P-ASSISTANT (PYTHON)

A

Mini Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

P.S.K.PAVAN KALYAN

1602-19-733-050

And

V.SRIRAM

1602-19-733-052

Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2019

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

DECLARATION BY THE CANDIDATE

We, **P.S.K.PAVAN KALYAN** and **V.SRIRAM** bearing hall ticket numbers, **1602-19-733-050 and 1602-19-733-052**, hereby declare that the project report entitled **"P-ASSISTANT (PYTHON)",** Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering**.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

P.S.K .PAVAN KALYAN (1602-19-733-050),

V.SRIRAM (1602-19-733-052).

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"P-ASSISTANT (PYTHON)"** being submitted by **P.S.K.PAVAN KALYAN** and **V.SRIRAM,** bearing, **1602-19-733-050 and 1602-19-733-052** ,in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

Dr. T. Adilakshmi,

Professor & HOD,

Dept. of CSE,

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to R.SATEESH KUMAR, our project guide, for his valuable guidance and support, along with his excellent teaching skills and persistent encouragement.

We are grateful to our Head of Department, Dr. T. Adilakshmi, for her steady support and the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal,

Dr. S.V. RAMANA, as well as the management of the institute, for having designed an excellent learning atmosphere.

ABSTRACT

The P-Assistant is a mobile application that serves as a real-time virtual assistant. It works on Android supported devices(windows too) and provides the users with features like stop watch, calculator ,notes and more. It is useful as a multi-purpose virtual assistant providing all the features in one single application. We used Python for its development and Kivy framework for the application design process along with other python libraries such as Os , Wikipedia ,Clock , KivyMd etc.

The application code is written purely in python and the application packaging is done using an external python library buildozer, which converts kivy application program into an apk file.

For the apk packaging, we used the Google colab resource provided by Google, for virtual Linux environment and installing the required dependencies.

The application comes with :

1.Notes maker

2.Notes viewer and editor

3.Calculator

4.Simple Wikipedia

5.Stopwatch and

6.A mini game (15-block puzzle)

The user can make and save simple notes, which will be stored in the internal memory of the mobile (android) device (handled by OS module). The user can also access a simple calculator and stopwatch and a mini game of 15 block puzzle. The application uses the Wikipedia library of python for some Wikipedia queries. It can always be extended by adding additional features.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The main aim of the project, P-Assistant is to provide android users with a multi purpose application that serves as a virtual assistant in daily life things. This mini project is the lite version of the P-Assistant that comes with lesser features. It provides the users with all the features in a single application ,which can work together and become more useful than regular applications for each of the features which are provided in regular android devices.

This lite version of P-Assistant is an android apk file that can be installed on any android supported device and uses only the internal memory of the device for saving the notes created by the user and some memory for apk, all the other features working using the required python modules or libraries packaged along with the apk. It covers some main needs in the day to day life of common people, most importantly students. The most useful feature is the note maker and editor that saves all the written data or information in a text(.txt) file which can be later viewed, edited or deleted. The next important feature is a stopwatch which uses the python clock library and kivy widgets for displaying the time with precision upto milliseconds with very small error(which occurs due to delay in python language itself). The working of the stopwatch is manipulated accordingly to reduce the error percentage.

The working of the stopwatch is manipulated accordingly to reduce the error to a minimum negligible percentage with required manipulation in the code to speed up the counting process so as to match with the real time with almost maximum possible accuracy. Then there is a simple wikipedia query answer feature that works using the wikipedia library of python for queries that are available in the Wikipedia, which is an open source platform containing tons of information from all over the world. The Wikipedia can be compared to a multilingual encyclopedia having information related to most of the things around us and the wikipedia module uses web scraping to collect information from the main website and displays for the user in one page of the same application, which is the main aim of the application. The other features include a calculator that supports complex mathematical functions including the irrational value “pi”, the logarithmic values (log(any_number , base_number) and log(any_number) which calculates the

logarithmic value to base “e” and including the special value “e” itself) and the square roots of the positive numbers along with all regular arithmetic operations of numbers (including x raised to the power y). The calculator in the lite version is limited to arithmetic and exponential mathematical functions whereas the main version includes the trigonometric, inverse-trigonometric and complex mathematics operations too. The application also has a game, the 15-block mini puzzle for entertainment which does not store any data and is completely dynamic, so that the users need not worry about the application taking up more space or resources.

The scope of this project is to help the users in their daily life tasks with these features, e.g., The notes feature can be used to write and save notes or the tasks needed to perform, the stopwatch can be used in multiple parts of daily life where the task at hand needs to be done in a specified time like exercises, the calculator is also very useful especially for students when they forget to carry their calculator and the game is simply for being busy with something when the users are bored, and the puzzle style of the game is also a benefit.

The whole project is written in python using the Kivy Framework, an open source framework for android application development using python, and some external python libraries/modules like Wikipedia. All these required additional dependencies along with the main file are packaged into the apk file using Buildozer, another open source tool that automates the entire build process of the apk using the python-for-android project tools and provides a buildozer.spec file that needs to be modified for personalization of the application.

The building process is done in google colab, a free product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs. Buildozer is primarily Linux and OSX based tool which requires a Linux environment for packaging source code into an apk file for android and requires an OSX environment for building an iOS application from the given Kivy-python source code of the application. The google colab works as

a very useful virtual Linux environment tool so that anyone can access the services of buildozer from any other OS based system, like Windows.

The application has a total size of 15MB and should be directly installed, i.e, it is not yet available on google playstore or any other app store, as it is only the lite version. Upon installing, the user has to allow the required permission to access internal storage of the application and internet access (if needed to use the Wikipedia feature). Then the application is ready to run and displays 6 buttons on the main screen upon starting.

The apk building agrees to all the terms and licenses during the build phase and uses or accesses the internet only for web scraping from Wikipedia website, only when the user requests a query, so the users need not worry about their privacy or other security issues. The application is completely safe and does not track the usage or any other specific data of the users for any reasons.

System Analysis/ Overview of proposed system

The proposed system of the application uses simple widgets provided by Kivy framework such as buttons, text input widgets, labels and a basic Float Layout, which acts as the container of all the other components. Most of the buttons are given specific call-back functions in the source code which are binded with the button widgets and get called everytime the specific button is pressed by the user, which is detected by the event manager of Kivy. The additional components used from the Kivy module include the ‘App’ widget, superclass of all applications developed using Kivy, that specifies the application class and is important in the building of the .apk file. The ‘Clock’ object provided by Kivy is also used, especially for the stopwatch scheduling.

The python libraries used are os, math and Wikipedia, each having their uses in different features such as notes saving, editing and deleting by os module, math.sqrt(), math.exp() and other functions from the python math module in the calculator working and Wikipedia module for simple web scraping from the Wikipedia main website using in-built functions for Wikipedia queries.

There are also some global variables, which have default values each time the application is opened newly but can be changed during the running of the application according to the user convenience. One such variable is the 'folder' variable, which contains a string, the path where the files created by the application are stored and retrieved from. The 'folder' variable can only be changed by the developer and not the user so as to ensure that all the files are safe and reachable and these notes/text files created in this specific app do not get mixed with the pre-existing or other text files from other sources in the internal storage of the device memory. This also helps in faster retrieval of the notes saved in this application compared to other file management applications that take up long time searching all through the memory for the required file.

The system workflow is very simple involving only buttons, text fields and respective call-back functions and a single layout to reduce the weight/usage of resources by the app. Kivy framework provides different types of layouts which can be used as suitable by the developer and the user namely float layout, grid layout (all widgets are arranged row-wise and column-wise), page layout (each widget takes a page which can be swiped for accessing the next widget, more like the pages in a book) etc. The Float Layout is the most suitable for mobile application development and the one used in this application where each widget is assumed to be freely floating in a layout space and the sizes and the position can be changed as per the convenience of the UI developer. The float layout is also the main page of the application on which all other widgets are drawn (added and removed) during the running of the application.

The home page is the main page which opens on starting the app, and has 6 main buttons, each responsible for one of the features mentioned. The buttons are arranged in three rows and two columns design, with the button in the first row and first column responsible for the Wikipedia service. The second button is for

the note making and saving feature. If there are any prewritten notes, they can be edited using the same button/feature. The left button in the second row opens the game, with all the tiles in a random manner and the user can start playing. The fourth button opens the calculator with all required buttons and display widgets. The last row has one button to view/read the pre-written notes and delete them in the first column and the last button opens the stopwatch. All these six buttons have similar call-back functions, which perform mostly the following three steps:

1.Remove the buttons displayed on the home page from the layout

2.Add the widgets required for the service to the layout

3.Provide required service to the user

4.Upon completion, if the user presses “Back”/“Quit” button, remove the widgets of the specific service from the layout

5.Add all the buttons of the home page to the layout.

During any of these steps, if the user closes the application, the application stops running and all resources are released. The application also does not use any threads, and so there is also no risk of deadlock or thread mishandling in any part.

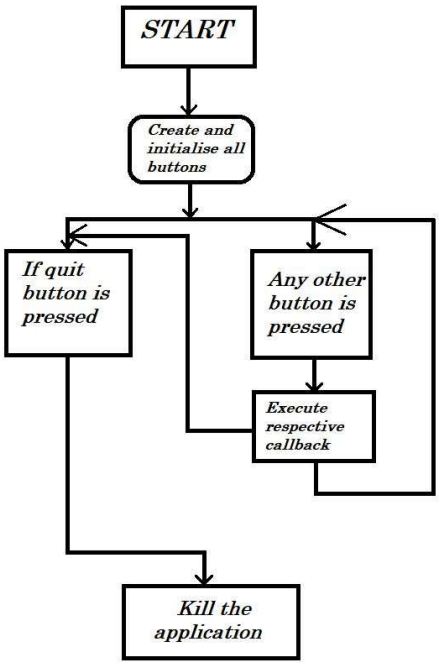

Fig.1.1 Basic work flow of application.

The application starts with the main function of the source code, which in turn makes an object of the App class, named as “DemoApp” and calls the run() method on the instance of the DemoApp class. The run() method calls the build() method of the class, which simply returns an instance of the layout designed for the application. The constructor function i.e., ‘_init_()’ function constructs the parent layout, that is the super parent class of Float Layout widget provided by Kivy. The second step, the program checks if or not there exists a folder for reading or saving the notes written by the user. It uses the ‘folder’ variable, mentioned in the introduction part, and the python os module for this purpose, checking if the folder exists and creates the folder if it does not exist. Then the constructor calls a function ‘ButtonMaker’, which creates all the required buttons for the application in all the features. After creating all the buttons, the buttons are added to specific lists based on where the buttons are used. Further during the execution, when a specific action has to be done, all the buttons in the list specific to only that action are added to the layout and others removed which is easier than adding each button individually every time. Then it calls the ‘BindButtons’ function defined in the Layout class which binds each button with its respective callback function. Upon returning from the above function, all the required buttons are created and binded with their respective call-backs and ready for events. Then the function “AddHomePage1” is added which adds the main buttons to be displayed on the home page to the layout and the application is ready to work.

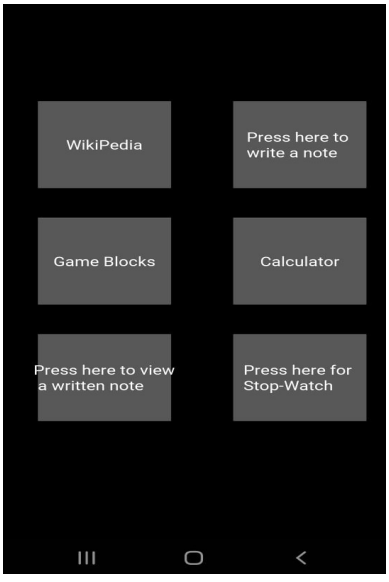

Fig.2.1 Display of homepage.

Each of the six main buttons on the main page i.e., the home page, are responsible for primary functions such as opening a particular feature page by removing earlier present widgets and adding the widgets required. The functions to the widgets are binded all at once in the beginning so the user does not experience any lag in the opening or working of any of the features as they are in a way always present but visible when the user needs them. This makes the app very static and dynamic with respect to the user thus using the benefits of both methods in a good way. Now let us see the main working of each of the buttons, i.e., how the features are programmed.

Wikipedia

The Wikipedia feature is the most extraordinary feature in the app that uses the Wikipedia module of python for simple web scraping (selective web scraping i.e., the module only tries to scrape data from the main Wikipedia website for only the given query) for a given no. of sentences.

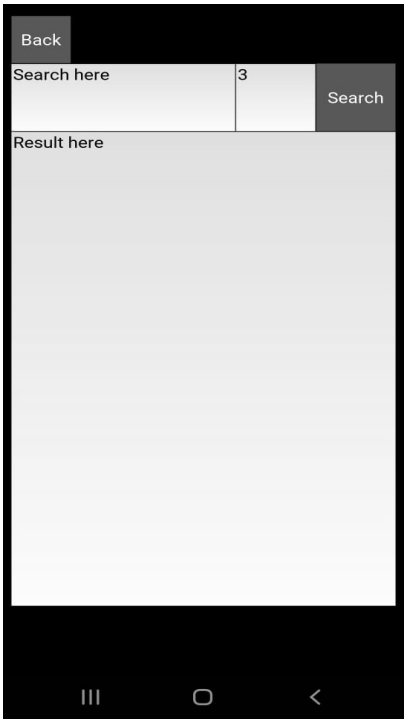

Fig.2.2 Wikipedia main layout

The default number of sentences scraped is ‘6’, which the user can change anytime. The module accesses internet and searches Wikipedia main page relative

to the query and if the particular required page is not found, throws back an error. If the connection isn't properly established it throws another error. However, all the errors are properly handled inside the program and proper information is displayed to the user in an understandable way using abstraction. The user has to respond according to the message displayed in case of error, however otherwise the information regarding the query is displayed without any error. If the no. of sentences is given as '0' or less (i.e., non-positive numbers), then the default number of lines are scraped and displayed, similarly in case the user inputs a large number as the number of lines to be scraped, then the number is changed to default value. If the number box is left empty, then '6' lines are scraped and the content of the box is also changed. In case the number box is filled with a non-numeric character, the result box displays "improper number of pages input", suggesting the user to change the number of sentences input.

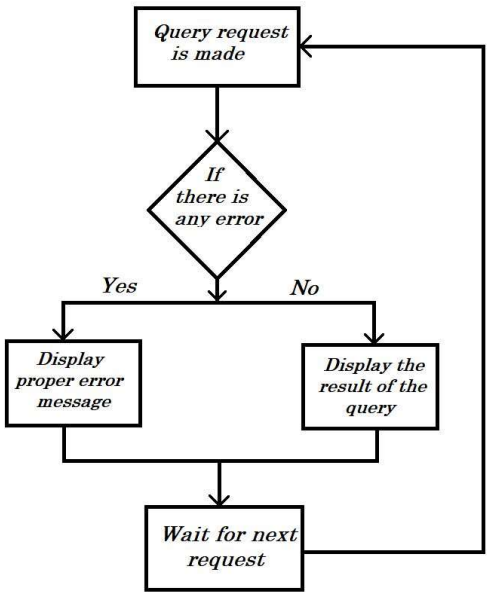

Fig.1.2 Flow chart of wikipedia algorithm

Error Handling : The main errors that can happen in the Wikipedia feature are :

1. Failure of internet connection
2. Improper Query input
3. Improper input of number of pages
4. Wikipedia-Module not found

These errors are handled without any problem for the user and ensuring that the app doesn't crash. Each of these error messages are displayed as shown in the following figures.

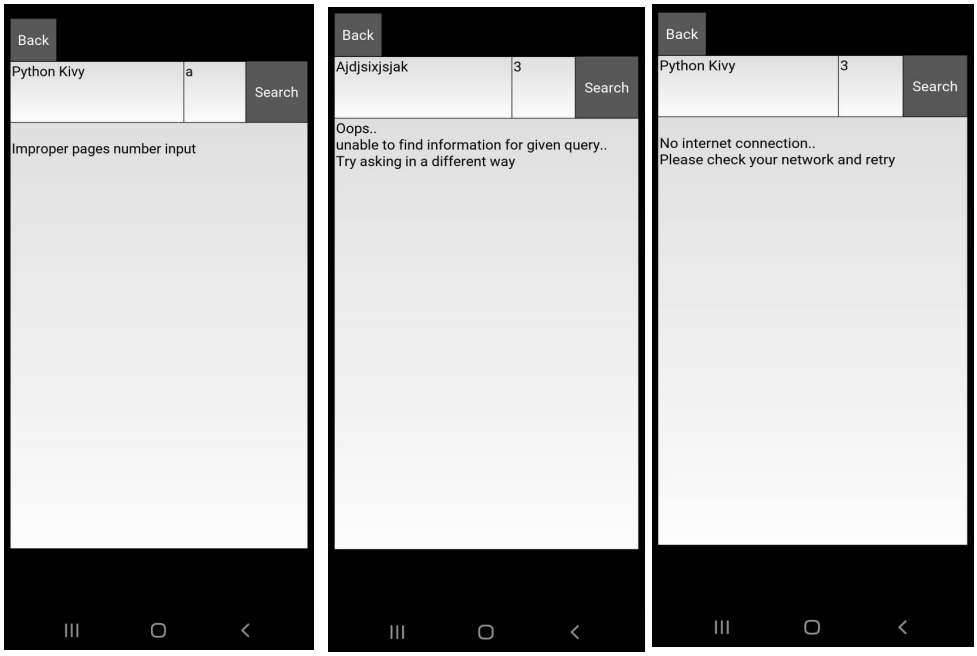

Fig.2.3,2.4,2.5 different possible errors in Wikipedia.

In case of no error the output is shown as below:

Fig.2.6 Output of Wikipedia query fetch

Notes maker and reader

The next best feature of the application is the notes saver. It allows the user to save any data in the form of text file in the external storage which can be accessed out of the app too (for sharing and other access purposes). The notes maker is given a separate button, which adds to the page three main widgets, one text field to take the name of the file as input and another text field to type the information. If the user wants to edit an already existing file, the user has to input the file name before writing anything in the data text field and press the button “Save”, which makes the program search for a file with the given name in the memory and if such a file exists, displays the content which was already in the existing file, to which the user can append new data or edit the existing data. If there is no such file, a new blank file is created.

It has to be noted that, if this process is not followed and the user types in information that has to be appended, and then gives the name of the file, then presses the save button, then the file is re-written, i.e., the old data is lost and replaced by the new data. If the user only wants to create a new file with new information/data, he can simply enter the needed information and give the name of the file, then press the save button to save the file .

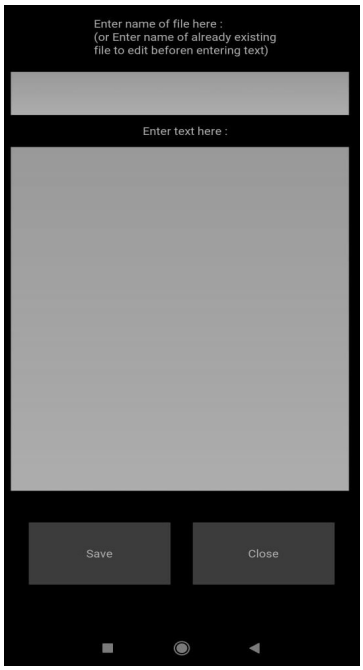

Fig.2.7 Notes maker main layout

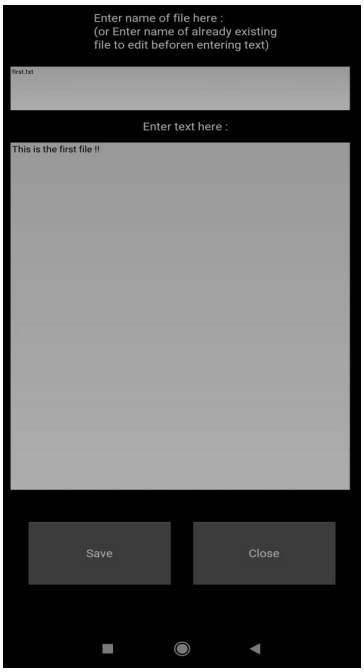

Fig.2.8 Creating and saving a file

It has to be noted here that if the user enters the name of an existing file for the new file, the last edited one is saved. There is one possibility of error in this feature, which is when the user doesn't enter any file name, which is handled by the app, i.e., whenever the user tries to save a file without any name, the app displays "Empty name error" at the lower part, which stays until the user enters a name and saves the file or quits from the application.

The notes viewer is simpler than the editor feature. It has four main widgets along with few other helper widgets. The first one is the name text field in which the user has to enter the name of the file to be shown. Similar to the editor feature, if the name field is left empty, the app displays "Empty name error" until the name of the file is entered. The next one is the display text field, in which the app displays the content of the file once the user enters the name of the file. Before the user enters any name of file, that is always when the feature is opened, the display text field shows the names of all the text files created and saved using the application which are not yet deleted. The user is prompted to choose a file name from the list displayed and this prompting continues until the user enters a valid name of file, i.e., a file name only from the displayed list. Then the user has to press the "Submit" button, which prompts the application to fetch the file from the memory and display the data in it in the display text field. The data is set to 'read-only' mode, which means that if the user wants to edit anything, the user has to open the edit and save feature. Although, this feature provides the "delete" option, below the submit button which immediately deletes the file which is opened at that time. Then the user is again prompted to enter name of any file, from the list of existing files. The user can view another file or leave depending on the need. The user should be careful with the 'delete' option provided, as the application doesn't ask for any confirmation from the user once the button is pressed. The file is completely permanently deleted, i.e., it is also removed from the external memory of the device.

One more important point to be noticed is that in every feature provided, there is a button provided to return to the main page which has to be used to return to the main page and not the regular android "back" button provided at the bottom of most of the touch-screen android devices. The "Quit" button discussed in the flowchart and the introduction part is this regular android "back" button. The

reason behind this is that this app works in a way that combines static and dynamic approaches, i.e., there is always only one single layout onto which different buttons are added depending on the feature needed by the user. So the app is always in the same layout and so the back button makes the app to return back to the android main screen leaving the application. So to return to the homepage, there is a button provided along with the other widgets needed in every feature. In fact, this “quit/back” button does not really take the application back or quits the feature. It simply removes the widgets of the currently opened feature and adds the six main buttons of the main page, which makes the user feel as if the application quit the feature and returned back to main screen. If the android “back” button is used, at any point of the running of the application, the application is killed and the android device returns to main page of android screen.

The notes maker in the lite version comes with lesser features than the main app, such as – no confirmation before deleting the file, saving only text files, the visibility of text files in the internal storage of the application (less security). The main version of the application is supposed to combine the notes maker app with the alarm manager, such that when the user wants to add a check-list or to-do-list and the time at which the tasks have to be done and sends notifications accordingly at the set times. But the lite version does not have an alarm manager/ reminder feature and so this to-do-list feature is also removed from the lite version.

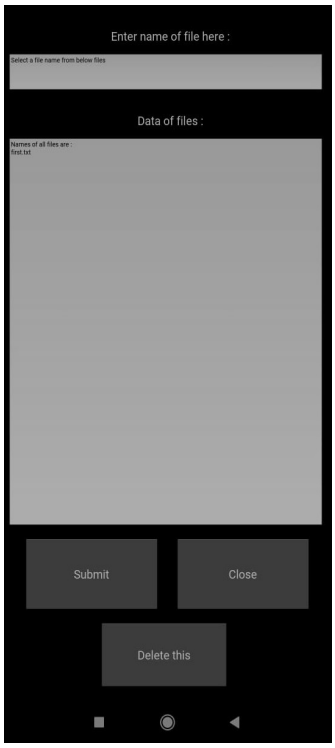

Fig.2.9 Notes Viewer main layout

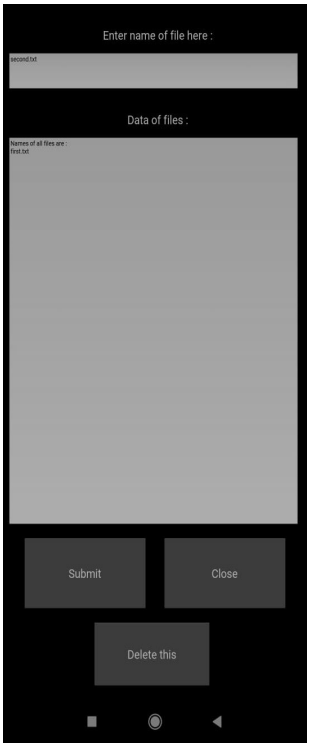

Fig.2.10 Giving input

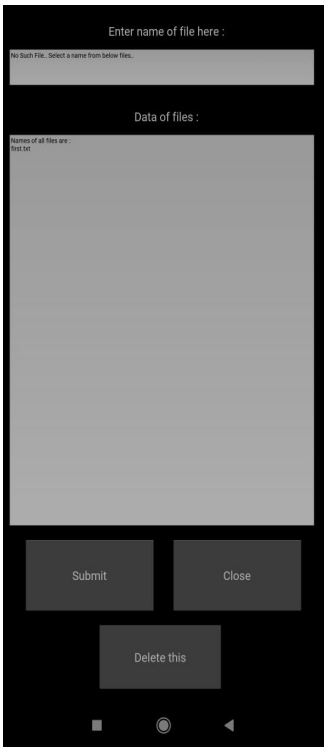

Fig.2.11 File not found error handled by the program

Game

The game in the application is the implementation of the infamous 15-block puzzle and the code written is completely original and not copied from any source, and customized accordingly with the kivy widgets and the needs of the android application with respect to size and movement.

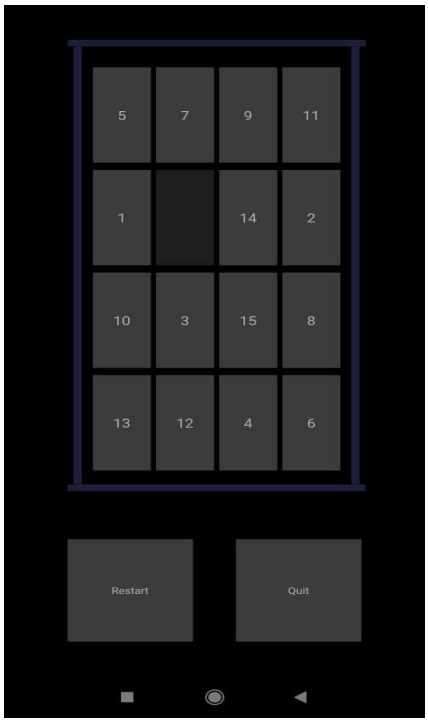

Fig.2.12 Main layout of game

The description of the game is given as :

The game consists of one main frame, in which 15 numbered tiles are present along with one empty space in a 4x4 matrix form, and in a shuffled way i.e., the numbered tiles do not have any particular order in the beginning. The objective of the game is to arrange the tiles in the maximum possible row-wise ascending order i.e., the first row must have the tiles in the order “1 2 3 4” and the next row “5 6 7 8” and so on. The empty space is automatically adjusted at the end of the frame, i.e., last row and last column. The phrase ‘maximum possible’ is to signify that the game puzzle cannot always be solved completely, and in such cases, the player must arrange the tiles in a way such that maximum number of tiles are present in the locations where they would have been present if the puzzle could be solved. The only constraint is that the tiles should only be slided/moved and not lifted and placed in some other location. So a particular tile can only be moved whenever there is an empty slot adjacent to it in any of the four directions (diagonal movement is not possible), then changing its location to the location where the empty slot was present earlier leaving an empty slot in its original location. Since the tiles are moved and not lifted, and only the tiles adjacent to the

empty space can be moved, only one tile can be moved at a time. When the maximum possible order is arranged following the constraints, the player wins and the puzzle can be played infinitely until the tiles are properly arranged. So this puzzle is a simple never-lose game.

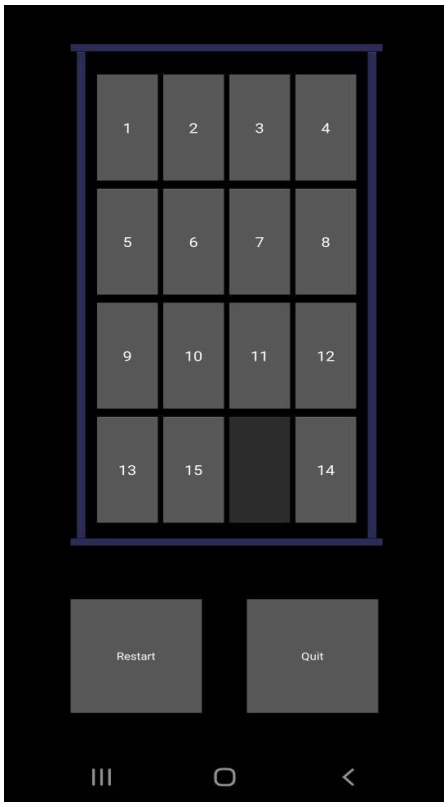

Fig.2.13 Bad solution case

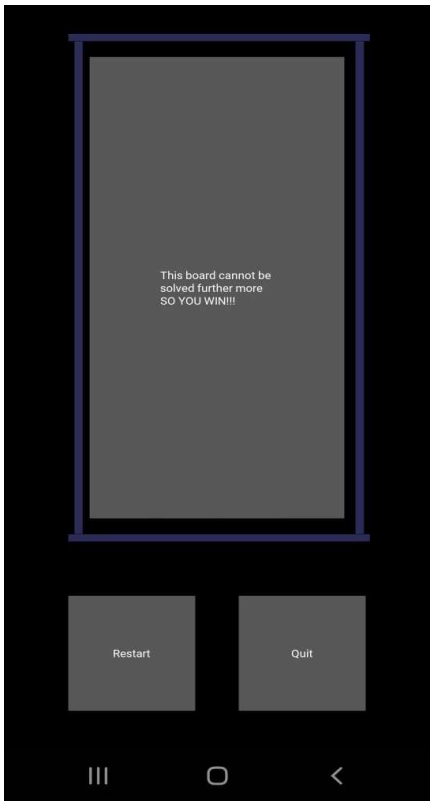

Fig.2.14 Display output of bad case

The puzzle game is implemented in the application using only simple kivy buttons. The frame does not exist to avoid heaviness of the application and all the buttons are simply added to the original layout itself. These buttons are numbered from 1 to 15 and have greyish-white color while the empty slot is also represented by a button and colored in dark grey. The buttons are binded to a single function which checks the constraints, i.e., if the tile can be moved and where it can be moved to, and then it exchanges the location, color and text of the buttons, (one of which is the empty slot and the other a numbered button) and removes them from the layout and adds them back to display the changes made. This looks to the user as if the buttons are sliding. All the numbered buttons along with the empty slot button are contained in a python list data type which makes it easier to add the buttons, check for constraints and also check if the puzzle is solved or not. The

user can move/slide the tiles represented by the buttons here and try to solve the puzzle as long as the user can and wants to. There is no restriction on the number of steps made or the time taken. The solved puzzle is checked based on particular conditions and using lists that contain the possible solutions to determine that the player solved the puzzle and has won. Since the buttons are maintained in a list, the order of the buttons in the list is compared to the order of the buttons in the solution list and if the order is same in both the lists, the application displays an appropriate message as a button, covering all the numbered buttons. On pressing this button, the game is restarted, i.e., the tiles are reset in random positions and the user can start moving i.e., playing again.

Apart from these, there are two buttons below, “Restart” and “Quit”. As mentioned earlier, the Quit button is used to return to the home page and leave the game. The game is completely dynamic, so the progress is not saved at any point during the game, i.e., if the user presses Quit button after solving a part of the puzzle, then on reopening , the old puzzle is replaced by a new random unsolved puzzle. There is also a refresh button, which refreshes the whole game, i.e., all the tiles are reshuffled (locations) and new random puzzle is generated. So, as the text suggests, the restart button completely restarts the game from the beginning. There are no chances of any error in the game. The execution part is divided into functions each having its own task. Finally, when the user arranges all the tiles according to the solution, the “checkWin” function is called which has to check if the player has won, i.e., all the tiles are properly arranged according to the solution (or partial solution) or not. To avoid high usage of resources, the function call to check if the puzzle is solved is done after each step i.e., one tile is moved, only when at least half of the puzzle is solved. The application checks the tiles at the end of the first half of the list and at the mid position of the first half of the list to check if the first half is solved. If both these tiles match with the tiles in the solution list at the same locations, the application assumes that the first half is solved and then checks for complete list solution. There is no end to the game, which means that the player can keep on playing the game, a new random puzzle is generated each time after the puzzle is solved.

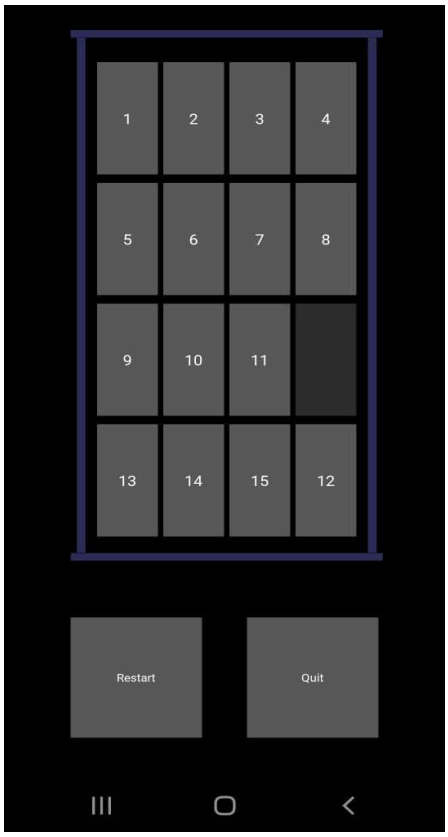

Fig.2.15 Good solution of game

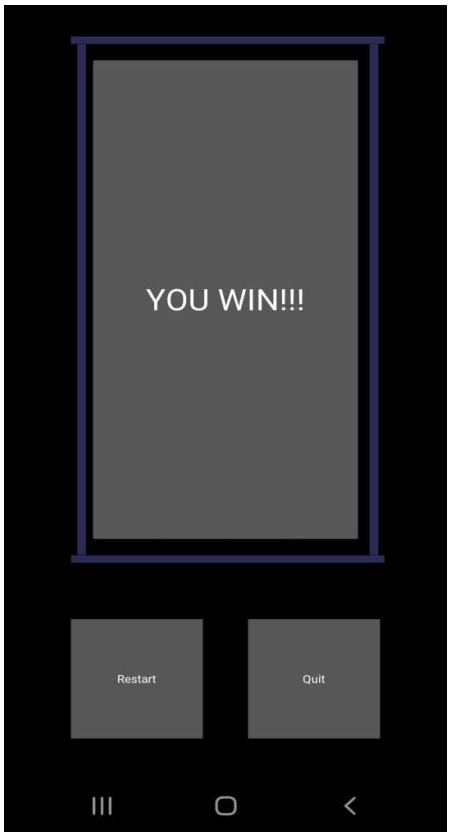

Fig.2.16 Display output of good case

Calculator

The application has a powerful calculator feature too, which uses a simple hack of python to work. It has so many buttons compared to the other features of the application. The total number of widgets in the calculator are ‘30’ (highest among all the features of this application) whereas the Wikipedia feature uses only ‘5’ widgets (least). There is one text display unit (read only possible type) in which the equation or the input and output is displayed.

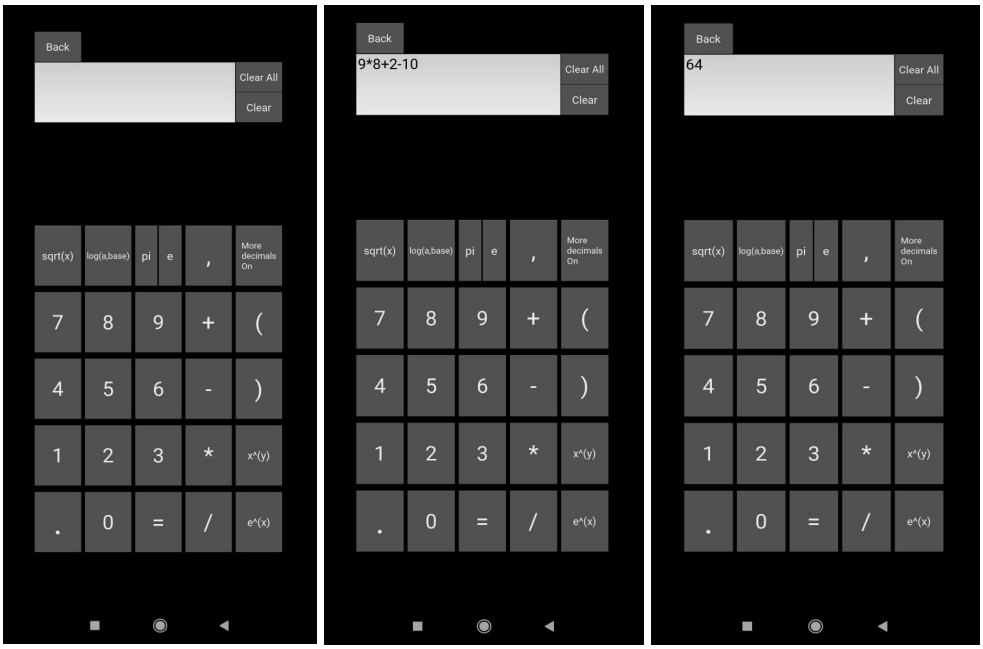

Fig.2.17 Main layout of calculator

Fig.2.18,2.19 Output of result of mathematical expression

The calculator has to solve the mathematical equation and display the result, so to avoid improper inputs from the user the text field is fixed as 'readonly' and any input has to be entered using the provided buttons below it. There are two buttons to clear the text field.

1. "Clear All" - clears the whole text field.
2. "Clear" - clears only the last character.

These two buttons are used to edit the content in the text field, in case any wrong input is given. The calculator is capable of solving all mathematical equations, with proper python syntax, but the lite version calculator is limited to only arithmetic, logarithmic and exponential functions. These are solved using the exec() function of python, which takes as input python code to be executed, as a string and if it is written with proper syntax, executed the code. This is very useful in the calculator feature as here also, the input mathematical equation is given as a sequence of individual characters. However, the output of the execution is by

default displayed in the console, so certain changes have to be made to the input string so as to display the result to the user using the UI elements/widgets. This process is hidden from the user and is done whenever the user presses the "=" button which indicates to execute the input given up to that point. If the input is syntactically correct, the exec() function executes successfully and the output is stored into a variable, because the string is manipulated as such. The output or the result of the execution which is stored in that variable, is then displayed in the same text field. In case the input is not syntactically correct, then the text field displays "Invalid Input – Syntax Error". To clear it again, the user can use the clear button mentioned above and to continue computation on the result, the user can simply input more characters/symbols and execute. The important point is that the execution is done only when button with '=' is pressed. The mathematical functions such as sqrt() which gives the positive square root of a non-negative number, exp() which gives the result of 'e' raised to the power of given number, the log(a, base) function which gives the logarithmic value of a number 'a' to the base 'b' and the mathematical constants symbols such as 'pi' and 'e' are imported from the python 'math' module.

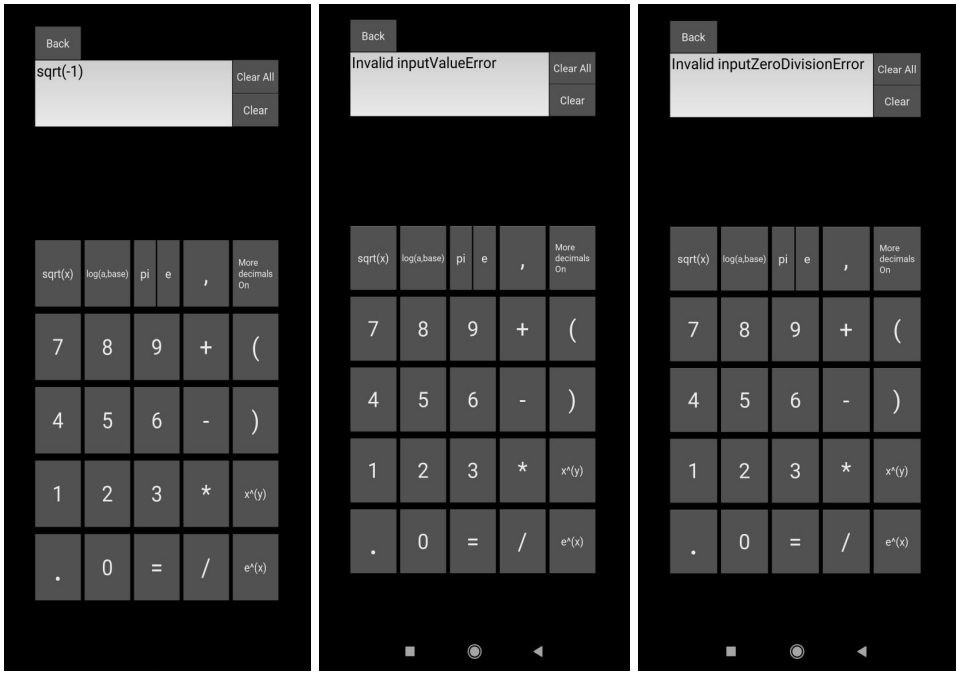

Fig.2.20,2.21,2.22 Various possible errors in calculator.

Stopwatch

The stopwatch is another useful feature of the application with simple design and widgets. The design is simple and adds only 6 widgets : 1 display field and 5 buttons. The display field is set to read only to make sure that the user cannot type anything into it. The stopwatch has maximum 1 day count i.e., it can count up to 23:59:59:99 after which it restarts the counter from 00:00:00:00. The stopwatch is simple to use and programmed in a very simple way.

The 6 widgets in the stopwatch feature comprise of one display field, which is a read only text field which displays the time count and there are 5 different buttons one of which is the “Close” button to return to the home page. The other four buttons are :

1.Start button : To start the counting of the stopwatch. This button is used to start from any state i.e., from pause state, or stop state or restart state. If the stopwatch is stopped or restarted or when opened without any event after opening, the start button starts counting the time elapsed from 00:00:00:00 and if the stopwatch count is paused, then on pressing the start button, the counting starts from the point of time where the stopwatch was paused. The start button is binded to a function ‘OnStart’ which starts the above mentioned functions of counting in different cases.

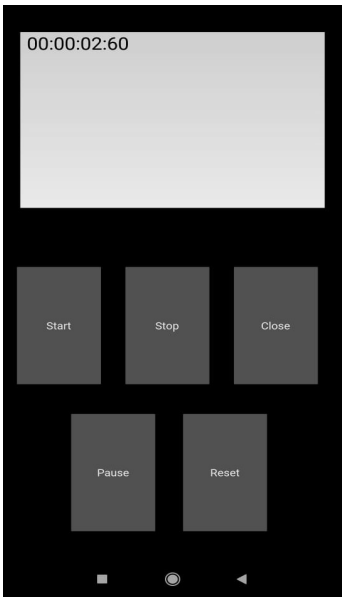

Fig.2.23 Execution of stop watch (start button)

2.Pause button : This button is used when the user has to pause the count in between and wishes to continue from the previously counted value later. In other words, this button stops the counting of the stopwatch temporarily which can be resumed later. This button is binded to a function ‘OnPause’ which changes the value of a variable used as a flag to indicate if the stopwatch is paused or not. The value of this flag variable is later checked in the ‘OnStart’ function to check if the stopwatch is paused or not.

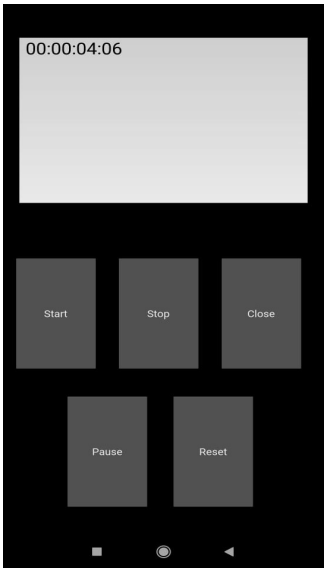

Fig.2.24 Execution of pause button.

3.Stop Button : This button is similar to the pause button but the user cannot resume from the point where the stop button was pressed. It means that whenever the stop button is pressed, the stopwatch dies there and the user has to start a new stopwatch for using it again. But to indicate the time elapsed until the point of stopping, after stopping the counting, the time still remains in the display unit. This button is bound to a function ‘OnStop’ which performs the mentioned actions and also changes the value of a flag variable which is used to indicate if stop button is pressed or not. It is important to note that the stop button has higher priority compared to the similar pause button. The greater priority makes the program to listen to the event of stop more than the event of pause. In simple terms, when the stopwatch is running, if the pause button is pressed, the stopwatch pauses temporarily and resumes whenever the start button is pressed again. But

before pressing the start button, if the user presses the stop button, the program considers that the stopwatch is stopped i.e., ended due to higher priority of Stop button. If the user presses the pause button after the stop button, it will be still treated as stopped but when the user presses pause button followed by pressing stop button, the program treats as if the stopwatch is stopped and starts from the starting value again i.e., from 00:00:00:00.

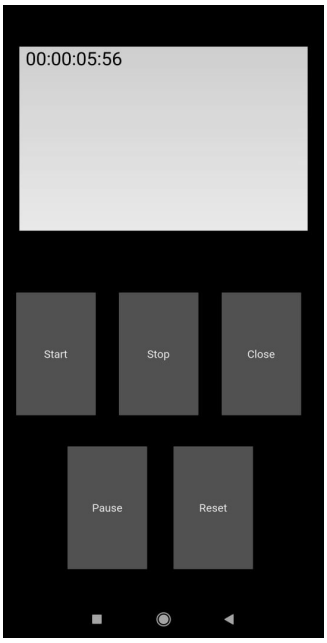

Fig.2.25 Execution of stop button

4.Reset Button : The restart button is used to restart the stopwatch. It ends the currently running stopwatch counter, initializes the stopwatch counter values to all zeroes again i.e., to 00:00:00:00 and then refreshes the display. It is different from the stop button in the feature that the stop button does not refresh the display. The stop button lets the time remain on the display unit but the refresh button clears everything and makes the stopwatch ready to start again i.e., restart as the name indicates.

These four buttons are used to control and use the stopwatch and the close button is used to leave the stopwatch and return to home page. During the execution of

the stopwatch, when the counter is running, if the user presses the start button, it starts again from 00:00:00:00. The important point is that these numbers denote the time elapsed in the format – " %HH : %MM : %SS : %mm" i.e., hours, minutes, seconds and millisecond multiple. This format is 24 hour format, so the maximum value of HH is 23, MM is 59, SS is 59 and mm is 99.\

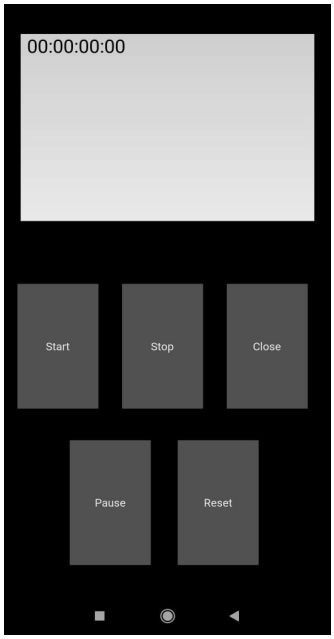

Fig.2.26 After pressing reset button

System design

The system design phase of the application is mostly dependent on Kivy framework only with few additional modules provided by python only for the Wikipedia feature. The Wikipedia feature requires a total module named 'wikipedia' to be packaged along with the application including extra requirements. These requirements and permissions are specified in the buildozer.spec file of the application during building. There are many lines in the spec file which determine the behavior of the application in android. The important ones include :

1. title = P-Assistant Lite
2. package.name = MiniProj
3. package.domain = PKVSorg.PKVS
4. source.dir = .
5. version = 1.0
6. requirements = python3, kivy, kivymd, wikipedia, sdl2_ttf==2.0.15, requests, chardet, idna, urllib3, certifi, soupsieve, beautifulsoup4
7. icon.filename = %(source.dir)s/data/icon.png
8. orientation = portrait
9. fullscreen = 0
10. android.permissions = INTERNET, READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE

These are the important lines of the buildozer spec file which have to be changed accordingly for the successful building of the application and running on android.

title = P-Assistant Lite

This line gives the title of the application. The title is displayed at the bottom of the icon in the android screen. The title can be any string, but it is better to give a unique and easy recognizable name for the user.

package.name= MiniProj

This line is used to declare the name of the package of the project. Since android is primarily Linux based, it uses packages (folders as viewed by user) to sort different apps and projects and files. Buildozer declares the package name of the application built as the name given (MiniProj here) and the apk file generated can be identified by the name of the package itself.

package.domain = PKVSorg.PKVS

This line is used to declare the domain of the package. This domain can be anything, i.e., if the user wants to add the application to an existing domain, the name of the domain can be given and if the user wants to make a new domain, a new name can be given which will be treated as the domain of the application being built. The domain name can have two parts : a main domain (general ones are ‘com’, ‘org’, etc.) followed by a subdomain (the domain specific to the application). Here PKVSorg is given as the main domain in which the subdomain is PKVS.

source.dir = .

This line is used to indicate the directory in which the source files and all documents related to the main source code of application are to be searched. It can be given a path when working outside of the source directory. It can also be given a simple ‘.’ character, which indicates to search for the source files in the current working directory. The ‘.’ defaults to the current working directory to search for the main.py file and other required files and documents. It is important to note that the source code file containing the main method of the application should only be named as ‘main.py’ and if the file is not named as ‘main.py’, buildozer throws an error that the main source file is not found.

version = 1.0

This line is used to indicate the minimum required version of android for the application working. This application takes the minimum required version as ‘1.0’ so that the application can work on any android device with android version equal to or above 1.0 . This line is particularly important for some applications which use external APIs or packages or modules with limited support on android versions, i.e., some APIs are not supported on some android version and require a minimum limit of android version. This application works only using python code for most of the features and so doesn’t need requirement of higher android versions.

requirements :

python3, kivy, kivymd, wikipedia, sdl2_ttf==2.0.15, requests, chardet, idna, urllib3, certifi, soupsieve, beautifulsoup4

These are the requirements of the application to build and work successfully. These requirements include external packages, dependencies and libraries required during the execution of the application. All the requirements have to be mentioned in this line in the buildozer.spec file for the successful running of application. Buildozer collects all the requirements mentioned in this line and packages those into the apk. If all the requirements are not mentioned, the apk file will still be generated but the app crashes on opening after installation. This is a serious issue as this error is not even displayed on the logs. Important point is that these requirements only mention the libraries and dependencies and APIs (if any) used and not android related or device related requirements. All these requirements such as wikipedia (the wikipedia module in python), requests, chardet, idna, urllib3, certifi, soupsieve and beautifulsoup4 are required for safer web scraping for wikipedia feature of the application.

icon.filename = %(source.dir)s/data/icon.png

This line is used to give a specific image for the application, which will be displayed as the icon on the android device after installation. We have to specify the complete path of the file we want to use (image file) as the icon of the application. We can comment or exclude this line from the buildozer.spec file which simply makes buildozer use the default kivy logo image for the icon of the developed application. Here the path '%(source.dir)s/data/icon.png' means that the icon to be used is in a folder named data existing in the current directory (source.dir= .) mentioned above and names as 'icon.png'.

orientation = portrait

This line indicates the possible orientations of the application in which the app can work. Here portrait is mentioned so that the application can work only in portrait mode even if the orientation is changed using auto-rotation feature of new android devices. This portrait orientation is fixed because of the positioning and UI design of the application source code. The other possible orientation values are 'landscape' which works only in landscape (horizontal) orientation and 'portrait | landscape' which works in both the orientation depending on the rotated orientation of the device.

fullscreen = 0

This line decides if the application should work in full screen mode or not. The default value when the buildozer.spec file is generated is ‘0’ which means the application will open and work in fullscreen mode.

android.permissions

INTERNET, READ_EXTERNAL_STORAGE,
WRITE_EXTERNAL_STORAGE

These are the android permissions to be granted to the application by the android device. The permissions have fixed names which have to be mentioned for the application to access the required content from the android device for running successfully. This application requires the above mentioned three permissions to work successfully. The details are –

INTERNET – this permission is required to access the internet connectivity of the device. The Wikipedia feature in the application searches the required query in the internet in the main website of wikipedia and scrapes the main content to be displayed and returns the scraped data.

READ_EXTERNAL_STORAGE – this permission is required to access and read the external storage of the device on which the application is installed. This along with the write permission to external storage (the permission given below) are the most dangerous permissions that can be given to android applications. These permissions allow the application to access the internal storage and manipulate all the data stored in the device including sensitive information regarding the device, the user usage data and even private information. But this application only uses one particular folder named ‘P-Assistant Files’ which will be created in the internal storage of the device automatically when the app starts running for the first time and also only the text files in the application will be read for the note viewer and editor feature of the application.

WRITE_EXTERNAL_STORAGE – this permission is similar to the above read permission but this allows the application to create and save new files in the

storage of the device. This permission is essential and important to use the make and save notes feature of the application. It allows the application the permission to create and save new files (text files as notes) and save them in the internal storage of the application to be viewed or edited or deleted later. The created files will be stored in the internal storage memory of the device until they are deleted, which can also be directly accessed by opening the folder directly in the device.

The system uses the python.os module for all the with the files i.e., creating, saving, retrieving and also deleting. The application doesn't go beyond its own folder and to other files ensuring security and privacy of the user. Also, the files created and saved by the user are only accessible in their device unless transferred voluntarily by any person to some other device. The files are also not uploaded to or sent via internet by the application for any reason.

Then the wikipedia module is installed and packaged with the apk and other dependencies like requests module which has specific internet requests, beautifulsoup4 module which filters and cleans the data scraped from the internet before displaying in the result field of the wikipedia module.

All the remaining UI design of the application is done using only Kivy widgets, and can be extended to using widgets from KivyMD package. KivyMD is an individual package apart from kivy, which contains collections of smarter widgets that can be used in applications designed using Kivy framework. These widgets have better graphics, themes and look better on the devices. KivyMD has to be packaged along with all the files and libraries included in the building of the application if the application uses the kivymd widgets.

Indirectly the kivy framework also uses a python module called 'pygame' that is packaged along with the kivy library for managing the display of the widgets and graphics in the GUI. Pygame is actually a python module that is primarily designed to make 2D games in a simple way using python programming language. But Kivy only uses pygame for the display graphics and android doesn't provide support for games written using pygame.

The application doesn't start unless the permissions are allowed to the application manually by the user. This is because, before running, the program tries to check if the folder required for saving and retrieving files exists in the internal memory

or not and if it does not exist, tries to create a new folder even before displaying the main screen or the home page. So, if the application is not provided permission to access internal storage and files, the application crashes on opening without even the home page. This application is also ad-free and doesn't track usage statistics or user choices or cookies.

PSEUDO ALGORITHM / IMPLEMENTATION

The best possible description of the approach/algorithmic style followed in the design process of this application will be the 'greedy approach' combined with the appropriate usage of the advantages in both static and dynamic methods. The application follows a greedy method because there is only one layout and one application that comes with several features and all of which are present in the same layout. The application is static in the sense that all the required widgets in the application are created once before the user starts operating i.e., the first steps of the program are to create and initialize the widgets which will be used during the running of the application. The application is dynamic because it is an interactive application where the actions of the user (events in the terminology of the application) determine the further actions of the app, like pressing a button will show the widgets of a specific application or perform a specific action assigned. The algorithm is simple, in each step, there is an actionlistener which keeps listening, i.e., checking if the user has performed any specific action, and if there exists a callback to the user action, perform the action and go back to the waiting stage to listen again. This is a very simple and straight forward approach and even the functions are written in a way to facilitate the easy usage of the callbacks and to make the functions bound to the buttons/widgets. As at the core of the intention behind this project, we only focus on singularity, i.e., making an application that can handle several features by itself without having multiple apps for each feature, even the code part is written completely in a single main.py file. All the functions and widgets belong to the class of a single layout that extends (is a subclass of) the main Layout class provided by Kivy framework. There is one class DemoApp which extends the main App class of Kivy. Each of the widgets are simply imported into the same file where all the work is done. The variables which are needed by multiple widgets and multiple functions are declared as global variables so as to avoid complexity of the code and all the widgets are declared as widgets of the layout class and not specific functions to ensure they are

easily accessed by different functions to perform required actions. This programming pattern is very different from the original app development programming patterns in other languages (like Java) where even each widget is treated as a separate object and so we declare classes as collections of widgets and their functions. This pattern also treats the widgets as objects but as objects of only one single class – the Layout class. This pattern achieves more readability and understandability and completely uses the flexibility of the programming language python. The code follows indentation, as a mandatory of python and all the widgets and functions are given identifiers such that anyone with basic knowledge of python programming language and good English speaking skills can decipher the idea behind the code and the use of the specific functions and widgets. As previously mentioned, this code is written in python (3) using Kivy framework, which is a cross-platform multitouch app development framework, so the code can be run on any Operating System (android, windows, macOS etc.) This code is developed on windows operating system using python compiler and packaged using Buildozer in a virtual Linux environment in google colab. This same code can be used to develop apps for macOS and windows too, which will be part of the future work of this project.

The simple algorithm of the application can be depicted by the following figure.

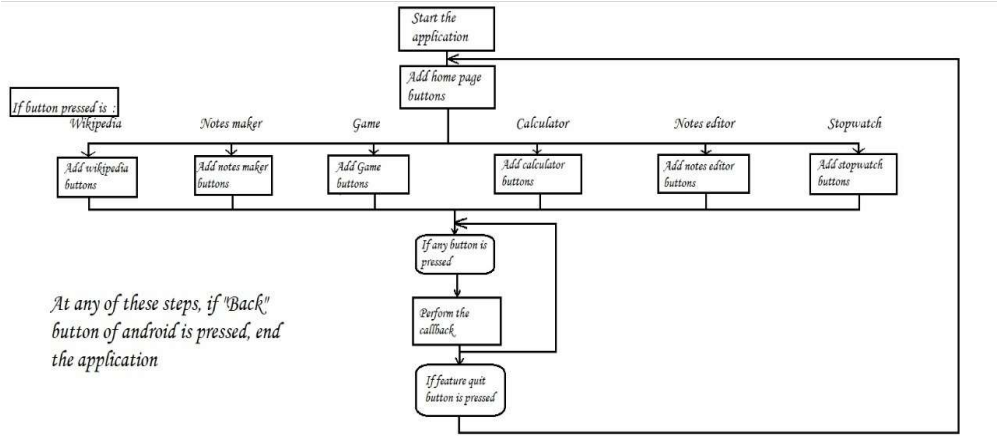

Fig.3 Flow chart representation of algorithm of the application.

All the modules used and widgets used for the app import statements :-

```
from kivy.app import App
from kivy.config import Config
from kivy.uix.button import Button
from kivy.uix.floatlayout import FloatLayout
import datetime
from kivy.uix.label import Label
from kivy.uix.textinput import TextInput
import os
from kivy.clock import Clock
from math import exp,sqrt,log,pi,e
```

Fig.4.1 Source code (Part-1)

Global variables declared which are used in the application while running:

```
folder='/storage/emulated/0/P-AssistantFiles'
OldNameOfFile=""
OldText=""
firsttime=1
pcheck=0
FileListString=""
badsol=["13","9","5","1","15","10","6","2","15","11","7","3"," ","12","8","4"]
solution=["13","9","5","1","14","10","6","2","14","11","7","3"," ","12","8","4"]
moredec=1
ans=0
```

Fig.4.2 Source code (Part-2)

Layout structure used in the application and all the functions used in the working of the application with function titles

```
class MyLayout(FloatLayout):
    def ButtonMaker(self):
    def BindButtons(self):
    def AddHomePage1(self):
    def RemHomePage1(self):

    def __init__(self,**kwargs):

    def StopWatchPress(self,button):
    def WikiBPress(self,button):
    def WriteBPress(self,button):
    def NoteBPress(self,button):

    def StopWatchClose(self,button):
    def Saveandwrite(self,button):
    def WriteBClose(self,button):
    def OpenFile(self,button):
    def NoteBDelete(self,button):
    def NoteBClose(self,button):
    def increment_STOPWATCHtime(self,interval):
    def OnPause(self,button):
    def OnReset(self,button):
    def OnStart(self,button):
    def OnStop(self,button):
    def Game(self,button):
    def AddButtons(self):
    def CheckWin(self):
    def GameShuffleButtons(self,button):
    def Move(self,Button):
    def QuitGame(self,button):
    def Calc(self,button):
    def UpdateBar(self,button):
    def WikiQuit(self,button):
    def WikiSearch(self,button):

    def UpdateDec(self,Button):
    def ClearCalcBar(self,button):
    def ClearCalcBarAll(self,button):
    def CalcQuit(self,button):
```

Fig.4.3 Source Code (Part-3)

The main class of App and the main function of the application:-

```
class DemoApp(App):
    def build(self):
        return MyLayout()

if __name__=="__main__":
    try:
        demo=DemoApp()
        demo.run()
    except Exception as e:
        try:
            f=open('/storage/emulated/0/PKErrorFile/Errorfile.text','w')
            f.write(e.__class__.__name__)
            f.write("\n")
            f.write(e)
            f.close()
        except:
            pass
```

Fig.4.4 Source code (Part-4)

RESULTS

App obtained after installation of the apk file in the mobile, Home page of P-assistant app.

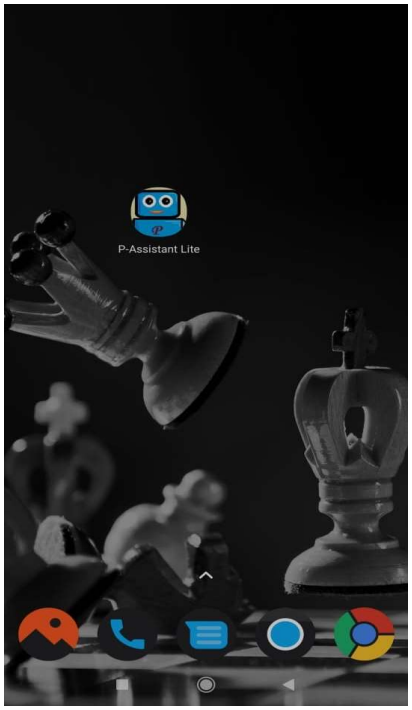

Fig.5.1 Icon of the app in mobile

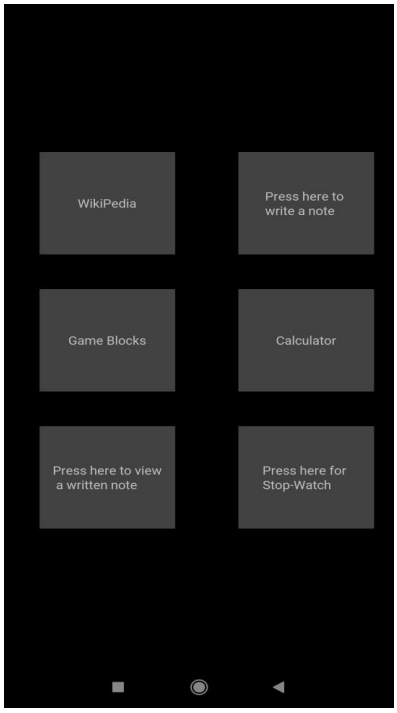

Fig.5.2 Working of app

TESTING

Output obtained when searched about python kivy in Wikipedia

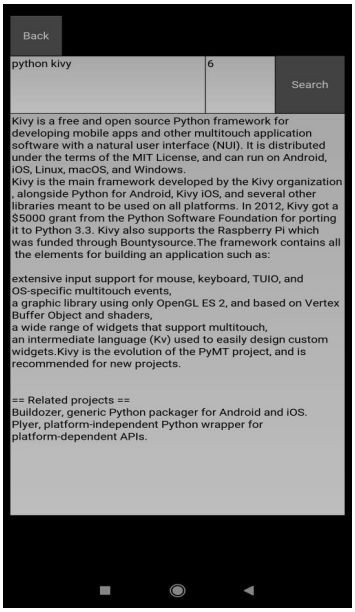

Fig.6.1 Wikipedia output

Image showing time on the stop watch

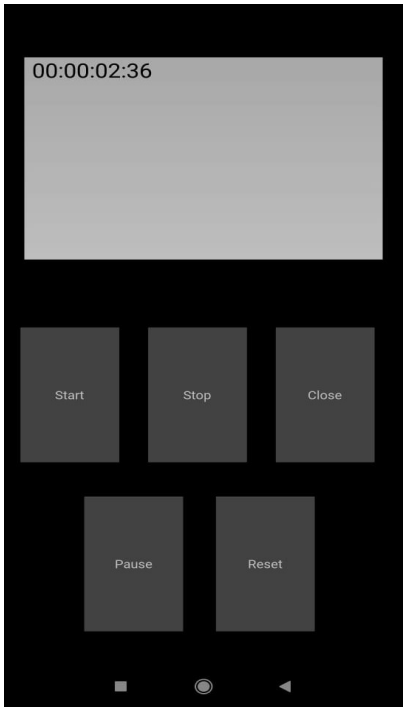

Fig.6.2 Stop watch output

Image showing text from a file that is saved

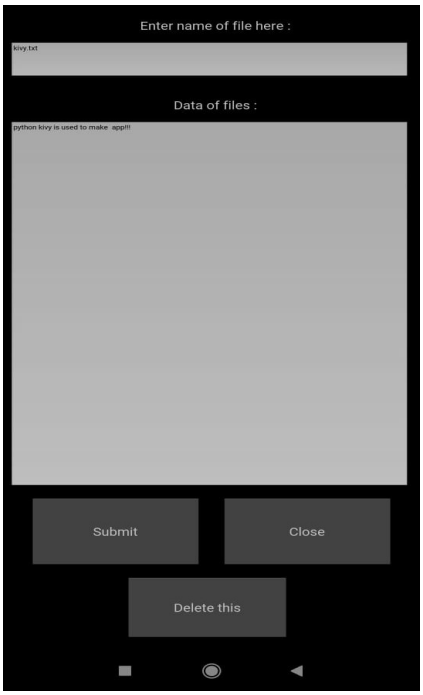

Fig.6.3 Output of notes viewer

Result of an arithmetic expression:

Fig.6.4, 6.5 Execution of calculator output.

CONCLUSION / FUTURE WORK

In conclusion, the aim behind the project is to develop a virtual assistant like application that helps the users to simplify their tasks along with some additional features for attraction and the app does that. The lite version of this application is developed with 6 features for the users and the main project will have even more features with improved working and design.

The future work of the application includes adding more important features such as the alarm, task reminder and a smart calendar design along with other possible features as needed by the users. The future work also includes extending the use of application scope from android to other operating systems.

The future work tries to make this application a full-scale project using android services that can allow a lot of other features to be included. The final aim of the project is to make a really helpful virtual assistant that can be installed on mobile devices which helps the users with a wide range of extraordinary features all in one single application to improve the efficiency by trying to coordinate the uses of different features.

REFERENCES

1. Kivy : Kivy is an open source software library for the rapid development of applications equipped with novel user interfaces, such as multi-touch apps.
   URL of official documentation : https://kivy.org/doc/stable/

2. URL of Buildozer official documentation : https://buildozer.readthedocs.io/en/latest/

3. .https://www.youtube.com/watch?v=WkGFqxHoOfI
   A youtube video by Edureka, an e-learning platform.

4. URL of official documentation of wikipedia module of python:
   https://pypi.org/project/wikipedia/