-2023 16:00	Microsoft Edge P...	90 KB
HSC1823101001	27-10-2023 16:00	Microsoft Edge P...	91 KB
HSC1823101002	27-10-2023 16:00	Microsoft Edge P...	91 KB
HSC1823101003	27-10-2023 16:00	Microsoft Edge P...	90 KB

If not use prefix. PDF files will be saved as



Name	Date modified	Type	Size
1623101001	27-10-2023 11:59	Microsoft Edge P...	90 KB
1722101001	27-10-2023 11:59	Microsoft Edge P...	90 KB
1722101002	27-10-2023 11:59	Microsoft Edge P...	91 KB
1722101003	27-10-2023 11:59	Microsoft Edge P...	90 KB
1723101001	27-10-2023 11:59	Microsoft Edge P...	91 KB
1723101002	27-10-2023 11:59	Microsoft Edge P...	90 KB
1723101003	27-10-2023 11:59	Microsoft Edge P...	90 KB
1723101004	27-10-2023 11:59	Microsoft Edge P...	90 KB
1723101005	27-10-2023 11:59	Microsoft Edge P...	90 KB
1823101001	27-10-2023 11:59	Microsoft Edge P...	91 KB
1823101002	27-10-2023 11:59	Microsoft Edge P...	91 KB
1823101003	27-10-2023 11:59	Microsoft Edge P...	90 KB
1823101004	27-10-2023 11:59	Microsoft Edge P...	90 KB
1823101005	27-10-2023 11:59	Microsoft Edge P...	90 KB

9. Once you clicked on Ok it will redirect you to the folder, where you can see the PDF which has been splitted.



10. Folder where your PDF are stored is named as, File name of pdf which you opened, then todays Date and Time Format.

Below is the Explanation of the code

```
import os
import fitz
import datetime
import subprocess
from PySide6.QtWidgets import QMainWindow, QApplication, QWidget, QScrollArea, QVBoxLayout, QMessageBox
from PySide6.QtWidgets import QPushButton, QLabel, QFileDialog, QMainWindow, QDialog, QTextEdit, QHBoxLayout
from PySide6.QtGui import QPixmap, QCursor, QPainter, QColor, QImage, QIcon, QMovie
from PySide6.QtCore import Qt, QRect,  QByteArray
```

Here, you import various libraries and modules that are used throughout your application. Notable libraries include fitz for working with PDFs, PySide6 for the user interface, and others for general functionality.

```
base64_image = b"iVBORw0KGgoAAAANSUhEUgAAAOEAAADhCAMAAAAJbSJIAAAAk1BMVEX///8EAgQAAADn5+eEhISgoKCKi4rX2NfFxcWUlZRmZmYoKSh
```

This line defines a base64-encoded image that represents your application's logo.

```
class ExtractedTextWindow(QMainWindow):
```

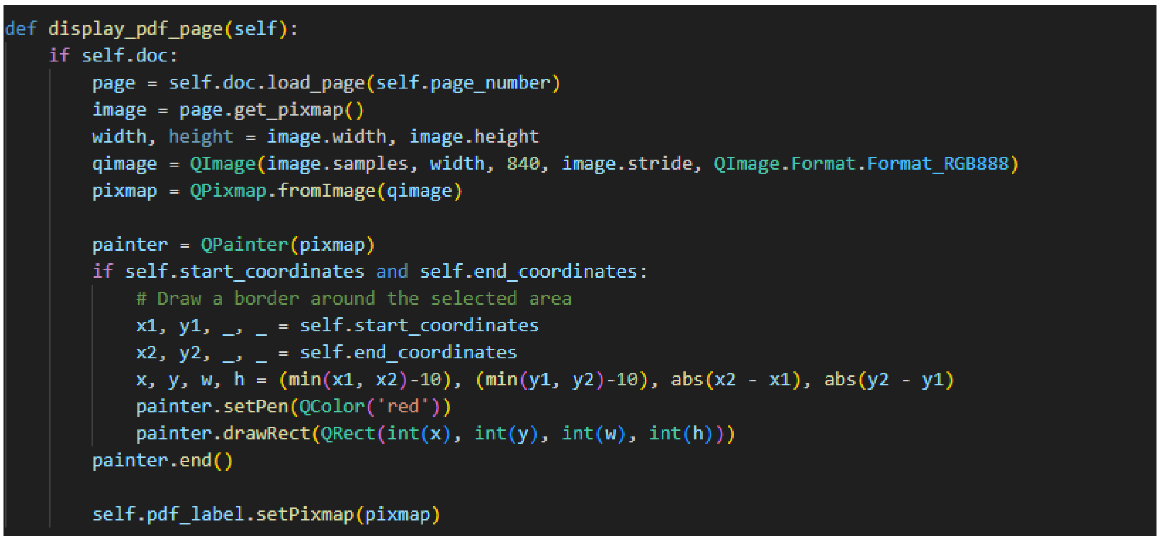
Here, you define a class for the main application window, which is derived from QMainWindow. This class contains all the functionality of your application.

```
def __init__(self):
    super().__init__()
    self.init_ui()
    self.doc = None
    self.page_number = 0
    self.start_coordinates = None
    self.end_coordinates = None
    self.selection_rect_item = None
```

In the constructor **init**, you initialize the window and some instance variables. init_ui() is a method that sets up the user interface.

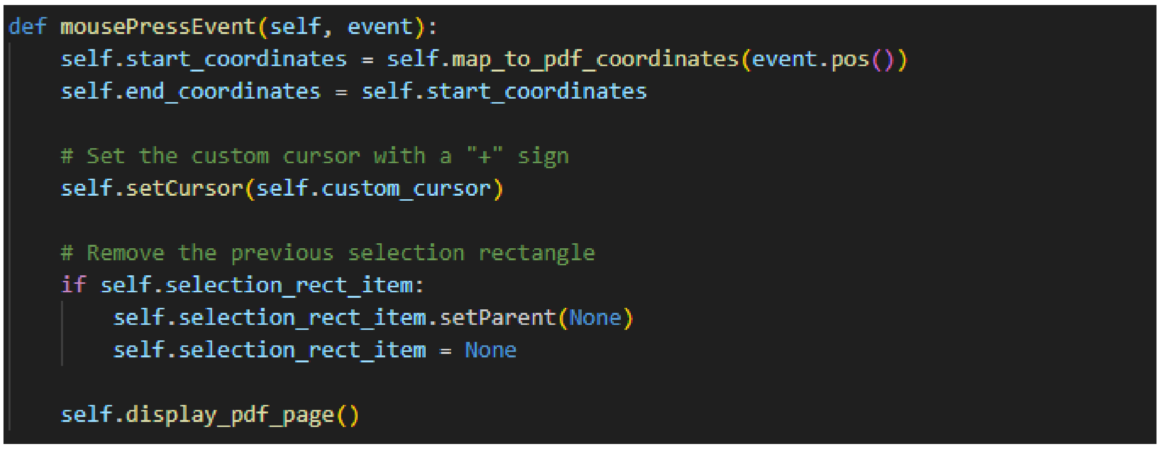
```
def open_pdf_and_display(self):
    dialog = QFileDialog(self)
    options = dialog.options()
    file_path, _ = dialog.getOpenFileName(self, "Select PDF File", "", "PDF Files (*.pdf);;All Files (*)", options=options)
    if file_path:
        self.selected_pdf_path = file_path
        self.doc = fitz.open(file_path)
        self.page_number = 0
        self.display_pdf_page()
```

This code opens a file dialog, lets the user select a PDF file, and if a file is selected, it stores the file path, opens the PDF, and sets the page number to 0. It then proceeds to display the selected page from the PDF on the application's interface.



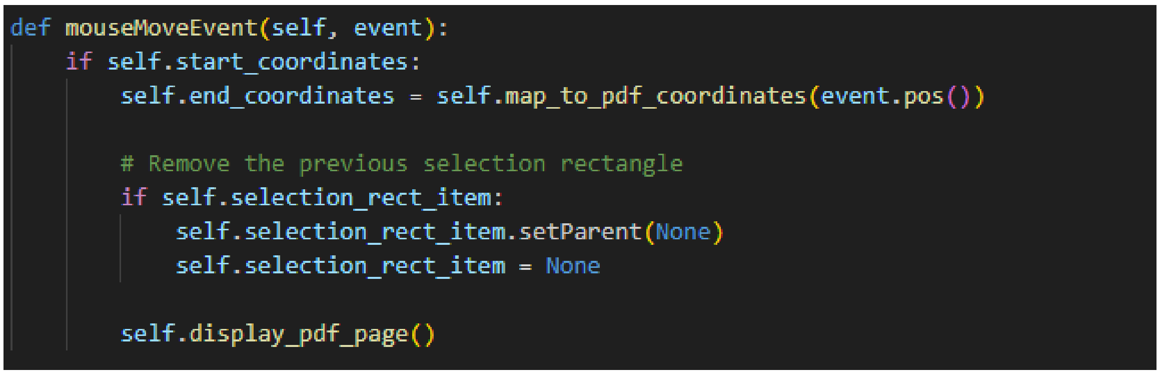
```
def display_pdf_page(self):
    if self.doc:
        page = self.doc.load_page(self.page_number)
        image = page.get_pixmap()
        width, height = image.width, image.height
        qimage = QImage(image.samples, width, 840, image.stride, QImage.Format.Format_RGB888)
        pixmap = QPixmap.fromImage(qimage)

        painter = QPainter(pixmap)
        if self.start_coordinates and self.end_coordinates:
            # Draw a border around the selected area
            x1, y1, _, _ = self.start_coordinates
            x2, y2, _, _ = self.end_coordinates
            x, y, w, h = (min(x1, x2)-10), (min(y1, y2)-10), abs(x2 - x1), abs(y2 - y1)
            painter.setPen(QColor('red'))
            painter.drawRect(QRect(int(x), int(y), int(w), int(h)))
        painter.end()

        self.pdf_label.setPixmap(pixmap)
```

This method displays a page from a PDF in the application's interface. It loads the PDF page, converts it to an image, and then displays it with the option to draw a red border around a selected area. The resulting image is shown in the interface using a QLabel widget.



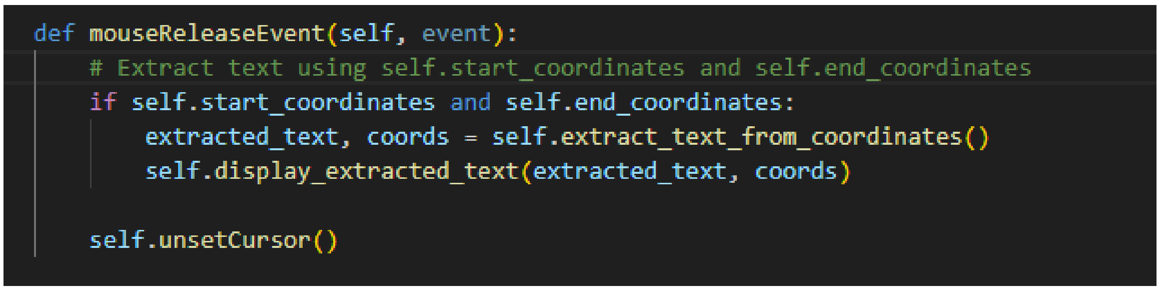
```
def mousePressEvent(self, event):
    self.start_coordinates = self.map_to_pdf_coordinates(event.pos())
    self.end_coordinates = self.start_coordinates

    # Set the custom cursor with a "+" sign
    self.setCursor(self.custom_cursor)

    # Remove the previous selection rectangle
    if self.selection_rect_item:
        self.selection_rect_item.setParent(None)
        self.selection_rect_item = None

    self.display_pdf_page()
```

This method responds to a mouse click event by setting the starting and ending coordinates for a selection rectangle. It also changes the mouse cursor to a custom cursor with a "+" sign, removes any previous selection rectangle, and updates the displayed page in response to the user's click.



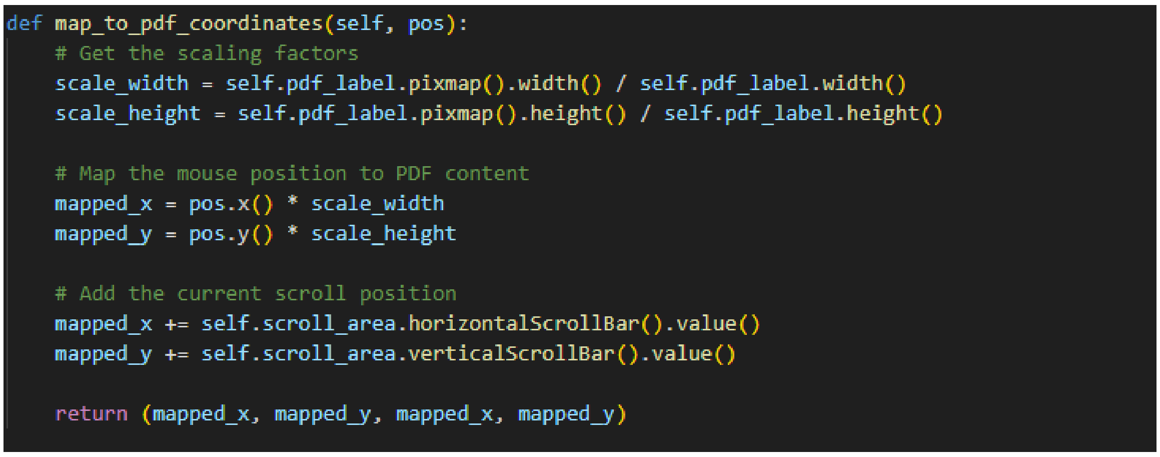
```
def mouseMoveEvent(self, event):
    if self.start_coordinates:
        self.end_coordinates = self.map_to_pdf_coordinates(event.pos())

        # Remove the previous selection rectangle
        if self.selection_rect_item:
            self.selection_rect_item.setParent(None)
            self.selection_rect_item = None

        self.display_pdf_page()
```

This method handles mouse movement events. It updates the ending coordinates of a selection rectangle based on the mouse cursor's position, removes any previous selection rectangle, and refreshes the displayed page accordingly.



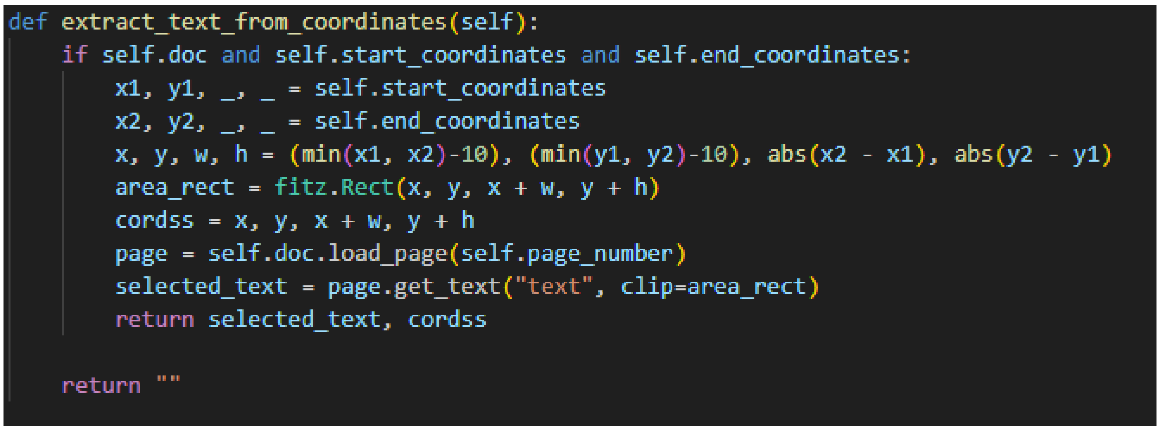
```python
def mouseReleaseEvent(self, event):
    # Extract text using self.start_coordinates and self.end_coordinates
    if self.start_coordinates and self.end_coordinates:
        extracted_text, coords = self.extract_text_from_coordinates()
        self.display_extracted_text(extracted_text, coords)

    self.unsetCursor()
```

This method handles mouse events, like once you release your mouse by dragging on text, it will extract the text and set the coordinates and then it will display the text and the coordinates.



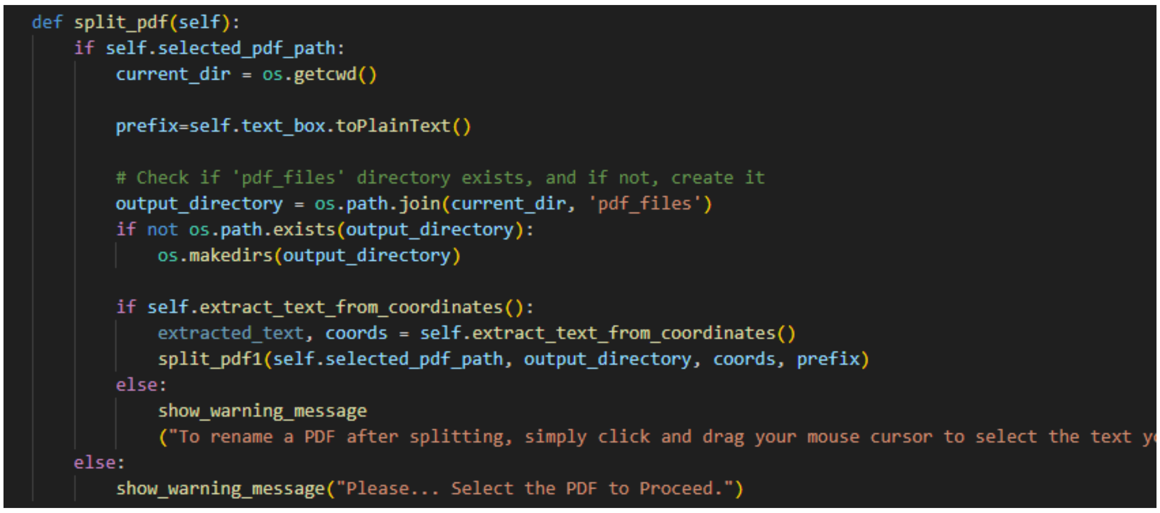
```python
def map_to_pdf_coordinates(self, pos):
    # Get the scaling factors
    scale_width = self.pdf_label.pixmap().width() / self.pdf_label.width()
    scale_height = self.pdf_label.pixmap().height() / self.pdf_label.height()

    # Map the mouse position to PDF content
    mapped_x = pos.x() * scale_width
    mapped_y = pos.y() * scale_height

    # Add the current scroll position
    mapped_x += self.scroll_area.horizontalScrollBar().value()
    mapped_y += self.scroll_area.verticalScrollBar().value()

    return (mapped_x, mapped_y, mapped_x, mapped_y)
```

This method serves the essential purpose of mapping the mouse cursor's position from the user interface coordinates to coordinates within the PDF content, taking into account any scaling and scrolling that the user might have applied. This ensures that the selection area is accurate and properly aligned with the content even when the user interacts with the PDF by zooming in, expanding the screen, or scrolling.



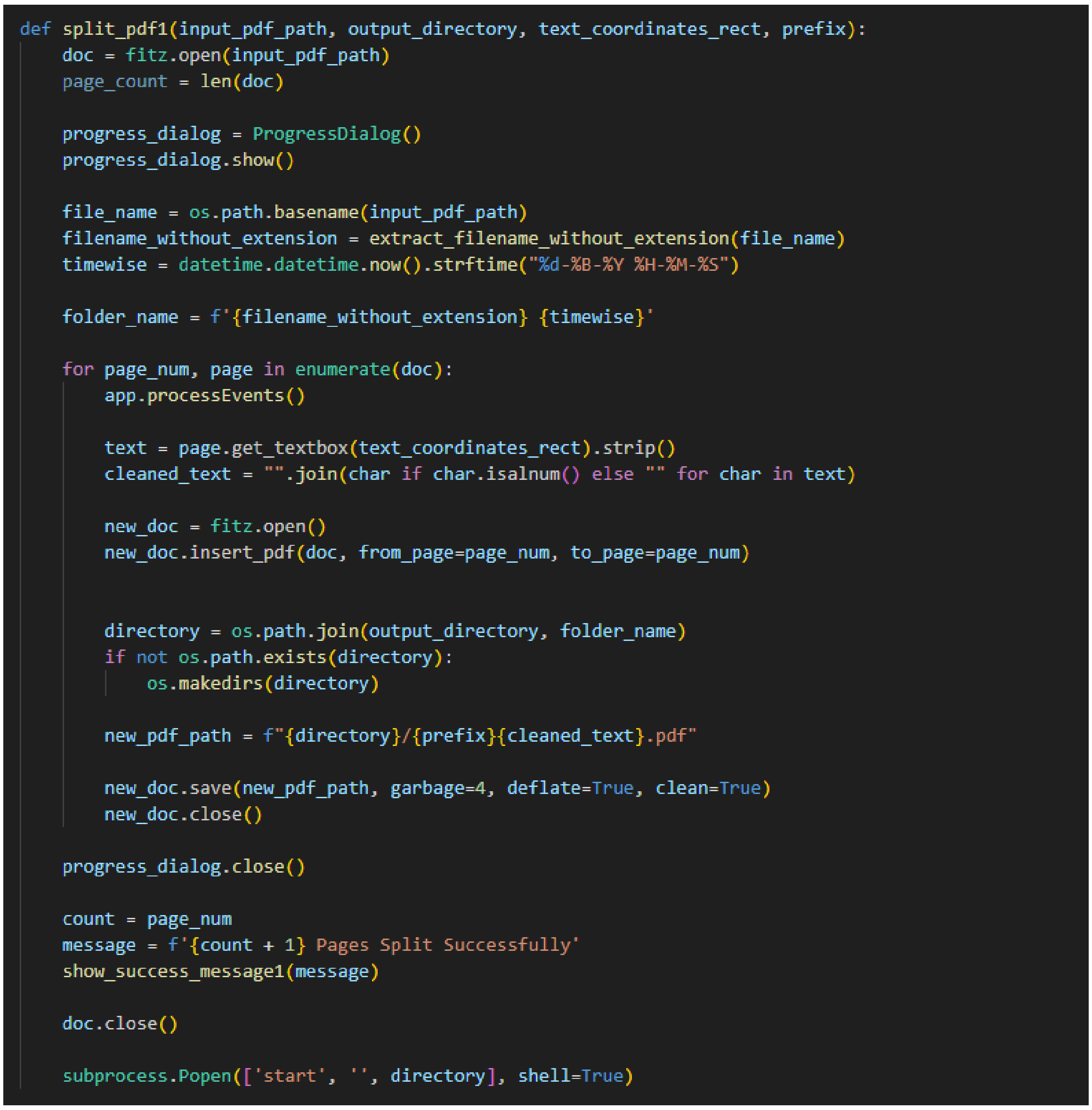
```python
def extract_text_from_coordinates(self):
    if self.doc and self.start_coordinates and self.end_coordinates:
        x1, y1, _, _ = self.start_coordinates
        x2, y2, _, _ = self.end_coordinates
        x, y, w, h = (min(x1, x2)-10), (min(y1, y2)-10), abs(x2 - x1), abs(y2 - y1)
        area_rect = fitz.Rect(x, y, x + w, y + h)
        cordss = x, y, x + w, y + h
        page = self.doc.load_page(self.page_number)
        selected_text = page.get_text("text", clip=area_rect)
        return selected_text, cordss

    return ""
```

This Method extract text from a defined rectangular area within a PDF page, considering specified coordinates. It returns the extracted text and the coordinates of the selected area.



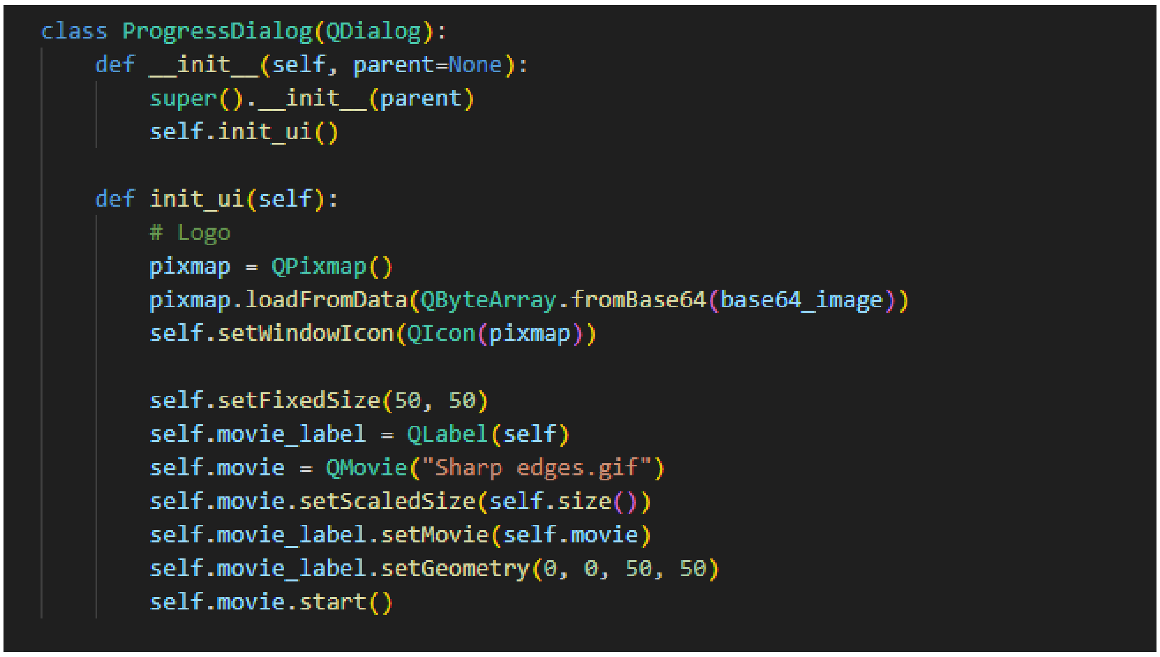
```python
def split_pdf(self):
    if self.selected_pdf_path:
        current_dir = os.getcwd()

        prefix=self.text_box.toPlainText()

        # Check if 'pdf_files' directory exists, and if not, create it
        output_directory = os.path.join(current_dir, 'pdf_files')
        if not os.path.exists(output_directory):
            os.makedirs(output_directory)

        if self.extract_text_from_coordinates():
            extracted_text, coords = self.extract_text_from_coordinates()
            split_pdf1(self.selected_pdf_path, output_directory, coords, prefix)
        else:
            show_warning_message
            ("To rename a PDF after splitting, simply click and drag your mouse cursor to select the text y
    else:
        show_warning_message("Please... Select the PDF to Proceed.")
```

- It checks if a PDF file has been selected (self.selected_pdf_path).
- It determines the current directory and creates an "pdf_files" directory if it doesn't exist.
- If text coordinates are available from the extract_text_from_coordinates function, it extracts the text and coordinates and uses them to split and rename the PDF pages using a function called split_pdf1.
- If text coordinates are not available, it displays a warning message to guide the user on how to select text for renaming.
- If no PDF file has been selected, it prompts the user to select one.



```python
def split_pdf1(input_pdf_path, output_directory, text_coordinates_rect, prefix):
    doc = fitz.open(input_pdf_path)
    page_count = len(doc)

    progress_dialog = ProgressDialog()
    progress_dialog.show()

    file_name = os.path.basename(input_pdf_path)
    filename_without_extension = extract_filename_without_extension(file_name)
    timewise = datetime.datetime.now().strftime("%d-%B-%Y %H-%M-%S")

    folder_name = f'{filename_without_extension} {timewise}'

    for page_num, page in enumerate(doc):
        app.processEvents()

        text = page.get_textbox(text_coordinates_rect).strip()
        cleaned_text = "".join(char if char.isalnum() else "" for char in text)

        new_doc = fitz.open()
        new_doc.insert_pdf(doc, from_page=page_num, to_page=page_num)


        directory = os.path.join(output_directory, folder_name)
        if not os.path.exists(directory):
            os.makedirs(directory)

        new_pdf_path = f"{directory}/{prefix}{cleaned_text}.pdf"

        new_doc.save(new_pdf_path, garbage=4, deflate=True, clean=True)
        new_doc.close()

    progress_dialog.close()

    count = page_num
    message = f'{count + 1} Pages Split Successfully'
    show_success_message1(message)

    doc.close()

    subprocess.Popen(['start', '', directory], shell=True)
```

- It opens the input PDF file using the fitz module (presumably PyMuPDF).
- It determines the total number of pages in the PDF and then shows a spinner until the pdf get splits.
- It extracts the base file name of the input PDF and creates a new folder name using the current date and time.
- It iterates through each page in the PDF, extracting text within the specified text_coordinates_rect and cleaning it to contain only alphanumeric characters.
- For each page, it creates a new PDF document and inserts the content of the current page.
- It creates a directory within the specified output directory with the folder name.
- It saves the new_doc as a PDF file within the created directory using the cleaned text as the filename.
- Then terminates the spinner and displays a success message after processing all pages.
- After Closing the success message it then opens the output directory in the file explorer.



```python
class ProgressDialog(QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.init_ui()

    def init_ui(self):
        # Logo
        pixmap = QPixmap()
        pixmap.loadFromData(QByteArray.fromBase64(base64_image))
        self.setWindowIcon(QIcon(pixmap))

        self.setFixedSize(50, 50)
        self.movie_label = QLabel(self)
        self.movie = QMovie("Sharp edges.gif")
        self.movie.setScaledSize(self.size())
        self.movie_label.setMovie(self.movie)
        self.movie_label.setGeometry(0, 0, 50, 50)
        self.movie.start()
```

The code defines a custom ProgressDialog dialog box with a fixed size (50x50 pixels), a logo, and an animated GIF ("Sharp edges.gif") to show visual progress feedback. When created, it displays the animated GIF in the dialog, making it suitable for indicating ongoing processes