

<sup>1</sup>Dorian NEDELICU, <sup>2</sup>Tihomir LATINOVIC, <sup>3</sup>Ljilja SIKMAN, <sup>4</sup>Mladen TODIC, <sup>5</sup>Aleksandar MAJSTOROVIC

# USING STREAMLIT AND BASIC4ANDROID (B4A) TO CREATE THE SAME APPLICATION – STREAMLIT VERSION

<sup>1</sup>Babes–Bolyai University Cluj–Napoca, Faculty of Engineering, Resita, ROMANIA.

<sup>2</sup>Faculty of Information Technology, University of Vitez, Vitez, BOSNIA & HERZEGOVINA.

<sup>3</sup>University of Banja Luka, Tehnological Faculty, BOSNIA & HERZEGOVINA

<sup>4</sup>University of Banja Luka, Mechanical Faculty, BOSNIA & HERZEGOVINA

<sup>5</sup>University of Travnik, Faculty of Technical Science, BOSNIA & HERZEGOVINA

**Abstract:** The paper describes “Unfold Sheets Parts” application designed to unfold the sheet metal parts (the unfolding of the following types of surfaces: cylinder intersected by two planes – 2 variants, intersection of two/ three cylinders of equal diameter and cylindrical elbow) designed with two programming languages: Streamlit and B4A. The Streamlit version of application use Python & Matplotlib to generate the unfolded geometry and to plot the numerical & graphical results; the application is publicly available on the Internet, does not contain any viruses and not store data to any external server. The B4A version of application is available to download for smartphones as “UnfoldSheetsParts.apk” in the “Programming” section of reference; this application does not share data over the internet, the code runs entirely on the user's smartphone, does not contain any viruses and not store data to any external server. Specific elements of the two languages are presented in comparison, as well as the conceptual differences resulting from their use in the creation of the application.

**Keywords:** sheet metal parts, cylinder intersected by two planes, Streamlit version, numerical & graphical results

## 1. INTRODUCTION

Streamlit is an open–source Python framework developed about 5 years ago to create dynamic data apps with only a few lines of code. Streamlit shortens the development time for the creation of web apps, allowing creating the app prototype in Python in hours instead of days. The “Part I” of the paper describe the Streamlit version of the “Unfold Sheets Parts” application, while the B4A app version will be presented in “Part II”. The required tools to write and edit the code, the command to launch the application in local browser and the option to deploy freely the app on Streamlit Community Cloud in just one click will be presented. Due to the page limit of the paper the code will be detailed explained only for the “One cylinder intersected with 2 planes – Version 1” geometry.

## 2. THE REQUIRED SOFTWARE PACKAGES

To develop a Streamlit app the following software packages are required:

- Visual Studio Code (VSC) – is used to write the code lines of the app and can be downloaded for free from the [5] reference;
- Python Extension for VSC – a high–level programming language. Python is free to use, even for commercial products, because of its OSI–approved open source license. Python is an interpreted, interactive, object–oriented programming language. To install Python language from VSC environment activate Extension option with Ctrl+Shift+X keyboard and select the Python module; also the Pylance and Python Debugger modules will be automatically installed to provide Python language support and to provide a debug experience with debugpy.

To develop a Streamlit app in VSC a folder must be created to hold the project structure like is presented in Figure 1, where:

- `__pycache__` – a folder created by the Python interpreter when it imports a module; it contains the compiled bytecode of the module, which can be used to speed up subsequent imports of the same module;
- `.streamlit` – contain the “`conFiguretoml`” file, a configuration file used to define the primary color for interactive elements like radio, button borders, etc., the background color for the main body of the app, the background color for sidebar and most interactive widgets and also the color used for the text [6];
- `Images` – a folder to hold the images used by the app; in our case this folder contain 11 image files;
- `pages` – contain the pages of the app which correspond to the Sidebar option; in our case this folder contain 6 files with Python code, Figure 2; the link between Sidebar options and the Python file to be

launched is made automatically by Streamlit if the option name in Sidebar is identically with the file name without “\_” character and also without “.py” extension, Figure 3;

- Video – contain the video files used by the app; in our case this folder contain 5 “mp4” video files, Figure 4;
- Main.py – the main app Python module which launch the app; to launch the app in a local browser a terminal window must be open in VSC selecting Terminal option menu follow by New terminal menu option selection; then write the following command in the terminal window: streamlit run Main.py; to deploy the app freely on Streamlit Community Cloud the steps from the reference [7] must be followed; also, it is possible to deploy the app on various cloud paid platforms and services [8].
- requirements.txt – a file that contains a list of additional modules needed to be loaded on the project, Figure 5.

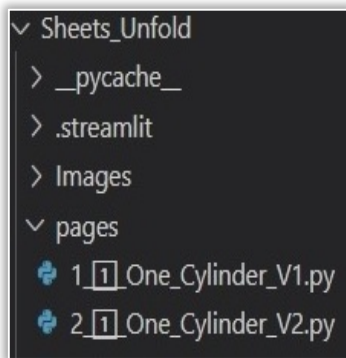


Figure 1. The “Unfold Sheets Parts” app structure

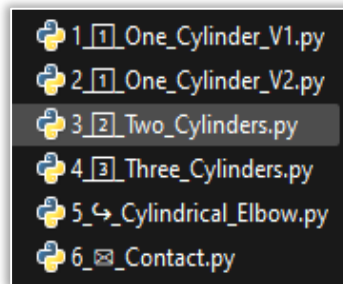


Figure 2. The “pages” Python files

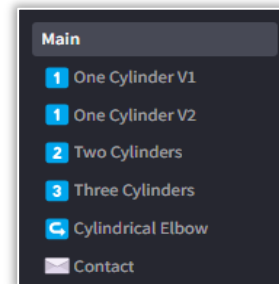


Figure 3. The option names in Sidebar

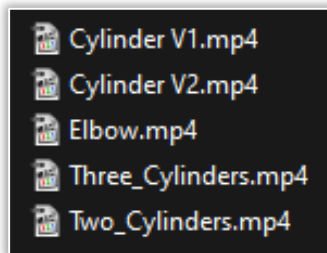


Figure 4. The content of Video folder

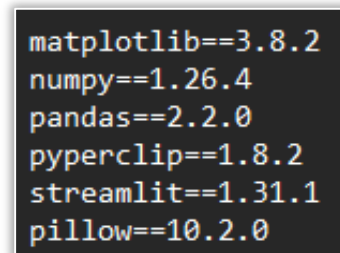


Figure 5. The “requirements.txt” content

### 3. THE MAIN MODULE

Figure 6 shows the main window of the Streamlit app:

- the Sidebar – contains selectable options to access the geometry for which the unfolding calculation is desired; also the “X” button hide the Sidebar to enlarge the Application screen area; the “>” button show again the Sidebar;
- the Application screen – contain the application title, some short information about the application and images of the geometry.

The Application screen is created by Main.py file code which contains 47 lines. Lines 1 to 3 import the required modules, line 5... 8 define the app title and the icon visible in browser tab and the Sidebar state “expanded” to see the options. Line 10...20 define the color of Streamlit main elements. By default, any HTML tags found in strings will be escaped and therefore treated as pure text; this behavior may be turned off by setting this argument to True in line 21, Figure 7.

The lines 22 to 47 of Main.py file is presented in Figure 8. Line 23 and 24 define the current folder and the folder of the image files respectively. Lines 26 define, with header command, the title “Unfold Sheets Parts” marked by blue square and arrow icons. Line 27...28 create a text with green color using markdown command. Line 29...30 create a new text with green color using write command. Line 31 ...33 create a new text with green color using link\_button command; also create a link to load the address <https://dorian-nedelcu.streamlit.app/> in the browser, where the same app in B4A version can be downloaded for smartphone. With divider command lines 35 and 47 display a horizontal line. Lines 36 and 43 insert a container element into the app that can be used to hold a single element, through empty command. Lines 37 and 44 create columns with 2 and 3 elements inside the previous created containers through columns command. Lines 38 and 41 insert images into column 1 and 2, loaded from Images

folder and also define the width and the title of those images. Line 45 insert one image into middle column no. 2, loaded from Images folder and also define the width and the title of the image.



Figure 6. The main window of the Streamlit app

```

Sheets_Unfold > Main.py > ...
1  import streamlit as st
2  from PIL import Image
3  from pathlib import Path
4
5  st.set_page_config(
6      page_title="Unfold Sheets Parts",
7      page_icon="🔵", # layout="wide",
8      initial_sidebar_state="expanded" )
9
10 dark = '''
11 <style>
12     .stApp {
13         primaryColor="Red"
14         backgroundColor="Black"
15         secondaryBackgroundColor="Black"
16         textColor="Gray"
17         font="sans serif"
18     }
19 </style>
20 '''
21 st.markdown(dark, unsafe_allow_html=True)
    
```

Figure 7. The lines 1 to 21 of Main.py file

```

Sheets_Unfold > Main.py > ...
22
23 current_dir = Path(__file__).parent if "__file__" in locals() else Path.cwd()
24 crt_dir_img = str(current_dir)+"/Images/"
25
26 st.header(":large_blue_square: Unfold Sheets Parts :arrow_down:")
27 st.markdown (""":green[This application has been developed in
28     Streamlit & Python & Matplotlib to unfold the sheet metal parts.]""")
29 st.write(""":red[The application is accesible through Internet,
30     does not contain any viruses and not store data to any external server.]""")
31 st.link_button(""":green[The same application created in B4A (Basic4Application)
32     is available as downloading for smartphones ('Unfold Sheets Parts.apk' file)
33     at the author page, section 'Programming'.]""", "https://dorian-nedelcu.streamlit.app/")
34
35 st.divider()
36 Place_Images = st.empty()
37 col1, col2 = Place_Images.columns(2)
38 col1.image(Image.open(crt_dir_img+"Cylinder_V1.jpg"), width=250, caption='One Cylinder - Version 1')
39 col2.image(Image.open(crt_dir_img+"Cylinder_V2.jpg"), width=200, caption='One Cylinder - Version 2')
40 col1.image(Image.open(crt_dir_img+"Two_Cylinders.jpg"), width=250, caption='Two Cylinders')
41 col2.image(Image.open(crt_dir_img+"Elbow.jpg"), width=225, caption='Cylindrical Elbow')
42
43 Place_Images = st.empty()
44 col1, col2, col3 = Place_Images.columns([0.01,0.8,0.1])
45 col2.image(Image.open(crt_dir_img+"Three_Cylinders.jpg"), width=550, caption='Three Cylinders')
46
47 st.divider()
    
```

Figure 8. The lines 22 to 47 of Main.py file

#### 4. THE "ONE CYLINDER INTERSECTED WITH 2 PLANES – VERSION 1" MODULE

Figure 9 show the window of the "One cylinder intersected with 2 planes – Version 1" option and Figure 10 show the numerical and graphical results generated through click on Calculate button.

This module is created by One\_Cylinder\_V1.py file code which contains 89 lines. The first 25 lines are presented in Figure 11. Lines 1 to 6 import the required modules. Line 8 to 10 defines the current folder and the folder of the image/video files respectively. Line 12 defines the module title with subheader command. Line 14 and 21 inserts a container element into the module that can be used to hold a single element, through empty command. Line 15 creates columns with 2 elements inside the previous

created container through columns command. Lines 16 to 19 insert images into column 1 and 2, loaded from Images folder and define the width and the title of those images. Line 22 creates columns with 3 elements inside the previous created container through columns command. Lines 23 to 25 open and insert video file into column 2, loaded from Video folder.

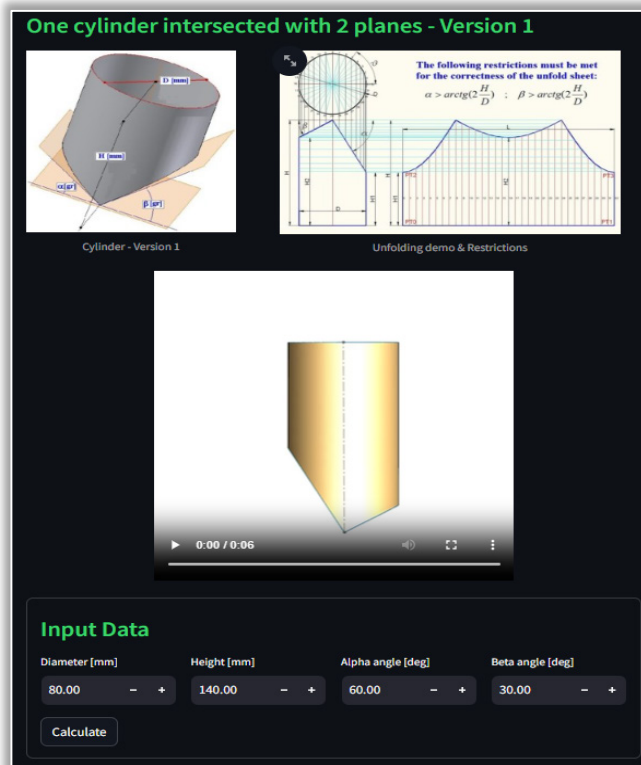


Figure 9. The window of the “One cylinder intersected with 2 planes – Version 1” geometry

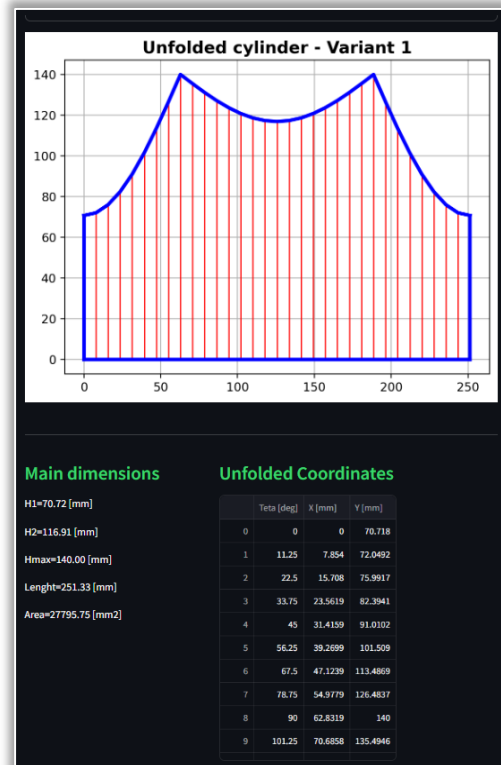


Figure 10. The results generated by this module

```

Sheets_Unfold > pages > 1_One_Cylinder_V1.py > ...
1 import streamlit as st
2 from PIL import Image
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import numpy as np
6 from pathlib import Path
7
8 current_dir = Path(__file__).parent if "__file__" in locals() else Path.cwd()
9 crt_dir_img= str(current_dir).rstrip("pages")+"/Images/"
10 crt_dir_vid= str(current_dir).rstrip("pages")+"/Video/"
11
12 st.subheader(":green[One cylinder intersected with 2 planes - Version 1]")
13
14 Place_Images = st.empty()
15 col1, col2 = Place_Images.columns([0.4,0.6])
16 col1.image(Image.open(crt_dir_img+"Cylinder_V1.jpg"),
17             width=240, caption='Cylinder - Version 1')
18 col2.image(Image.open(crt_dir_img+"Cylinder_V1_Demo.jpg"),
19             width=390, caption='Unfolding demo & Restrictions')
20
21 Place_Images = st.empty()
22 col1, col2, col3 = Place_Images.columns([0.2,0.6,0.2])
23 video_file = open(crt_dir_vid+'Cylinder V1.mp4', 'rb')
24 video_bytes = video_file.read()
25 col2.video(video_bytes, format="mp4", start_time=0)
    
```

Figure 11. The lines 1 to 25 of One\_Cylinder\_V1.py file

The 27 to 56 lines are presented in Figure 12. Line 27 creates a form where input data will be specified. Line 28 defines the form title with subheader command. Line 29 creates columns with 4 elements. Lines 30 to 33 define the fields of input data into the previous columns created, using number input command. The module requires 4 numerical data: Diameter, Height, Alfa & Beta angles and the number

input command offer the default values that can be changed by the user. Line 34 creates the button with the form\_submit\_button command and set the caption Calculate. Line 35 verifies if the button was clicked and only for True value run the lines 36 to 56 where unfold calculation is made. Line 36 calculates the value of constant  $\pi$ . Line 37 calculates the value of maximal angle  $U_{max}$ . Line 38 and 39 verify if any value of input data was unspecified in the field and if it is the case a warning message arise through warning command. Line 40 and 41 verify if Alfa & Beta angles respect the restriction formula: Alfa >  $U_{max}$  or Beta >  $U_{max}$ ; for True value a warning message arise through warning command; for True value the lines 43 to 46 are launched. Lines 43 to 47 calculate the following variables: NrPct – no. of points calculate for the unfolding geometry, the angles Alfa & Beta in radians, the heights H1 & H2, the length Lungime and area Aria of the unfolding geometry. Lines 49 to 56 calculate the unfold geometry numerical values: angle Teta, coordinates X & Y. Those values are memorised in array variables initialized in line 48.

```




Sheets_Unfold > pages > 1 [i] One_Cylinder_V1.py > ...
27 with st.form('Input_Data'):
28     st.subheader(':green[Input Data]')
29     col1, col2, col3, col4 = st.columns(4)
30     Diameter = col1.number_input('Diameter [mm]', value=80.0)
31     Height = col2.number_input('Height [mm]', value=140.0)
32     Alfa = col3.number_input('Alpha angle [deg]', value=60.0)
33     Beta = col4.number_input('Beta angle [deg]', value=30.0)
34     submit_button = st.form_submit_button('Calculate')
35     if submit_button:
36         PI=np.arctan(1)*4
37         Umax= np.arctan(2 * Height / Diameter) * 180 / PI
38         if Diameter==" " or Height==" " or Alfa==" " or Beta==" " :
39             st.warning("Please fill all fields with numerical data !")
40         elif (Alfa > Umax or Beta > Umax):
41             st.warning("Alpha and Beta angles cannot be greater than "+str(Umax)+" grade")
42         else: # st.success("Success !")
43             NrPct = 8 ; ALFAR=Alfa * PI / 180 ; BETAR=Beta * PI / 180
44             H1 = Height - Diameter * np.tan(ALFAR) / 2
45             H2 = Height - Diameter * np.tan(BETAR) / 2
46             Lungime = PI * Diameter # Lungime desfasurata
47             Aria = Diameter * (Height * PI - Diameter * (np.tan(ALFAR) + np.tan(BETAR)) / 2)
48             XP=[] ; YP=[] ; Teta=[]
49             for i in range(1, 4 * NrPct + 2):
50                 UT=(i - 1) * PI / 2 / NrPct
51                 Teta.append(UT*180/PI)
52                 XP.append(UT * Diameter / 2)
53                 if ((UT < PI / 2) or (UT > 3 * PI / 2)):
54                     YP.append(Height - Diameter / 2 * np.cos(UT) * np.tan(ALFAR))
55                 else:
56                     YP.append(Height + Diameter / 2 * np.cos(UT) * np.tan(BETAR))

```

Figure 12. The lines 27 to 56 of One\_Cylinder\_V1.py file

The 58 to 89 lines are presented in Figure 13. Line 59 inserts a container element into the module that can be used to hold the unfolded drawing, through empty command. Line 60 calculates the chart limits on X and Y coordinates. Line 61 creates the figure chart. Lines 62 and 63 define the plot title, the fontsize, the fontweight and the title color. Line 64 activates the grid drawing. Line 65 & 66 plot the red vertical lines inside the unfolded geometry, Figure 10. Line 67 plot the blue upper curve of the the unfolded geometry. Line 68 & 70 plot the two vertical & one horizontal blue lines of the unfolded geometry. Finally, line 71 places the chart in the Place\_Chart container.

Lines 73 to 89 lines show the chart main dimensions & coordinates. With divider command lines 74 and 89 display a horizontal line. Line 75 creates columns with 2 elements. Line 76 defines the title of column no. 1 with subheader command. Lines 77 to 81 display in column no. 1 the main dimensions: H1, H2, Height, Lungime & Aria. Lines 82 to 86 create data frame with the Teta, XP, YP arrays. Line 87 defines the title of column no. 2 with subheader command. Line 88 put the data frame numerical values into column no. 2.

When the mouse pointer is move inside the column no. 2 where coordinates are placed in tabular format, three icons appear under the column title:  to download the table in CSV format,  to search values in the table and  to maximize the table on the browser window.

```

Sheets_Unfold > pages > 1_One_Cylinder_V1.py > ...
58 # Chart drawing
59 Place_Chart = st.empty()
60 Xmax=XP[4 * NrPct] ; H1=YP[0]
61 fig=plt.figure()
62 plt.title("Unfolded cylinder - Variant 1", fontsize=14,
63         fontweight='bold',color='Black')
64 plt.grid(True)
65 for i in range(1, 4 * NrPct):
66     plt.plot([XP[i], XP[i]],[0, YP[i]], color='Red', linewidth=1)
67     plt.plot(XP,YP, color='Blue', linewidth=3)
68     plt.plot([0,0],[0,H1], color='Blue', linewidth=3)
69     plt.plot([0,Xmax],[0,0], color='Blue', linewidth=3)
70     plt.plot([Xmax,Xmax],[0,H1], color='Blue', linewidth=3)
71     Place_Chart.write(fig)
72
73 # Chart dimensions & coordinates
74 st.divider()
75 col1, col2 = st.columns([0.4,0.6])
76 col1.subheader(":green[Main dimensions]")
77 col1.write("H1="+'%0.2f' % H1+" [mm]")
78 col1.write("H2="+'%0.2f' % H2+" [mm]")
79 col1.write("Hmax="+'%0.2f' % Height+" [mm]")
80 col1.write("Lenght="+'%0.2f' % Lungime+" [mm]")
81 col1.write("Area="+'%0.2f' % Aria+" [mm2]")
82 df= pd.DataFrame(
83     {'Teta [deg]': Teta,
84      'X [mm]': XP,
85      'Y [mm]': YP
86     })
87 col2.subheader(":green[Unfolded Coordinates]")
88 col2.dataframe(df)
89 st.divider()

```

Figure 13. The lines 58 to 89 of One\_Cylinder\_V1.py file

## 5. REMARKS ABOUT STREAMLIT

Streamlit is an easy way to create web applications due to the language simplicity and various tools that can be used. The syntax of the commands is easy to understand even for beginners programmers. A complete documentation with various examples is available at reference [9]. Examples of projects created with Streamlit can be found in reference [10], with different topics: LLMs (artificial neural networks), data visualization, geography & society, sport & fun, science & technology, Natural Language Processing, Finance & Business and others. On YouTube can be found channels about Streamlit like the following references: Complete Streamlit Python Course – 46 videos [11], How to Build Your First Data Science Web App in Python – 53 videos [12], Streamlit Full Course 2023 – 13 videos [13], Introduction to Streamlit. What is Streamlit? Streamlit Tutorial – 9 videos [14], Product Announcements – 12 videos [15]. Also, a number of books with Streamlit topic can be found to learn the language: [16–18]. With Streamlit multiple tools can be used to plot charts: Matplotlib and Seaborn, Plotly, Bokeh, Altair and PyDeck, tools that can be used to create different type of charts and diagrams: Basic Charts, Line Chart, Scatter Chart, Bar Chart, Pie Chart, Histogram, Box Plot, Time-Series Charts, Geospatial Charts, Animated Bubble Map, Animated Bar Chart. In Streamlit connection with databases it is also possible, like: MySQL, SQLite, PostgreSQL, SQLAlchemy, MongoDB, Snowflake, BigQuery. For each database type, it is possible connection with the database, write SQL queries and then make example apps. Streamlit offers a rich set of input widgets to use in the project: Buttons, Download Button, Checkbox, Radio Button, Select Box, Multiselect Box, Slider, Select Slider, Text Input, Number Input, Text Area, Date Input, Time Input, File Uploader, Camera Input, Color Picker, Forms, Displaying Images, Displaying an Audio Player, Displaying Video, Sidebar, Columns, Expander, Container, Placeholder, Displaying Status and Progress, Spinner, Messages, Exception, Balloons & Snow. Streamlit is continuously updated by the startup team to offer new tools and improvements. Some advantages of Streamlit are: write beautiful & easy-to-read code, live editing because the app is updated instantly as you edit your script and is open-source & free. To deploy an app on Streamlit Community Cloud the following steps must be taken:

1. create a github repository with the code and with a requirements.txt file (where the app dependencies must be specified);
2. sign in <https://streamlit.io/>;
3. connect the github repo with your streamlit account;
4. deploy your app linking to the github repo where the code was stored;
5. in some minutes the app will be deployed; the app can be set public from settings; the app subdomain will be something like: [myapp.streamlit.app](https://myapp.streamlit.app).

## 6. CONCLUSIONS

The application was created in Streamlit & Python to exemplify the easy way to create a Web app even for beginners. Also, Python associated modules were used: Matplotlib, NumPy, Pandas Pyperclip and Pillow. The app complete code is available for download at reference [19] and can be tested at reference [3]. The part II of this paper will focus on programming the same app in Basic4Android language.

### References

- [1] \*\*\* Introduction to Streamlit How does work? <https://streamlit.io/>
- [2] \*\*\* Basic4Android (currently known as B4A) How to work?, <https://www.b4x.com/b4a.html>
- [3] \*\*\* Nedelcu D Unfoldeparts application <https://unfoldeparts.streamlit.app/>
- [4] \*\*\* Nedelcu D Programming section <https://dorian-nedelcu.streamlit.app/>
- [5] \*\*\* Visual Studio Code Free download and inst. <https://code.visualstudio.com/>
- [6] \*\*\* Streamlit How to configure api reference? <https://docs.streamlit.io/develop/api-reference/configuration/configuretoml>
- [7] \*\*\* Streamlit How to deploy your application? <https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app>
- [8] \*\*\* Streamlit How to apply sensitivity labels to your files and email? <https://docs.streamlit.io/deploy/tutorials>
- [9] \*\*\* Streamlit How to get started with Streamlit? <https://docs.streamlit.io/>
- [10] \*\*\* Streamlit How to working on App Gallery? <https://streamlit.io/gallery>
- [11] \*\*\* TinyURL This is a URL shortening web service <https://tinyurl.com/2jtqr9nd>
- [12] \*\*\* TinyURL This is a URL shortening web service <https://tinyurl.com/233qlue7>
- [13] \*\*\* TinyURL This is a URL shortening web service <https://tinyurl.com/2ynh7koc>
- [14] \*\*\* TinyURL This is a URL shortening web service <https://tinyurl.com/22ogng9e>
- [15] \*\*\* TinyURL This is a URL shortening web service <https://tinyurl.com/29whyzha>
- [16] Richards T 2023 Streamlit for Data Science, Packt Publishing
- [17] Richards T 2021 Getting Started with Streamlit for Data Science, Packt Publishing
- [18] Khorasani M, Abdou M and Hernández J 2022 Web Application Development with Streamlit, Packt Publishing
- [19] \*\*\* Nedelcu D Github application [https://github.com/DorianNedelcu/Unfold\\_Sheets\\_Parts](https://github.com/DorianNedelcu/Unfold_Sheets_Parts)

**Note:** This paper was presented at **International Conference on Applied Sciences – ICAS2024**, organized by University Politehnica Timisoara, Faculty of Engineering Hunedoara (ROMANIA) and University Vitez, Travnik (BOSNIA & HERZEGOVINA), from May 30 to June 1, 2024, in Travnik (BOSNIA & HERZEGOVINA).



ISSN 1584 – 2665 (printed version); ISSN 2601 – 2332 (online); ISSN-L 1584 – 2665

copyright © University POLITEHNICA Timisoara, Faculty of Engineering Hunedoara,  
5, Revolutiei, 331128, Hunedoara, ROMANIA

<http://annals.fih.upt.ro>