CONVO-LYSER

MOBILE APPLICATION DEVELOPMENT

(IT7013)

MEMBERS-

Kirupalini S (2018506054)

Gowri S (2018506034)

Swetha S (2018506134)

Amruthavarshini R (2018506015)

Aditi Baskar (2018506006)

AIM-

To detect and extract various features from chats including polarity, sentiment, category, semantic relationships and bot detection. Additionally the conversations are also checked for paedophilic content.

DESCRIPTION-

This app has been developed using KivyMD framework (Python) using the PyCharm IDE. The main aim of this app is to analyse conversations for grooming characteristics and subsequently report them to the concerned authorities. The app also contains a portal for users to report abusive and offensive chats. The features of this app include-

1. Account Verification
2. Polarity analyser
3. Sentiment analyser
4. Aspect analyser
5. Semantic Analyser
6. Word Cloud
7. Dynamic statistics
8. Grooming Detection
9. Report Portal

Account Verification- Once the user signs up, his or her account is verified. The user needs to upload an ID proof containing their photo and date of birth. The app then requires camera permission to take a picture of the user. Using the photo in the ID proof and the current photo, the user is verified. Additionally, the age of the user is also verified using Photo OCR, so that the conversations uploaded by users below the age of 18 (minors) can be checked for grooming characteristics. OCR is implemented using Tesseract and facial recognition is implemented using an inbuilt package face_recognition.

Polarity, Sentiment and Aspect based analysis- Sentiment analysis or opinion mining combines NLP and ML techniques to analyse unstructured text data for opinions and emotions. IBM Watson tone analyser and Natural

Language Understanding packages are used to analyse the conversation for polarity, emotions and aspects.

Semantic Analyser- Using Latent Dirichlet Allocation (LDA), words that are frequently used together in the conversation are grouped. LDA is an unsupervised method of performing topic modelling as the topics are not specified beforehand. The grouping is depicted with the help of a tree map where each section represents a particular grouping along with the corresponding words.

Word Cloud- The word cloud is composed of frequently used words in the conversation. The size of each word denotes its frequency, i.e., how often it is used. Therefore, the more frequently a word is used in the conversation, the larger it appears in the word cloud. This feature is implemented with the help of the inbuilt package wordcloud.

Dynamic statistics- This Module is meant to display women related news exclusively in the page. It uses BeautifulSoup to scrap the news from the news website. The links of the news is then copied on to a csv file. By importing Article from Newspaper3k package,the article is downloaded from the link, parsed using nlp and then the title of the articles are displayed as headlines.

Grooming Detection- A dictionary consisting of the words from the LIWC 2007 dictionary, common slang words used on social media platforms and sexual and abusive words were collected and split across the six grooming stages. The words in each line of the conversation were assigned a stage and the highest stage was recorded for the corresponding lines. The count of these six stages (S1- S6) are given as input to the classifier. A Support Vector Machine (SVM) was used to classify the conversation as grooming or non-grooming based on the features determined by the LIWC process. In case a conversation is found to display grooming characteristics, the concerned authorities will be informed.

Report Portal-The users can also voluntarily report incidents of abusive or offensive chats. They will have to provide few such instances along with the correspondent’s number. The report will be forwarded to the concerned authorities and they will get in touch with the reporter.

SOURCE CODE-

Helpers.py

```
lusername_input = """
MDTextField:
    hint_text: "Enter username"
    helper_text: "Enter the full name"
    helper_text_mode: "on_focus"
    required: True
    icon_right: "human"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.5}
    size_hint_x:None
    width:250
"""
lpassword_input ="""
MDTextField:
    hint_text: "Enter password"
    helper_text: "Check if caps lock is on"
    helper_text_mode: "on_focus"
    password: True
    required: True
    icon_right: "lock-outline"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.4}
    size_hint_x:None
    width:250
"""



username_input = """
MDTextField:
    hint_text: "Enter username"
    helper_text: "Enter the full name"
    helper_text_mode: "on_focus"
    required: True
    icon_right: "human"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.7}
    size_hint_x:None
    width:250
"""
mycontact_input = """
MDTextField:
    hint_text: "Enter contact number"
    helper_text: "Number must contain 10 digits"
    helper_text_mode: "on_focus"
    required: True
    icon_right: "cellphone"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.6}
    size_hint_x:None
    width:250
"""
email_input = """
MDTextField:
    hint_text: "Enter email id"
    helper_text: "Optional"
    helper_text_mode: "on_focus"
    required: True
    icon_right: "email"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.5}
    size_hint_x:None
    width:250
"""
password_input ="""
MDTextField:
    hint_text: "Enter password"
    helper_text: "Password must contain at least 8 letter."
    helper_text_mode: "on_focus"
    password: True
    required: True
    icon_right: "lock-outline"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.4}
    size_hint_x:None
    width:250
"""
abusername_input = """
MDTextField:
    hint_text: "Enter abuser name"
    helper_text: "Enter the full name"
    helper_text_mode: "on_focus"
    icon_right: "human"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.6}
    size_hint_x:None
    width:250
"""
contact_input = """
MDTextField:
    hint_text: "Enter abusers contact number"
    helper_text: "Number must contain 10 digits"
    helper_text_mode: "on_focus"
    required: True
    icon_right: "cellphone"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.5}
    size_hint_x:None
    width:250
"""
reason_input = """
MDTextField:
    hint_text: "Enter reason"
    helper_text: "Include some of the abusive messages"
    helper_text_mode: "on_focus"
    required: True
    multiline: True
    icon_right: "information"
    icon_right_color: [1,1,1,1]
    pos_hint:{'center_x': 0.5, 'center_y': 0.4}
    size_hint_x:None
    width:250
"""
```

Main3.py

```
# Final report portal screen linked to home screen with navigation drawer
from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.core.window import Window
from kivymd.theming import ThemableBehavior
from kivymd.uix.boxlayout import BoxLayout
from kivy.uix.scrollview import ScrollView
from kivy.uix.screenmanager import ScreenManager
from kivymd.uix.button import MDRectangleFlatButton, MDFlatButton, MDTextButton
from kivymd.uix.dialog import MDDialog
from kivymd.uix.list import MDList
from kivy.graphics import Color
from kivy.utils import get_color_from_hex
import helpers
import re

regex = '^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$'

Window.size = (300, 500)

navigation_helper = """
Screen:
    id: home
    name: 'home'
    NavigationLayout:
        ScreenManager:
            Screen:
                BoxLayout:
                    orientation: 'vertical'
                    MDToolbar:
                        title: "HOME"
                        elevation: 10
                        left_action_items: [['menu', lambda x: nav_drawer.toggle_nav_drawer()]]

                    Widget:

                    DrawerList:
                        id: features

                        MDList:
                            OneLineIconListItem:
                                text: "Conversational Analyser"

                                IconLeftWidget:
                                    icon: "chat-processing"

                            OneLineIconListItem:
                                text: "Stats"
                                IconLeftWidget:
                                    icon: "chart-line"

                            OneLineIconListItem:
                                text: "Report Portal"
                                on_release: root.manager.current = 'portal'
                                IconLeftWidget:
                                    icon: "notebook-outline"
                                    on_release: root.manager.current = 'portal'
                    ScrollView:
                    MDBottomNavigation:


                        MDBottomNavigationItem:
                            name: 'contact'
                            icon: 'phone'



                        MDBottomNavigationItem:
                            name: 'query'
                            icon: 'comment-question'

                        MDBottomNavigationItem:
                            name: 'account'
                            icon: 'account'

                        MDBottomNavigationItem:
                            name: 'settings'
                            icon: 'settings'


        MDNavigationDrawer:
            id: nav_drawer


            ContentNavigationDrawer:
                orientation: 'vertical'
                padding: "8dp"
                spacing: "8dp"

                MDLabel:
                    text: "CONVO-LYSER"


                    theme_text_color: "Custom"
                    text_color: 1, 1, 1, 1
                    font_style: "H6"
                    size_hint_y: None
                    height: self.texture_size[1]

                ScrollView:
                    DrawerList:
                        id: md_list

                        MDList:
                            OneLineIconListItem:
                                text: "How to use"

                                IconLeftWidget:
                                    icon: "help"

                            OneLineIconListItem:
                                text: "FAQ"

                                IconLeftWidget:
                                    icon: "frequently-asked-questions"

                            OneLineIconListItem:
```

```
                text: "About this app"

                IconLeftWidget:
                    icon: "application"

            OneLineIconListItem:
                text: "Privacy policy"

                IconLeftWidget:
                    icon: "security"

            OneLineIconListItem:
                text: "Logout"

                IconLeftWidget:
                    icon: "login"


"""

screen_helper = """
Screen:
    id: portal
    name: 'portal'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'Report Portal'
            left_action_items: [["alert-circle-outline", lambda x: app.info()]]
            elevation:10

        Widget:



        MDBottomAppBar:
            MDToolbar:
                icon: 'home'
                type: 'bottom'
                on_action_button: root.manager.current = 'home'


"""

login_helper = """
Screen:
    id: login
    name: 'login'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'Login'
            elevation:10

        Widget:

        MDTextButton:
            text: 'Click here to sign up'
            pos_hint: {'center_x': 0.5, 'center_y': 0.3}
            on_press: root.manager.current = 'signup'
        MDBottomAppBar:
            MDToolbar:
                icon: 'kodi'
                type: 'bottom'



"""

signup_helper = """
Screen:
    id: signup
    name: 'signup'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'Signup'
            elevation:10

        Widget:



        MDBottomAppBar:
            MDToolbar:
                icon: 'kodi'
                type: 'bottom'



"""


class ContentNavigationDrawer(BoxLayout):
    pass


class DrawerList(ThemableBehavior, MDList):
    pass


sm = ScreenManager()


class DemoApp(MDApp):

    def build(self):
        self.theme_cls.primary_palette = "Blue"
        self.theme_cls.primary_hue = "500"  # "700"
        self.theme_cls.theme_style = "Light"

        screen = Builder.load_string(login_helper)
        self.lusername = Builder.load_string(helpers.lusername_input)
        self.lpassword = Builder.load_string(helpers.lpassword_input)
        button = MDRectangleFlatButton(text='Submit',
                                       pos_hint={'center_x': 0.5, 'center_y': 0.3},
                                       on_release=self.log_show_data
                                       )
        screen.add_widget(self.lusername)
```

```python
        screen.add_widget(self.lpassword)
        screen.add_widget(button)
        sm.add_widget(screen)

        screen = Builder.load_string(signup_helper)
        self.username = Builder.load_string(helpers.username_input)
        self.mycontact = Builder.load_string(helpers.mycontact_input)
        self.email = Builder.load_string(helpers.email_input)
        self.password = Builder.load_string(helpers.password_input)
        button = MDRectangleFlatButton(text='Submit',
                                       pos_hint={'center_x': 0.5, 'center_y': 0.3},
                                       on_release=self.sign_show_data
                                       )

        screen.add_widget(self.username)
        screen.add_widget(self.mycontact)
        screen.add_widget(self.email)
        screen.add_widget(self.password)
        screen.add_widget(button)
        sm.add_widget(screen)

        screen = Builder.load_string(navigation_helper)
        sm.add_widget(screen)

        screen = Builder.load_string(screen_helper)
        self.abusername = Builder.load_string(helpers.abusername_input)
        self.contact = Builder.load_string(helpers.contact_input)
        self.reason = Builder.load_string(helpers.reason_input)
        button = MDRectangleFlatButton(text='Submit',
                                       pos_hint={'center_x': 0.5, 'center_y': 0.3},
                                       on_release=self.show_data)
        screen.add_widget(self.abusername)
        screen.add_widget(self.contact)
        screen.add_widget(self.reason)
        screen.add_widget(button)
        sm.add_widget(screen)
        return sm

    def show_data(self, obj):

        if self.contact.text != "" and self.reason.text != "":
            if len(self.contact.text) == 10 and self.contact.text.isdigit():
                print("ABUSER NAME- " + self.abusername.text)
                print("CONTACT NUMBER- " + self.contact.text)
                print("REASON- " + self.reason.text)
                # self.reason.text, self.contact.text, self.username.text = ""
                self.abusername.text = ""
                self.contact.text = ""
                self.reason.text = ""
                user_error = "Your response has been noted. The immediate responders will contact you soon."
            else:
                user_error = "Please enter a valid contact number."
        else:
            user_error = "Please enter the required fields"
        self.dialog = MDDialog(
            text=user_error, size_hint=(0.8, 1),
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()
    def close_dialog(self, obj):
        self.dialog.dismiss()
        # do stuff after closing the dialog

    def info(self):
        self.dialog = MDDialog(
            text='The information entered below will be forwarded to the respective authorities.', size_hint=(0.8, 1),
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def log_show_data(self, obj):
        if self.lusername.text != "" and self.lpassword.text != "":
            if len(self.lpassword.text) >= 8:
                sm.switch_to(Builder.load_string(navigation_helper))
            else:
                user_error = "Incorrect password. Please try again"
                self.dialog = MDDialog(
                    text=user_error, size_hint=(0.8, 1),
                    buttons=[MDFlatButton(text='Close', on_release=self.close_dialog), ]
                )

                self.dialog.open()



        else:
            user_error = "Please enter the required details"
            self.dialog = MDDialog(
                text=user_error, size_hint=(0.8, 1),
                buttons=[MDFlatButton(text='Close', on_release=self.close_dialog), ]
            )

            self.dialog.open()

    def close_dialog1(self, obj):
        self.dialog.dismiss()

        # do stuff after closing the dialog

    def sign_show_data(self, obj):

        if self.username.text != "" and self.mycontact.text != "" and self.email.text != "" and self.password.text !=
"":
            if len(self.mycontact.text) == 10 and self.mycontact.text.isdigit():
                if re.search(regex, self.email.text):
                    if len(self.password.text) >= 8:

                        print("USERNAME- " + self.username.text)
                        print("CONTACT NUMBER- " + self.mycontact.text)
                        print("EMAIL- " + self.email.text)
                        print("PASSWORD- " + self.password.text)
                        # self.reason.text, self.contact.text, self.username.text = ""
                        self.username.text = ""
                        self.mycontact.text = ""
                        self.email.text = ""
                        user_error = ""
                    else:
                        user_error = "Please enter a valid password"
                else:
```

```
                user_error = "Please enter a valid email id"
            else:
                user_error = "Please enter a valid contact number."


        else:
            user_error = "Please enter the required fields"
        if user_error == "":
            sm.switch_to(Builder.load_string(navigation_helper))
        else:
            self.dialog = MDDialog(
                text=user_error, size_hint=(0.8, 1),
                buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
            )
            self.dialog.open()


DemoApp().run()
```

Sa_fet.py

```
from kivymd.app import MDApp
from kivymd.theming import ThemableBehavior
from kivy.uix.screenmanager import ScreenManager
from kivymd.uix.list import MDList
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDRectangleFlatButton, MDFlatButton, MDTextButton
from kivy.lang import Builder
from kivy.core.window import Window
from kivy.uix.boxlayout import BoxLayout
from kivymd.uix.taptargetview import MDTapTargetView

Window.size = (300, 500)
conv_anal = """
Screen:
    id: home
    name: 'home'
    NavigationLayout:
        ScreenManager:
            Screen:

                BoxLayout:
                    orientation: 'vertical'
                    MDToolbar:
                        title: 'CONVO-LYSER'
                        left_action_items: [['alert-circle-outline', lambda x: app.info4()]]
                        elevation:2
                    MDLabel:
                        text: ""
                        font_style:"H6"
                        halign: "center"

                    MDLabel:
                        text: "272"
                        font_style:"H1"
                        halign: "center"
                    MDLabel:
                        text: "MESSAGES IN TOTAL"
                        halign: "center"
            BoxLayout:
                orientation: 'horizontal'
                MDFloatingActionButton:
                    id: button
                    icon: "plus"
                    pos: 10, 10
                    on_release: app.tap_target_start1()
            DrawerList:
                id: features
                MDList:
                    OneLineIconListItem:
                        text: "SENTIMENT ANALYSIS"
                        on_release: root.manager.current = 'sent'
                        IconLeftWidget:
                            icon: "chart-pie"
                            on_release: root.manager.current = 'sent'

                    OneLineIconListItem:
                        text: "EMOTIONAL ANALYSIS"
                        on_release: root.manager.current = 'emo'
                        IconLeftWidget:
                            icon: "chart-line"
                            on_release: root.manager.current = 'emo'

                    OneLineIconListItem:
                        text: "ASPECT BASED ANALYSIS"
                        on_release: root.manager.current = 'aspect'
                        IconLeftWidget:
                            icon: "chart-bar"
                            on_release: root.manager.current = 'aspect'

            ScrollView:
            MDBottomNavigation:

                MDBottomNavigationItem:
                    name: 'query'
                    icon: 'comment-question'

                MDBottomNavigationItem:
                    name: 'home'
                    icon: 'home'


MDNavigationDrawer:
    id: nav_drawer


    ContentNavigationDrawer:
        orientation: 'vertical'
        padding: "8dp"
        spacing: "8dp"

        MDLabel:
            text: "CONVO-LYSER"


            theme_text_color: "Custom"
            text_color: 1, 1, 1, 1
            font_style: "H6"
            size_hint_y: None
```

```
          height: self.texture_size[1]


"""
screen_helper1 = """
Screen:
    id: sent
    name: 'sent'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'SENTIMENTAL ANALYSIS'
            left_action_items: [["alert-circle-outline", lambda x: app.info1()]]
            elevation:10
        MDLabel:
            text: ""
            font_style:"H6"
            halign: "center"

        MDLabel:
            text: "POSITIVE: 9.98%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        MDLabel:
            text: "NEGATIVE: 65.3%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        MDLabel:
            text: "NEUTRAL: 24.7%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        Widget:
        MDBottomAppBar:
            MDToolbar:
                icon: 'home'
                type: 'bottom'
                on_action_button: root.manager.current = 'home'


"""
screen_helper2 = """
Screen:
    id: emo
    name: 'emo'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'EMOTIONAL ANALYSIS'
            left_action_items: [["alert-circle-outline", lambda x: app.info2()]]
            elevation:10
        MDLabel:
            text: ""
            font_style:"H6"
            halign: "center"
        MDLabel:
            text: "SADNESS: 29%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        MDLabel:
            text: "JOY: 20%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        MDLabel:
            text: "FEAR: 10%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        MDLabel:
            text: "DISGUST: 26%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        MDLabel:
            text: "ANGER: 15%"
            halign: "center"
            font_style: "H5"
            theme_text_color: "Custom"
        Widget:
        MDBottomAppBar:
            MDToolbar:
                icon: 'home'
                type: 'bottom'
                on_action_button: root.manager.current = 'home'


"""

screen_helper3 = """
Screen:
    id: aspect
    name: 'aspect'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'ASPECT BASED ANALYSIS'
            left_action_items: [["alert-circle-outline", lambda x: app.info3()]]
            elevation:10
        MDLabel:
            text: ""
            font_style:"H6"
            halign: "center"
        MDLabel:
           text: "ART & ENTERTAINMENT"
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        MDLabel:
            text: "HEALTH AND FITNESS"
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        MDLabel:
            text: "SOCIETY"
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        MDLabel:
            text: "EDUCATION"
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        MDLabel:
            text: "HOBBIES & INTERESTS"
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        MDLabel:
            text: "SPORTS"
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        MDLabel:
            text: "STYLE & FASHION"
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        Widget:
        MDBottomAppBar:
            MDToolbar:
                icon: 'home'
                type: 'bottom'
                on_action_button: root.manager.current = 'home'


"""


class ContentNavigationDrawer(BoxLayout):
    pass


class DrawerList(ThemableBehavior, MDList):
    pass


sm = ScreenManager()


class conversation_analyser(MDApp):

    def build(self):
        screen = Builder.load_string(conv_anal)
        self.tap_target_view = MDTapTargetView(widget=screen.ids.button, title_text="USER 1      USER 2",
                                               description_text="   118                  154 \nMESSAGES    MESSAGES",
                                               widget_position="center", title_position="right_top",
                                               title_text_size="20sp", outer_radius=250, )
        # self.tap_target_view = MDTapTargetView(widget=screen.ids.button,title_text="USER
2",description_text="154 MESSAGES",widget_position="right", outer_radius=100,)
        sm.add_widget(screen)
        screen = Builder.load_string(screen_helper1)
        sm.add_widget(screen)
        screen = Builder.load_string(screen_helper2)
        sm.add_widget(screen)
        screen = Builder.load_string(screen_helper3)
        sm.add_widget(screen)
        return sm

    def tap_target_start1(self):
        if self.tap_target_view.state == "close":
            self.tap_target_view.start()
        else:
            self.tap_target_view.stop()

    def tap_target_start2(self):
        if self.tap_target_view.state == "close":
            self.tap_target_view.start()
        else:
            self.tap_target_view.stop()

    def close_dialog(self, obj):
        self.dialog.dismiss()
        # do stuff after closing the dialog

    def info1(self):
        self.dialog = MDDialog(
            text='Sentimental Analysis displays the polarity ie. positive, negative and neutral polarity values of the
whole conversation',
            size_hint=(0.9, 0.5), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def info2(self):
        self.dialog = MDDialog(
            text='Emotional Analysis displays the various emotions and their value of the whole conversation',
            size_hint=(0.9, 0.5), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def info3(self):
        self.dialog = MDDialog(
            text='Aspect Based Analysis displays the most talked category of the whole conversation',
            size_hint=(0.9, 0.5), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def info4(self):
        self.dialog = MDDialog(
            text='Analysis on emotions and tones of conversations and visualise the results in the form of graphs.',
            size_hint=(1, 0), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def callback(self, instance):
        print("Button is pressed")
        print('The button % s state is <%s>' % (instance, instance.state))


root = conversation_analyser()
root.run()
```

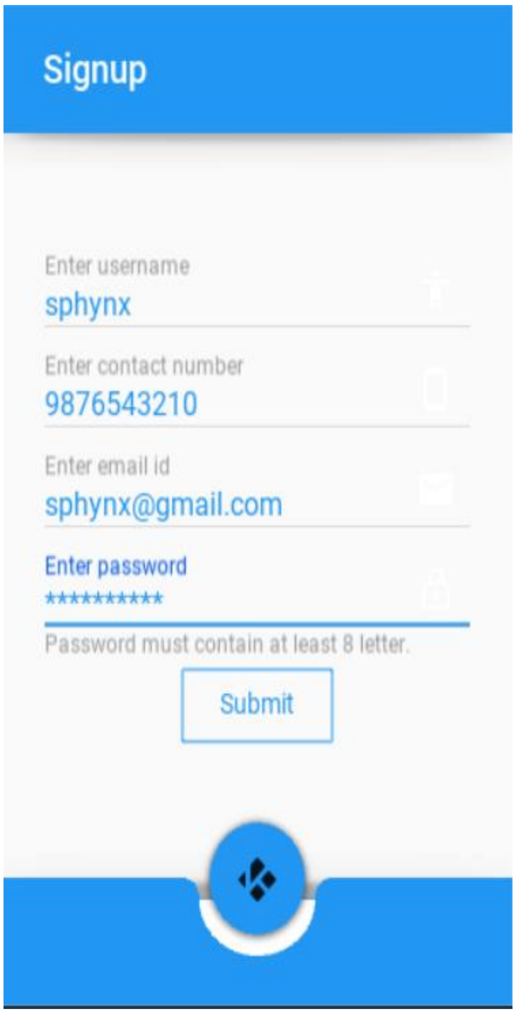
Stats.py

```
from kivymd.app import MDApp
from kivymd.theming import ThemableBehavior
from kivy.uix.screenmanager import ScreenManager
from kivymd.uix.list import MDList
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDRectangleFlatButton, MDFlatButton, MDTextButton
from kivy.lang import Builder
from kivy.core.window import Window
from kivy.uix.boxlayout import BoxLayout
from kivymd.uix.taptargetview import MDTapTargetView

Window.size = (300, 500)
newsscraping = """
Screen:
    id: home
    name: 'home'
    NavigationLayout:
        ScreenManager:
            Screen:
                BoxLayout:
                    orientation: 'vertical'
                    MDToolbar:
                        title: 'TRENDING ARTICLES ON WOMEN'
                        left_action_items: [['alert-circle-outline', lambda x: app.info4()]]
                        elevation:2
                    MDLabel:
                        text: ""
                        font_style:"H6"
                        halign: "center"
                    MDLabel:
                        text: "Active Social Media Users"
                        halign: "center"
                    MDLabel:
                        text: "3.80"
                        font_style:"H3"
                        halign: "center"
                    MDLabel:
                        text: "BILLION"
                        halign: "center"
                    BoxLayout:
                        orientation: 'horizontal'
                        MDFloatingActionButton:
                            id: button
                            icon: "plus"
                            pos: 10, 10
                            on_release: app.tap_target_start1()
                    DrawerList:
                        id: features
                        MDList:
                            OneLineIconListItem:
                                text: "Iraq urged to investigate attacks on women human rights defenders"
                                on_release: root.manager.current = 'sent'
                                IconLeftWidget:
                                    icon: "square"
                                    on_release: root.manager.current = 'sent'
                            OneLineIconListItem:
                                text: "Thailand: More than 100 companies pledge to strengthen women's economic
empowerment"
```

```
                        on_release: root.manager.current = 'emo'
                        IconLeftWidget:
                            icon: "square"
                            on_release: root.manager.current = 'emo'
                    OneLineIconListItem:
                        text: "Most countries failing to protect women from COVID-19 economic and social
fallout"
                        on_release: root.manager.current = 'aspect'
                        IconLeftWidget:
                            icon: "square"
                            on_release: root.manager.current = 'aspect'
                ScrollView:
                MDBottomNavigation:
                    MDBottomNavigationItem:
                        name: 'query'
                        icon: 'comment-question'
                    MDBottomNavigationItem:
                        name: 'home'
                        icon: 'home'
        MDNavigationDrawer:
            id: nav_drawer
            ContentNavigationDrawer:
                orientation: 'vertical'
                padding: "8dp"
                spacing: "8dp"
                MDLabel:
                    text: "ARTICLES ON WOMEN"
                    theme_text_color: "Custom"
                    text_color: 1, 1, 1, 1
                    font_style: "H6"
                    size_hint_y: None
                    height: self.texture_size[1]
"""
screen_helper1 = """
Screen:
    id: sent
    name: 'sent'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'Iraq urged to investigate attacks on women human rights defenders'
            left_action_items: [["alert-circle-outline", lambda x: app.info1()]]
            elevation:10
        MDLabel:
            text: ""
            font_style:"H6"
            halign: "center"
        MDLabel:
            text: "UN-appointed independent rights experts have urged the Iraqi authorities to investigate the murder
of a female human rights defender, and the attempted killing of another, targeted “simply because they are
women”."
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"

        Widget:
        MDBottomAppBar:
            MDToolbar:
                icon: 'home'
                type: 'bottom'
                on_action_button: root.manager.current = 'home'
"""
screen_helper2 = """
Screen:
    id: emo
    name: 'emo'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'Thailand: More than 100 companies pledge to strengthen women’s economic empowerment'
            left_action_items: [["alert-circle-outline", lambda x: app.info2()]]
            elevation:10
        MDLabel:
            text: ""
            font_style:"H6"
            halign: "center"
        MDLabel:
            text: "Chief executives of 110 companies in Thailand on Wednesday, have signed up to a new set of UN
principles on women’s economic empowerment, pledging to improve gender equality in the boardroom, equal
pay for equal work, and safer and more inclusive workplaces."
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"

        Widget:
        MDBottomAppBar:
            MDToolbar:
                icon: 'home'
                type: 'bottom'
                on_action_button: root.manager.current = 'home'
"""

screen_helper3 = """
Screen:
    id: aspect
    name: 'aspect'
    BoxLayout:
        orientation: 'vertical'
        MDToolbar:
            title: 'Most countries failing to protect women from COVID-19 economic and social fallout'
            left_action_items: [["alert-circle-outline", lambda x: app.info3()]]
            elevation:10
        MDLabel:
            text: ""
            font_style:"H6"
            halign: "center"
        MDLabel:
           text: "The COVID-19 pandemic is “hitting women hard”, but most nations are failing to provide
sufficient social and economic protection for them, the head of the UN gender empowerment agency said on
Monday."
            halign: "center"
            font_style: "H6"
            theme_text_color: "Custom"
        Widget:
        MDBottomAppBar:
            MDToolbar:
                icon: 'home'
                type: 'bottom'
                on_action_button: root.manager.current = 'home'
"""
```

```
class ContentNavigationDrawer(BoxLayout):
    pass


class DrawerList(ThemableBehavior, MDList):
    pass


sm = ScreenManager()


class news_scrape(MDApp):

    def build(self):
        screen = Builder.load_string(newsscraping)
        self.tap_target_view = MDTapTargetView(widget=screen.ids.button,
                                               title_text="Teens having \nat least 1 social \nmedia profile",
                                               description_text="   75%                  ",

                                               widget_position="center", title_position="right_top",
                                               title_text_size="20sp", outer_radius=250, )
        # self.tap_target_view = MDTapTargetView(widget=screen.ids.button,title_text="USER
2",description_text="154 MESSAGES",widget_position="right", outer_radius=100,)
        sm.add_widget(screen)
        screen = Builder.load_string(screen_helper1)
        sm.add_widget(screen)
        screen = Builder.load_string(screen_helper2)
        sm.add_widget(screen)
        screen = Builder.load_string(screen_helper3)
        sm.add_widget(screen)
        return sm

    def tap_target_start1(self):
        if self.tap_target_view.state == "close":
            self.tap_target_view.start()
        else:
            self.tap_target_view.stop()

    def tap_target_start2(self):
        if self.tap_target_view.state == "close":
            self.tap_target_view.start()
        else:
            self.tap_target_view.stop()

    def close_dialog(self, obj):
        self.dialog.dismiss()
        # do stuff after closing the dialog

    def info1(self):
        self.dialog = MDDialog(
            text='2nd October, 2020\nHuman Rights',
            size_hint=(0.9, 0.5), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def info2(self):
        self.dialog = MDDialog(
            text='30th September, 2020\nWomen Empowerment',
            size_hint=(0.9, 0.5), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def info3(self):
        self.dialog = MDDialog(
            text='28th September, 2020\nWomen Safety',
            size_hint=(0.9, 0.5), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def info4(self):
        self.dialog = MDDialog(
            text='This section shows the latest articles on child and women empowerment, abuse, welfare and such
obtained from the UN Women Updates.',
            size_hint=(1, 0), radius=[20, 7, 20, 7],
            buttons=[MDFlatButton(text='Close', on_release=self.close_dialog)]
        )
        self.dialog.open()

    def callback(self, instance):
        print("Button is pressed")
        print('The button % s state is <%s>' % (instance, instance.state))


root = news_scrape()
root.run()
```

SNAPSHOTS-


Signup

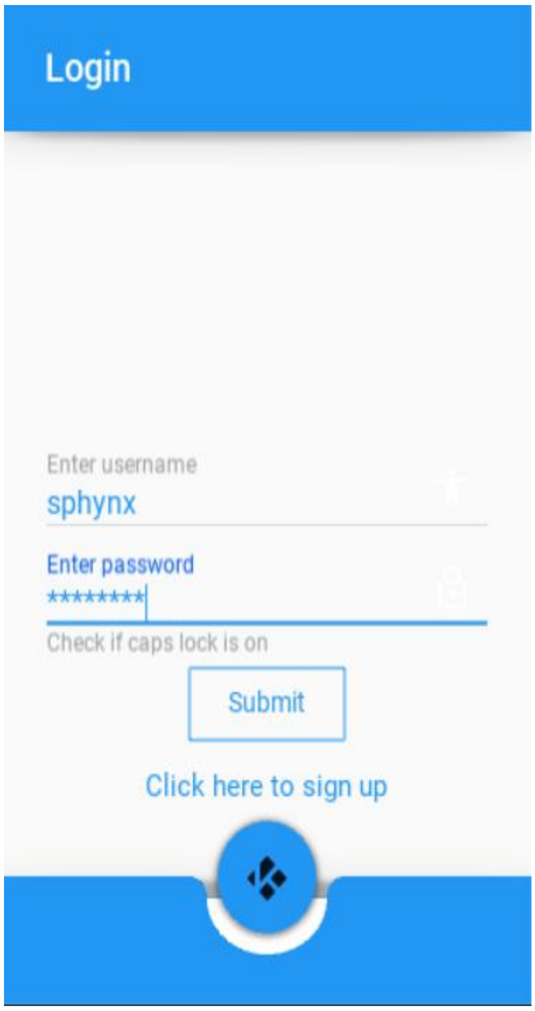
Enter username
sphynx

Enter contact number
9876543210

Enter email id
sphynx@gmail.com

Enter password
****

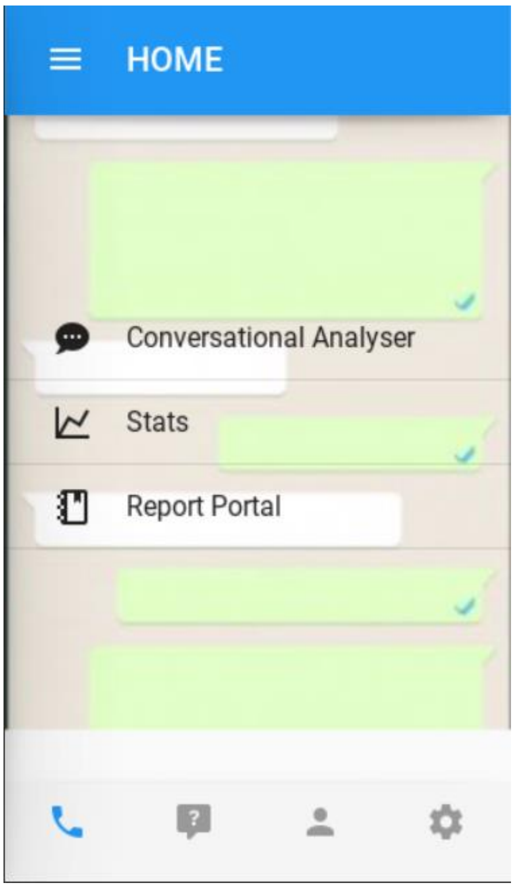
Password must contain at least 8 letter.

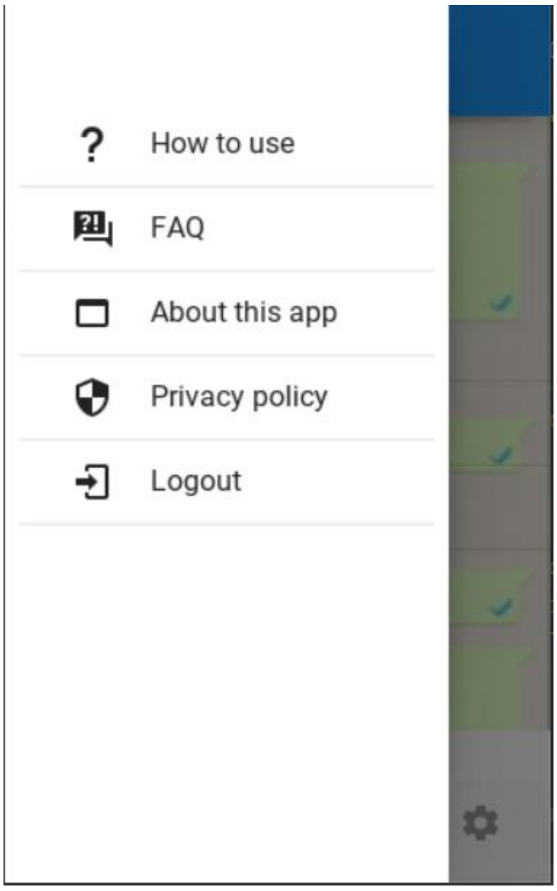

Login

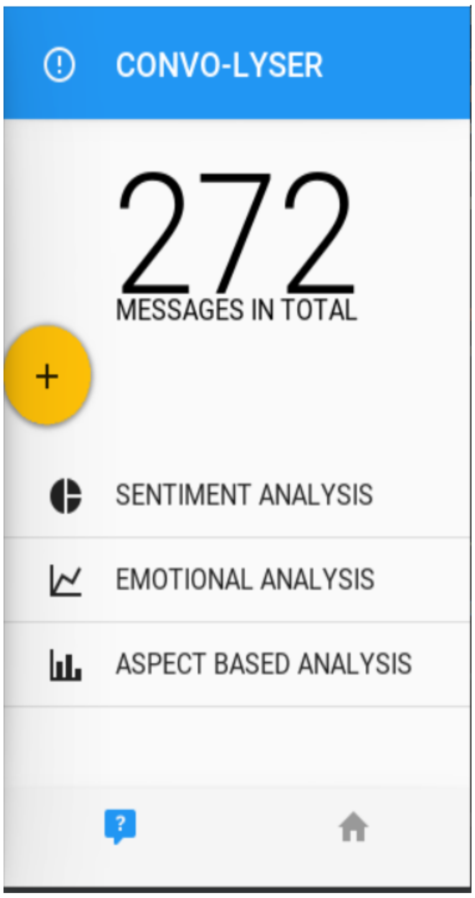
Enter username
sphynx

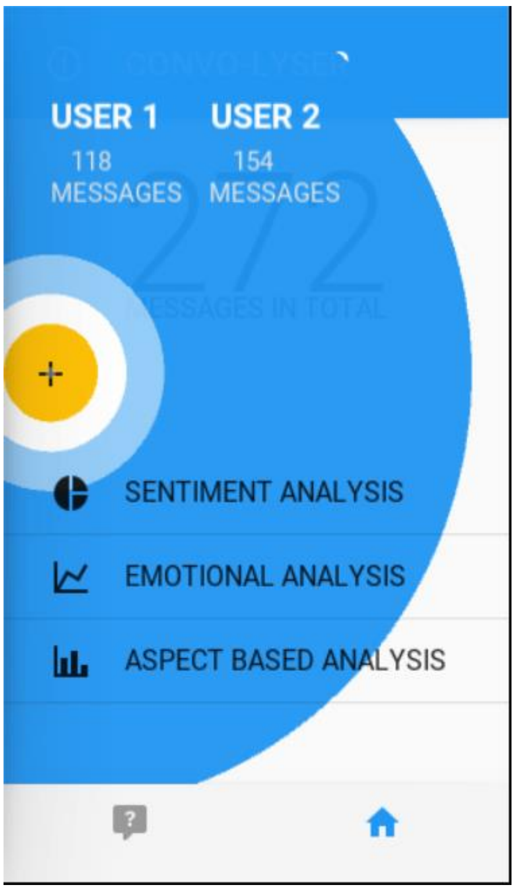
Enter password
****

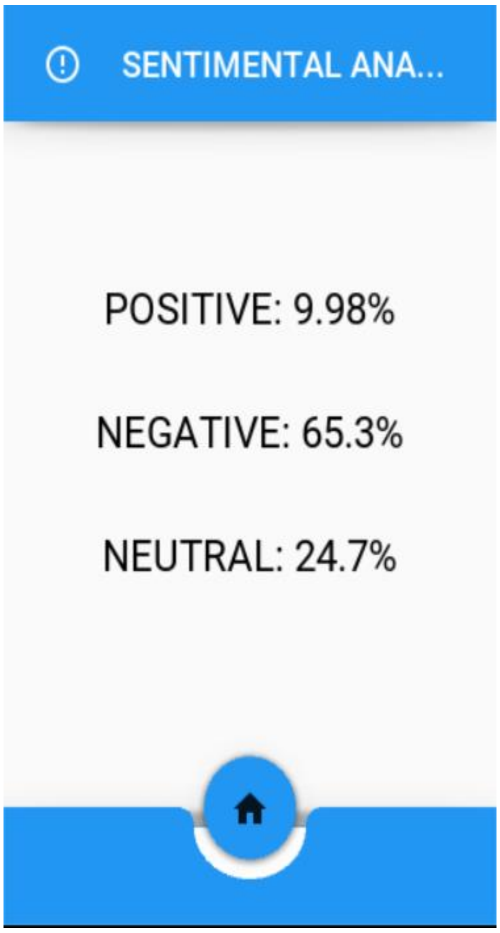
Check if caps lock is on


HOME
Conversational Analyser
Stats
Report Portal


How to use
FAQ
About this app
Privacy policy
Logout


CONVO-LYSER
272
MESSAGES IN TOTAL
+
SENTIMENT ANALYSIS
EMOTIONAL ANALYSIS
ASPECT BASED ANALYSIS

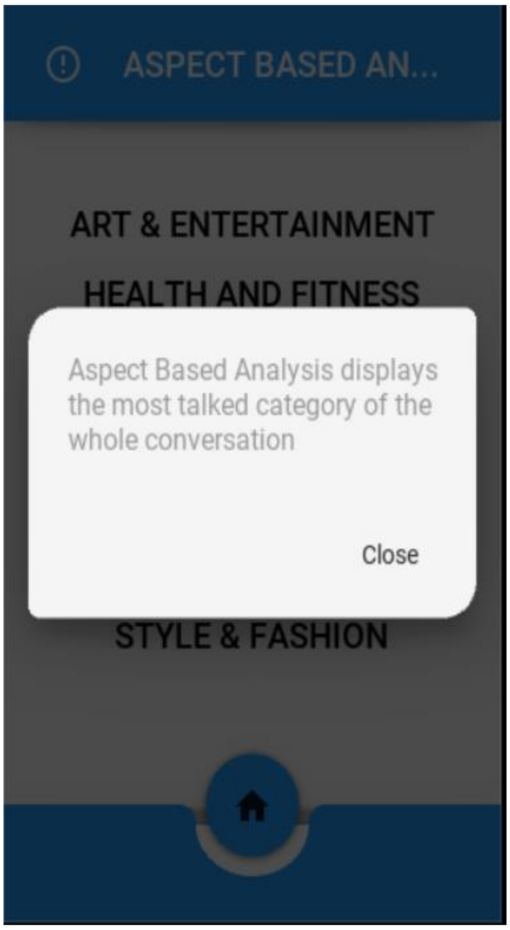

USER 1
118
MESSAGES
USER 2
154
MESSAGES
+
SENTIMENT ANALYSIS
EMOTIONAL ANALYSIS
ASPECT BASED ANALYSIS


SENTIMENTAL ANA...

POSITIVE: 9.98%

NEGATIVE: 65.3%

NEUTRAL: 24.7%


ASPECT BASED AN...

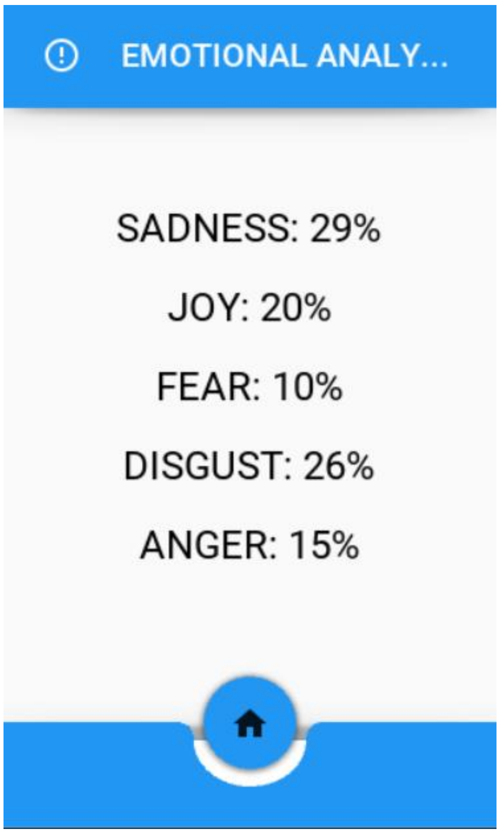
ART & ENTERTAINMENT

HEALTH AND FITNESS

Aspect Based Analysis displays the most talked category of the whole conversation

Close

STYLE & FASHION


EMOTIONAL ANALY...

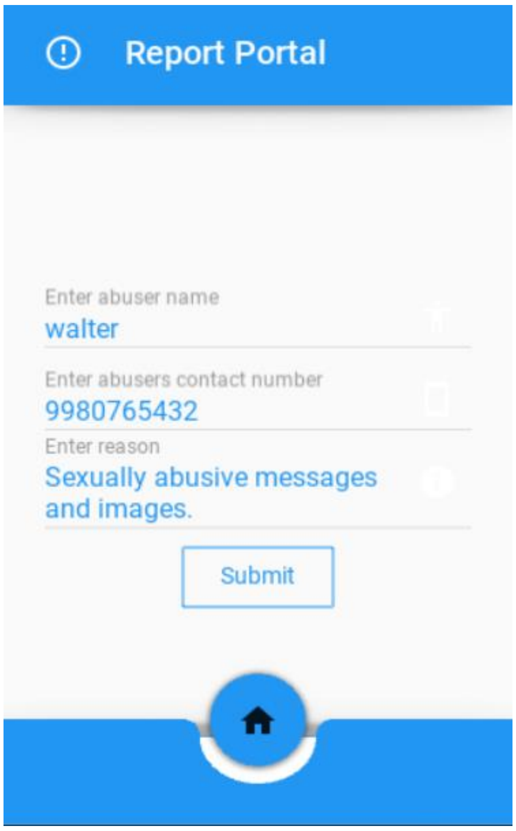
SADNESS: 29%

JOY: 20%

FEAR: 10%

DISGUST: 26%

ANGER: 15%

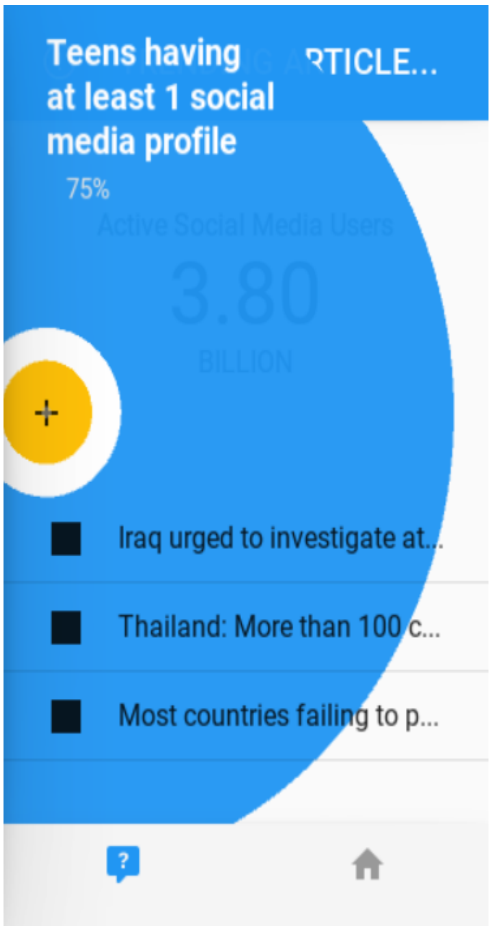

Report Portal

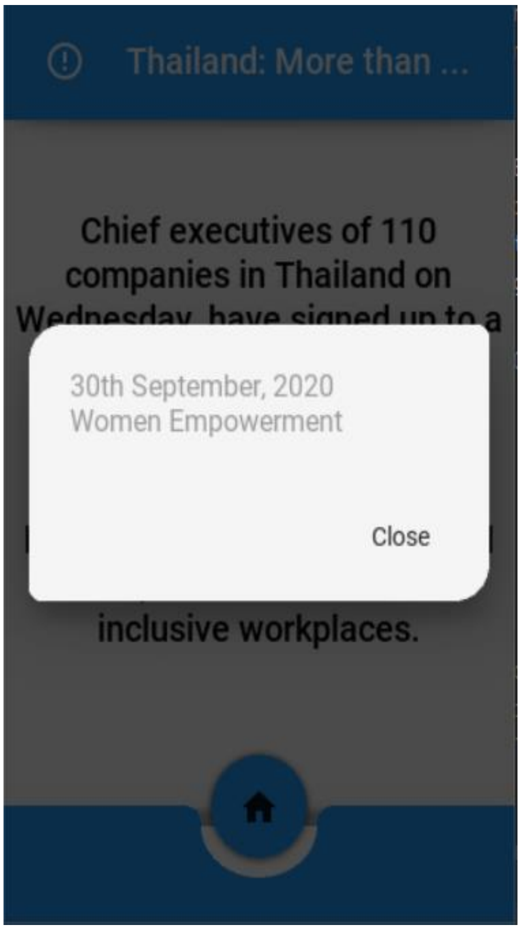
Enter abuser name
walter

Enter abusers contact number
9980765432

Enter reason
Sexually abusive messages and images.

Submit


Teens having at least 1 social media profile
75%
RTICLE...
Active Social Media Users
3.80
BILLION
Iraq urged to investigate at...
Thailand: More than 100 c...
Most countries failing to p...


Thailand: More than ...
Chief executives of 110 companies in Thailand on Wednesday, have signed up to a
inclusive workplaces.
30th September, 2020
Women Empowerment
Close