Development of Luntian: A Chatbot Companion

Eliza B. Ayo[a], Brian Paolo Calinisan, Ralph Reñier Gondra,
Michael Vince Manubay, Harold Vincent Roldan, Emmanuel Punsalan

[a] ebayo@ceu.edu.ph
Centro Escolar University, Manila, Philippines 1005

Abstract

This study focuses on designing and developing an AI chatbot application that uses a controlled dataset. The primary emphasis of the dataset lies in providing the AI with relevant information to effectively recognize user emotions and deliver suitable and precise responses. The purpose of the study is to provide companionship to the user. The technology used for the study includes Dialog flow and Android Studio. The methodology involved testing the chatbot with 50 students and employees currently enrolled or working in Centro Escolar University, who completed the Software Process and Product ISO Standards Survey for evaluation. The statistical treatment employed was random sampling. The findings indicated that the respondents generally agreed on the functionality, efficiency, and portability criteria while the respondents strongly agreed on the reliability and usability criteria of the application. The results suggest that the software was able to perform its task effectively based on the Software Process and Product ISO Standards Survey. The development of this application opens opportunities for individuals who struggle to express their emotions and demonstrates evidence of computer interaction concerning emotion.

Keywords: AI, Chatbot, Functionality, Usability, Efficiency, Portability, Application, Emotion

1. Introduction

Collecting data through scientific observation entails watching a subject behave in a given way or as they progress toward a certain behavior. It's normal to occasionally feel anxious, particularly if your life is demanding. However, excessive worry and anxiety that is uncontrollable and interferes with daily tasks may be a sign of generalized anxiety disorder according to a study from Guzelhan, Y. (2022) titled “Mental health in pre-COVID-19 and during the COVID-19 pandemic: a comparative study using symptom checklist-90- R”. Generalized anxiety disorder can affect people of any age, including children.

A study spanning 32 countries examined mental health during the COVID-19 pandemic. Results showed significant concerns: 28% had depression, 26% had anxiety, 36.5% had stress, 50% had psychological distress, 24.1% had post-traumatic stress, and 27.6% had sleep issues (Nochaiwong et al., 2021). Notably, Thailand, Pakistan, and the Philippines had the highest stress and depression scores; Thailand, Pakistan, and Malaysia topped the anxiety scores (Wang et al., 2021). Gender disparities were seen, with 10.2% of Filipino women and 7.6% of men displaying mental illness symptoms (Puyat et al., 2021). "Loneliness" marked moderate to severe depression, while the general populace cited a "lack of enjoyment" (Puyat et al., 2021).

As a result, the researchers were inspired to develop Luntian, a companion chatbot program, whose primary function is to provide a secure environment for users to express their feelings, particularly if they find it difficult to do so. The researcher's main objectives were to design and develop a machine learning algorithm to produce the chatbot, to train the chatbot using a defined dataset, and to determine the user's acceptance and adoption of LUNTIAN using Software Process and Product ISO Standards Survey. With the use of the following tools such as Dialogflow for the data set and Android studio as their programming platform, the researchers used Dialogue flow as an algorithm and NLP (Natural Language Processing) due to its speech recognition, part of speech tagging and Word sense disambiguation. This study will scholastically contribute to the future programmers that could make a mark in the innovating society.

2. Objectives of the Study

This study aimed to design and develop LUNTIAN, a chatbot that will act as a companion to a user and promote self-expression. Specifically, it has accomplished the three main objectives.

- Designed and developed a machine learning algorithm to produce the chatbot.
- Trained the chatbot using a defined dataset.
- Determined user acceptance and adoption of LUNTIAN using Software Process and Product ISO Standards Survey.

3. Related Literature and Findings

In recent years we have witnessed a disturbing rise in rates of loneliness throughout society. To face this challenge head-on researchers have proposed utilizing technology- specifically artificial intelligence- to create supportive companions for people suffering from isolation and detachment. The study authored by (Merrill Jr, et al, 2022), delves into how social presence affects the effectiveness of AI-powered companions intended as a remedy for loneliness. The authors maintain that establishing this sense is integral if we wish these technologies to make any significant impact on reducing feelings associated with solitude. Artificial intelligence (AI) has quickly developed into a component of daily life with a wide range of uses. An international technology corporation called Siemens has also been investigating the potential of AI, particularly in the creation of digital assistants that can improve human-machine connection. Siemens highlights the advantages and potential of AI digital companions in this article they contend, however, that these drawbacks are outweighed by the advantages of social presence since it can offer lonely people a special and priceless source of friendship.

In the study of Siemens (2023) entitled "Artificial Intelligence: Your Digital Companion." Siemens emphasizes the importance of virtual friends as a tool to customize interactions between people and machines. AI-powered virtual friends may help users with a variety of tasks, like navigating through complicated user interfaces or making recommendations based on their interests. The capacity of digital companions to learn from users and adjust to their demands over time, according to Siemens, makes them an efficient and effective means of boosting productivity and user experience. Siemens also talks about how digital companions could enhance well-being and healthcare. Digital companions, for instance, can be used to measure and monitor health indicators, offer medical guidance, or offer emotional support. People with chronic illnesses or those in need of individualized healthcare solutions may particularly benefit from this. The acknowledgment from Siemens regarding potential ethical challenges arising from creating AI digital companions is commendable. It is crucial to address critical issues related to privacy violations or bias while developing this technology. Transparency accountability needs to be considered during its deployment for better results.

Youth loneliness is a serious problem, with social isolation and a lack of meaningful connections being the main causes. A potential answer to this issue is the creation of companions with artificial intelligence (AI). These companions can help alleviate the causes of loneliness by offering emotional support, company, and aid with everyday duties. The creation of AI companions has the potential to lessen youth loneliness, according to this essay, which will address both their possible advantages and drawbacks. In the article by Bishop E. (2021) titled Artificial Intelligence Companions for the Young and Lonely Emily Bishop addresses the advantages of AI companions, including how they could boost mental health, lessen social isolation, and offer individualized support. Young people who might not have access to a support system can receive emotional support from AI companions, which can help to enhance their mental health. They can also foster a sense of community and companionship, which can assist in lessening social isolation. AI companions can also offer individualized help depending on a person's needs and preferences. The ethical ramifications of AI companions, particularly the possibility of abuse and the effect on human relationships, are a matter of some concern.

Furthermore, this part of the research shows that aside from literature there are also applications that are connected and used to this research. OodlesAI is an AI that helps healthcare, commercial enterprise, and different fields. Chatbots with Dialogue Flow and machine learning algorithms created via means of the Oodles era. Oodles era is attempting to enhance health diagnostics with the strength of AI. But how does this software work? Healthcare chatbots categorize sufferers throughout a set of a category of relying on how the affected person feels. It presents an automatic response that suits if not too accurately, however the closest to what the affected person is feeling with only treatment or first aid. If the affected person seeks clinical recommendation the bot might additionally determine what better medication to consume relying on the facts the affected person has given. After the whole thing is achieved and the affected person must peer at the medical doctor it additionally has an appointment characteristic in which the affected person might ask the bot approximately about his/her appointment. The healthcare enterprise is slowly reconstructing with the help of Artificial intelligence making it simpler to treat, speak and discover a method to the affected person's needs. The pandemic took a toll on us, and those sorts of modern creations might lighten the luggage we stock and assist us in view that its far constructed with text-primarily based clinical assistance, clinical consultations might now no longer be difficult as before.

In addition, one of the trending applications is Simsimi. This is an application where you get a real-time response from an AI (artificial intelligence). Simsimi is an application where most of the users are young adults. How Simsimi works, Simsimi works like any other AI chatbot but this has its unique response depending on what the user is typing and notes this one does not use frequently used lines, where users just pick out what sentence they want to use. How does the Dialog Flow algorithm work on Simsimi since Simsimi has a default response to certain greetings and its response is usually the same? Simsimi also recognizes tone when it comes to chatting and sometimes changes their response, adapting depending on the place or language used by the user. Simsimi is indeed a wonderful and fun application for creating a conversation for people who want to talk to an AI or someone who is just looking for entertainment without having anyone use your words against you. But be careful since Simsimi is created for entertainment and learning purposes but be cautious for it is also sometimes used for cyberbullying since it has a function to hide the users' names. With all these topics presented on Statistics on Mental Health, Research that shows the benefits of expressing emotions for people with mental health issues, Tools for Psychologies to diagnose mental health issues, and chat-bot applications were used as the foundation of this study. The chatbot was developed in large part thanks to relevant research in AI and chatbot design. The chatbot has been developed to consider psychological factors, align with user behaviors and preferences, make use of NLP and machine learning breakthroughs, and increase user engagement and happiness. The integration of these linked studies has made it possible to create a virtual chatbot that is smarter, more user-focused, and psychologically supportive.

4. Theoretical Framework

The Conversation Theory is a theoretical framework explaining the learning process between the living organism and machines which occurs when a conversation about a subject matter is exchanged between the two. Pask, G. (2022)

The concept is applied as a framework by having both human and chatbot converse; as the human expresses his or her emotions through the interface, the chatbot will help the human by using a scientific approach used by psychologists.

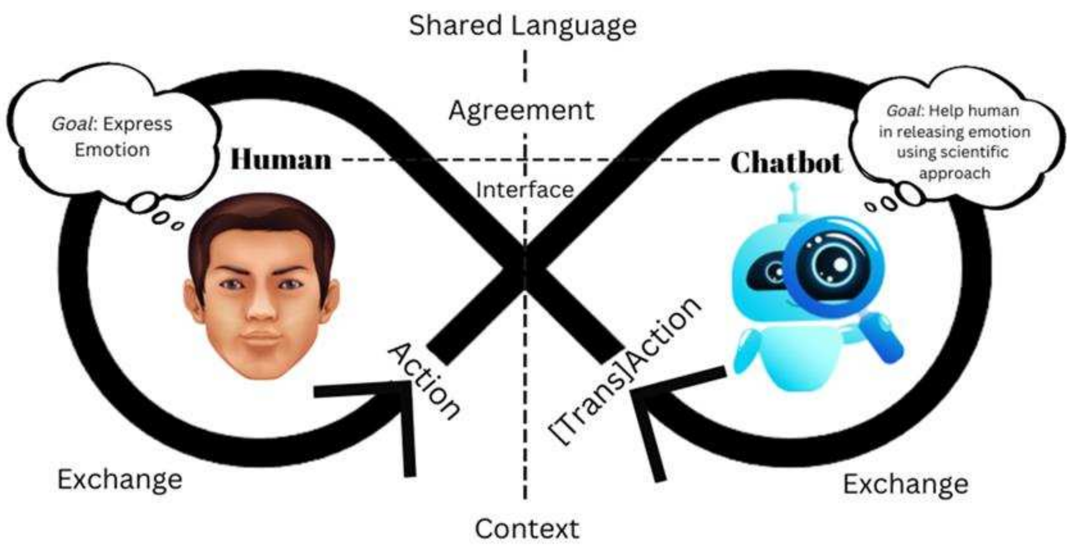


Fig. 1. Theoretical Framework

5. Research Methodology

Several software tools and platforms were needed for the software project during the development process; these were selected to suit the project's requirements. Android Studio was the Integrated Development Environment (IDE) used, and Android 13 was the operating system. The team used Dialog Flow for Natural Language Processing (NLP), Dart SDK for programming, Flutter SDK for building the user interface, Canva and Adobe Photoshop for creating the graphic design elements, and Java Development Kit (JDK) for the development process.

The software was executed on hardware setups for testing and implementation. The development system included a GEFORCE GTX 1650 GPU, an AMD Ryzen 5 3500 CPU, an AMD B450 motherboard, 16GB of 3200MHz DDR4 RAM, and 500GB of SSHD storage. Samsung Galaxy S10 with Exynos 9820 Octa-Core chipset/CPU, 8 GB of RAM, Mali-G76 MP12 GPU, and 128 GB of storage served as the testing device.

These hardware and software elements were crucial for the effective creation and operation of the software project throughout the implementation process.

6. Luntian: A Chatbot Companion – Use Case and Architecture

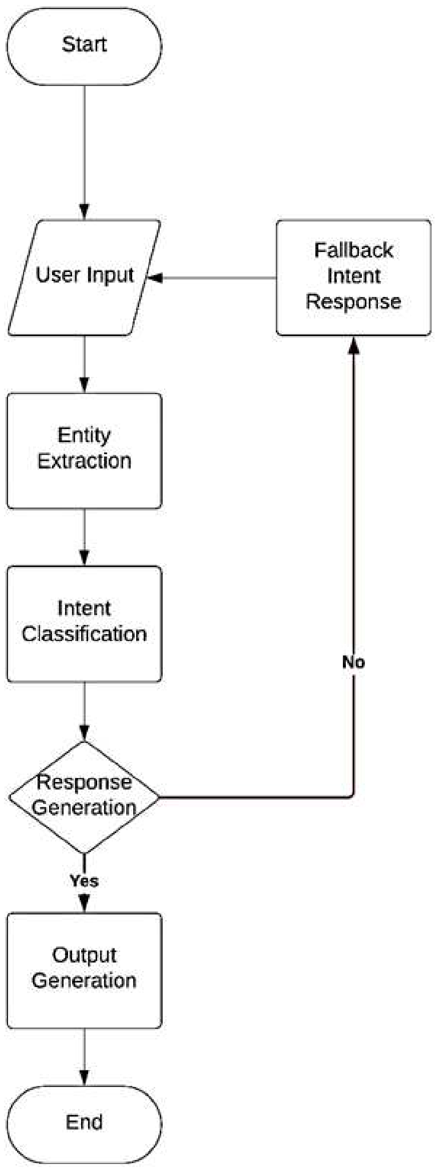


Fig. 2. Flowchart Algorithm

The thesis paper examines the backend process of a chatbot system, involving stages from user input to response generation. User input is preprocessed using NLP techniques like tokenization and stopword removal. Intent classification determines the user's intent through machine learning algorithms. Entity extraction identifies relevant entities from the input using predefined rules or machine learning models. Context and session management maintain coherent interactions. Webhook integration enables interaction with external services. The response is generated based on intent, entities, and context, either through predefined templates or machine learning models. The response is then presented to the user as text or speech. ML, NLP, and NLU techniques are integrated throughout the process, assisting in intent classification, response generation, preprocessing, and entity extraction.

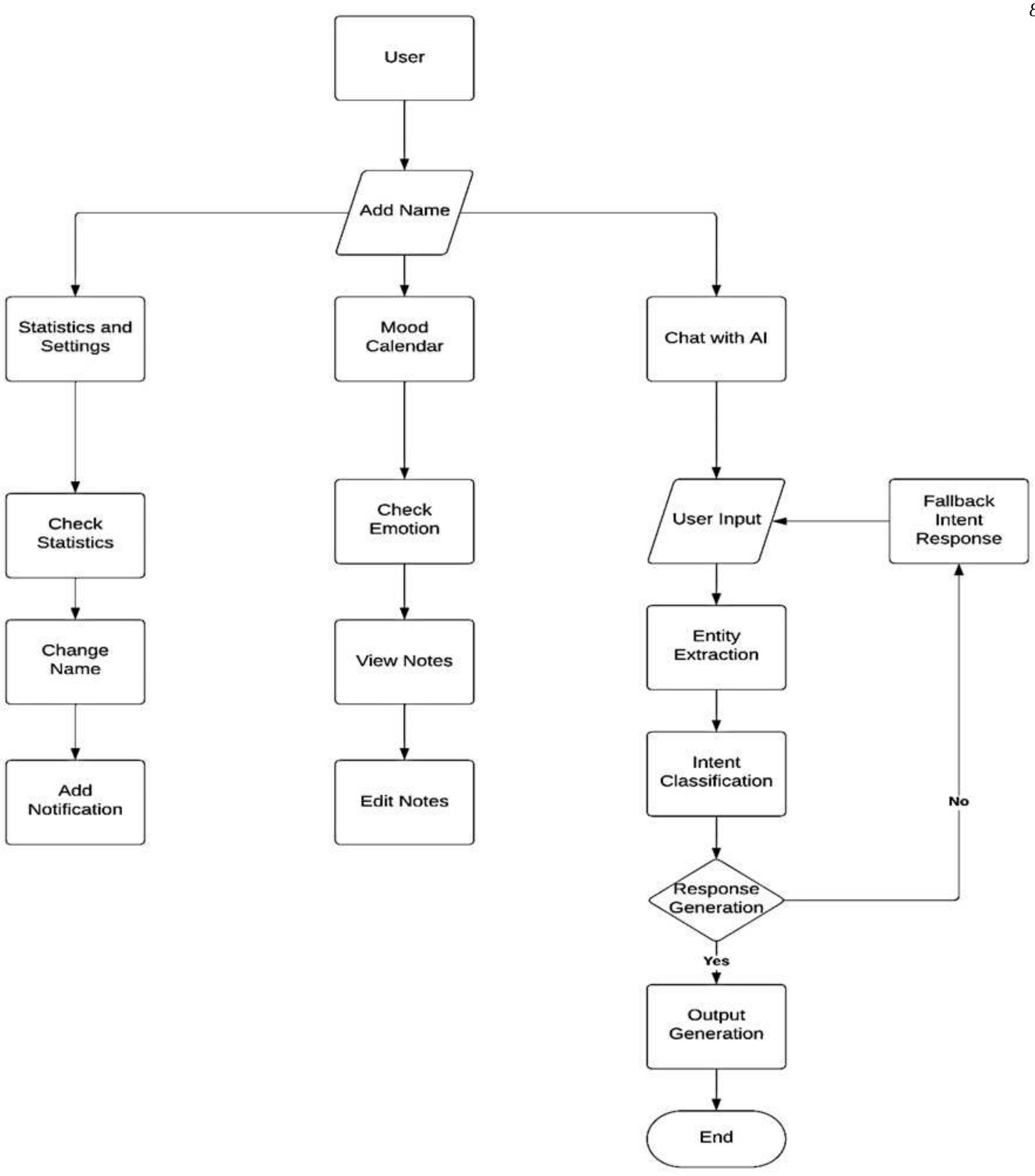


Fig. 3. User Perspective Algorithm

The user opens the app, which initiates the flow of this use case. The app then requests that the user enter their name. The user is then presented with three options. They decide to explore Statistics and Settings and interact with the AI, and the Mood Calendar. Additionally, users have three alternatives through the Mood calendar: check the day's feelings, examine the notes the user wrote that day, or change the note the user wrote that day. Finally, the user has three choices under Statistics and Settings: modify the name, establish a notification timer, and view statistics of the emotion on the calendar.

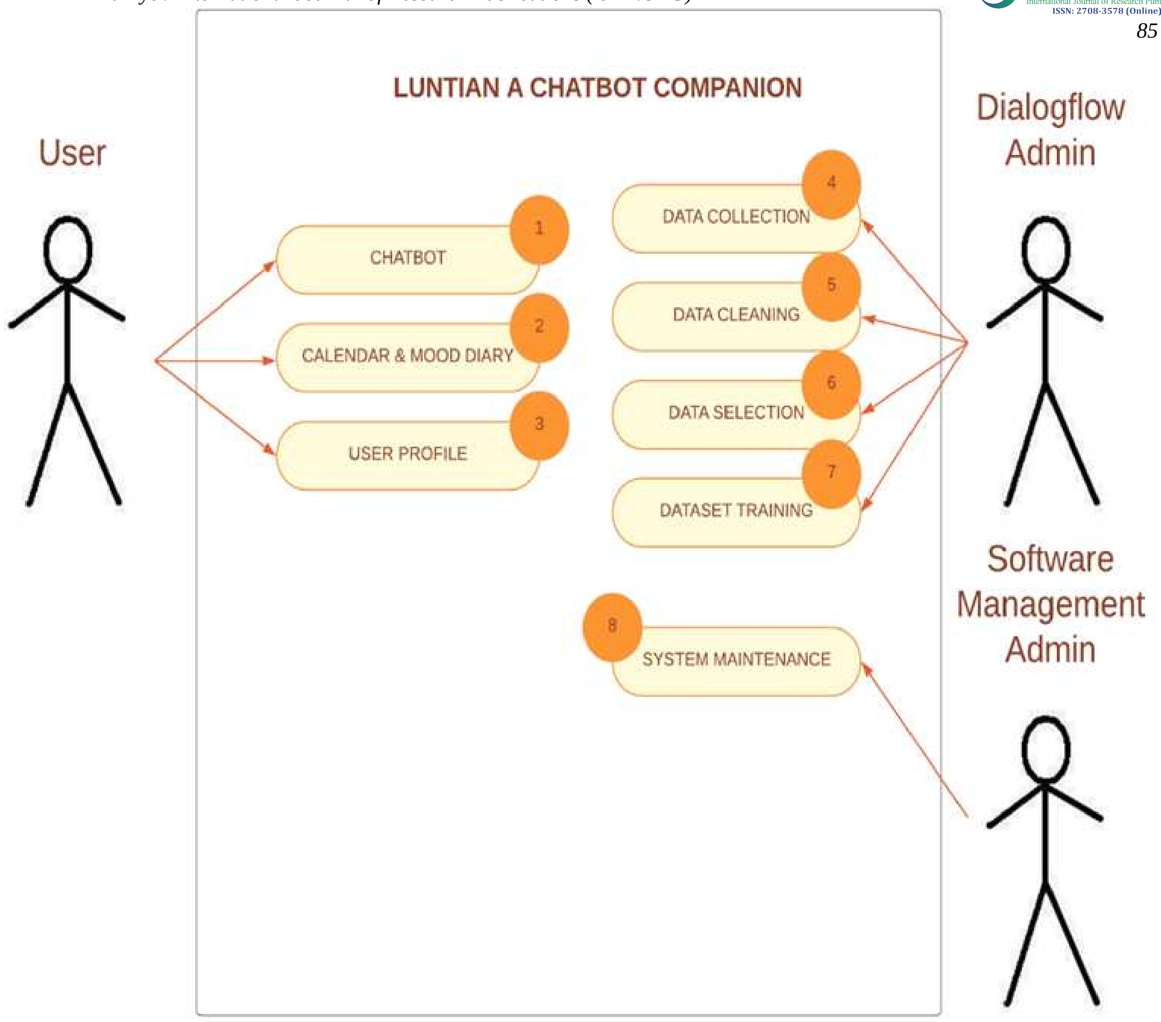


Figure 4. User Case Diagram

The Luntian use case diagram involves three main actors, as depicted in Figure 4. The user is provided with three options for selecting use cases, represented by numbers 1, 2, and 3. Option 1 enables user interaction with the chatbot utilizing machine learning, Natural Language Processing (NLP), and Natural Language Understanding (NLU) technologies. Option 2 offers an independent dashboard for managing user mood inputs. Option 3 includes a separate dashboard that relies on data from Option 2.

On the Dialogflow admin side, several processes numbered 4, 5, 6, and 7 are involved. These processes follow a sequential arrangement, starting from 4 and progressing to 7. This sequence repeats whenever the datasets are expanded and improved upon.

The Software Management Admin ensures the operational functionality of the software and conducts regular maintenance to address issues and enhance its performance.

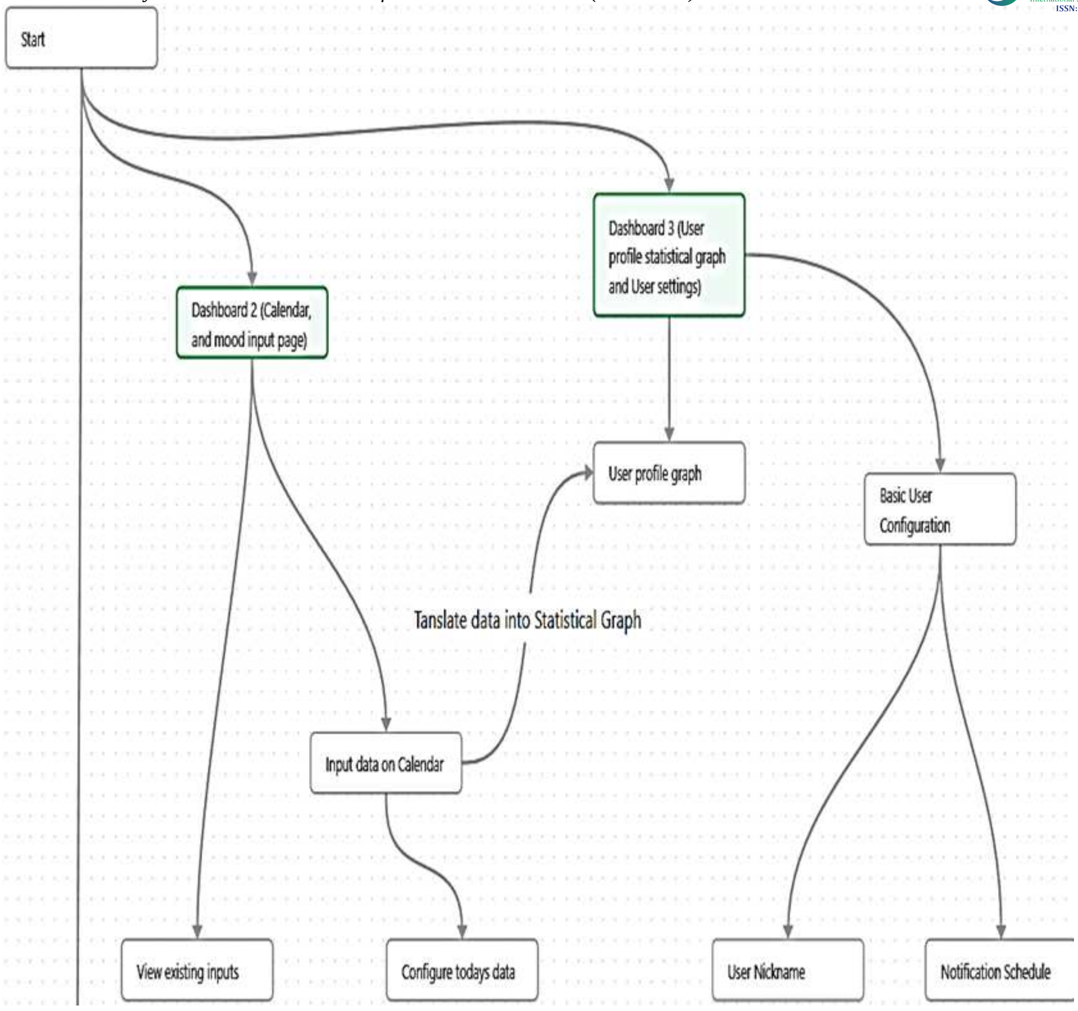


Figure 5. Application Features

The Dashboard 1 employs machine learning techniques to interpret user input and classify it into relevant groups based on a pre-trained dataset. Once the user intent is classified, Dialogflow selects an appropriate dialogue model for communication. The chatbot then executes the chosen model, completing the process, and repeats the entire cycle if new user inputs are received.

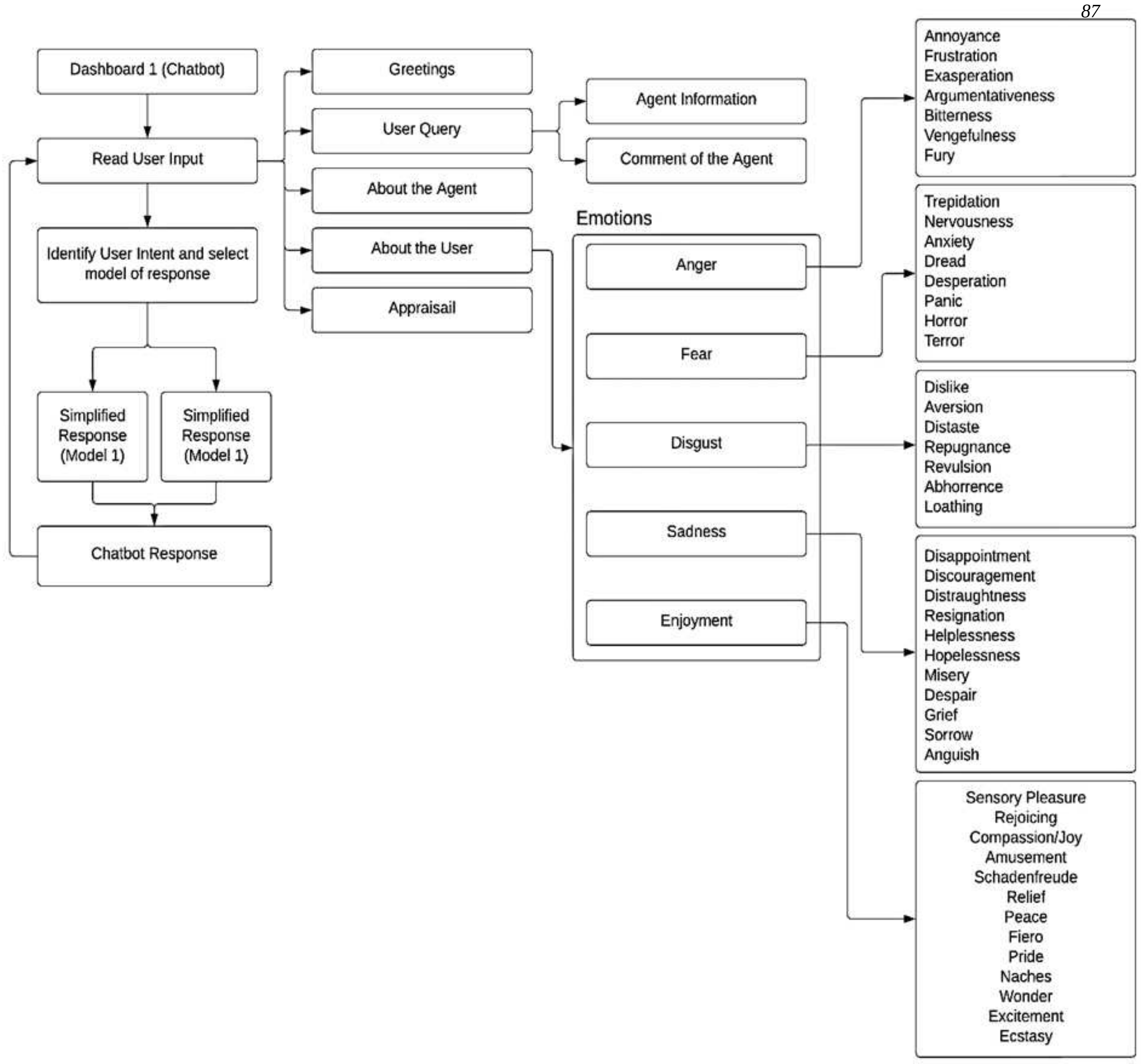


Figure 6. Chatbot Functions

The Dashboard 2 is designed to facilitate the collection of user data by providing a user-friendly interface for manual input of daily mood. Additionally, it offers the ability to access and review previous input from past dates. It is important to note that users can only input data for the current date and are unable to input data for the day prior or subsequent to the current date.

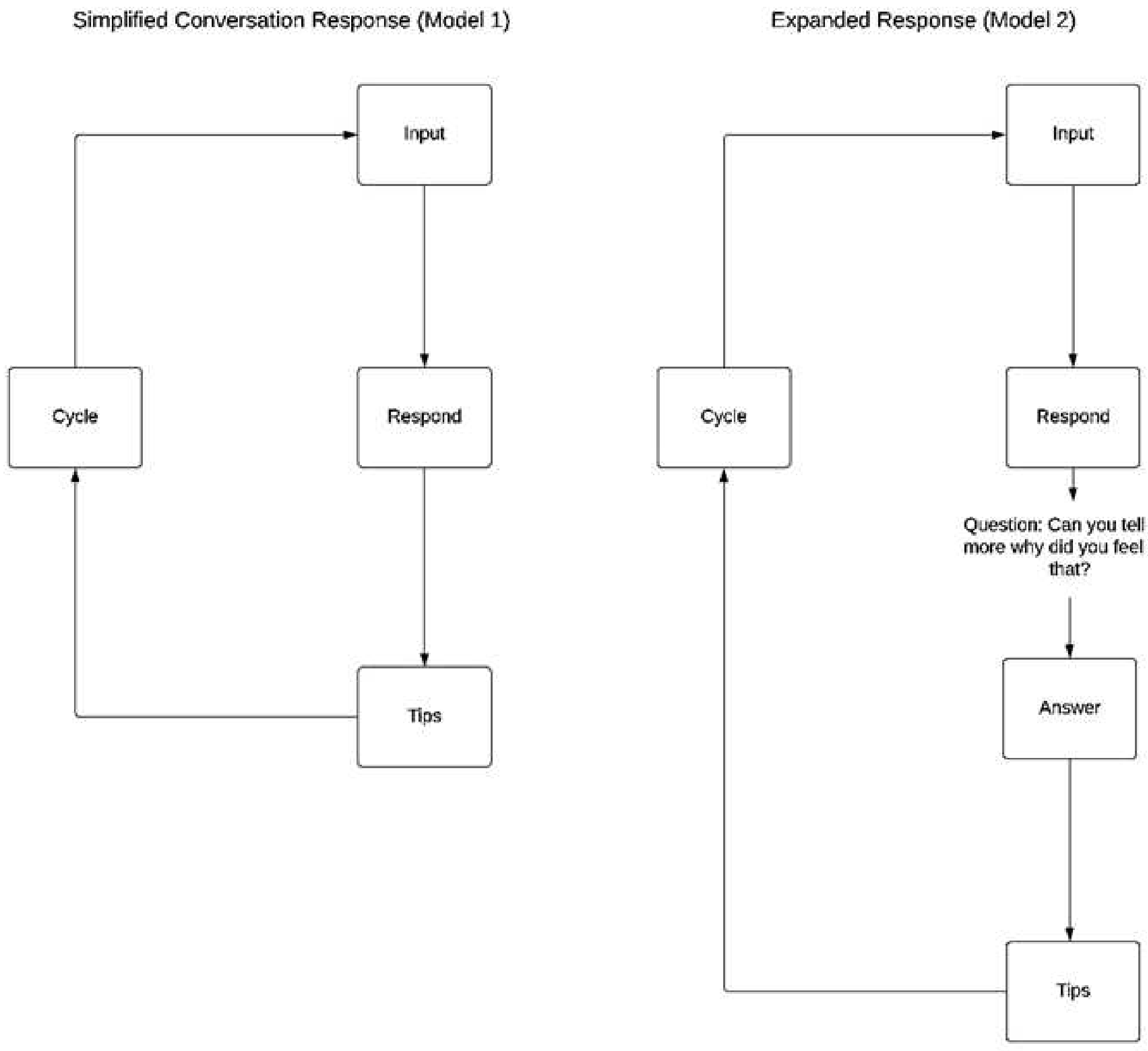

Simplified Conversation Response (Model 1)
Input
Respond
Tips
Cycle
Expanded Response (Model 2)
Input
Respond
Question: Can you tell me more why did you feel like that?
Answer
Tips
Cycle

Figure 7. Flow of Chatbot Conversation

The Dashboard 3 translates the data collected from Dashboard 2 into graphical representations and offers user settings, including options to change their nickname and configure notification schedules. Regarding communication models, Model 1 utilizes a simplified dialogue version, while Model 2 adopts an expanded dialogue version.

7. Hardware and Software Specifications

Table 1. Software Specifications

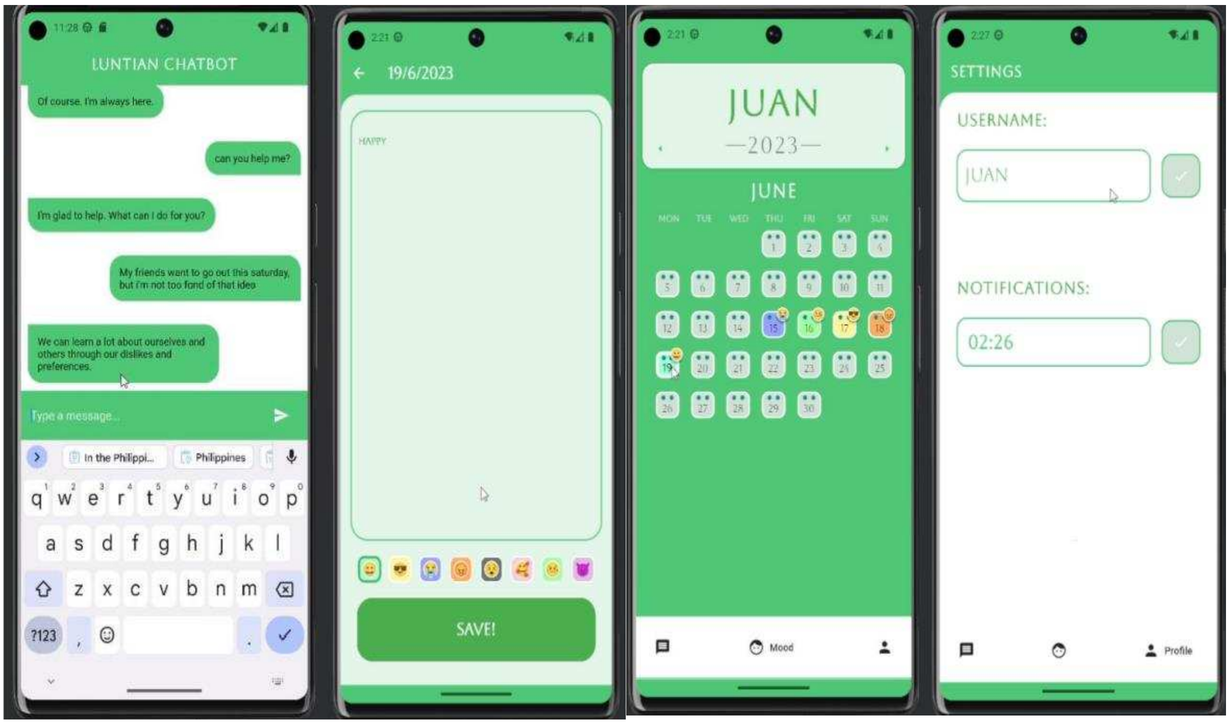
Software Requirements	Implementation
DEVELOPMENT: Operating System Android Studio Dialog Flow Dart SDK Flutter SDK Canva Adobe Photoshop Java Development Kit (JDK)	Android 13 Version (Virtual Device from Android Studio) Android 12 Version (Samsung Galaxy S10)

Table 2. Hardware Specifications

Hardware Requirements	Implementation
CPU: AMD Ryzen 5 3500 Motherboard: AMD B450 RAM: 16GB 3200mHz DDR4 GPU: GEFORCE GTX 1650 Storage: 500 GB SSD	Device: Samsung Galaxy S10 Chipset/CPU: Exynos 9820 Octa-Core RAM: 8 GB GPU: Mali-G76 MP12 Storage: 128 GB

During the development stage of Luntian (application), the researchers use a computer with specs of AMD Ryzen 5 3500, AMD B450, Ram of 16GB DDR4 3200Mhz each used for processing while the designing and storing of the application the researchers the GTX 1650 GPU they were able to create a user interface design using Adobe Photoshop and Canva and during the main development of the AI chatbot application the researchers used Android studio with JDK, Dart SDK, and Flutter SDK. With the integration of Dialog Flow data set inside the application for the implementation in an Android 12 or 13 version mobile phone that has hardware specifications of Exynos 9820 octa-core chipset, a graphics processor of Mali-G76 MP12 and a storage of 128 GB.

8. User Interface Design



Of course. I'm always here.

can you help me?

I'm glad to help. What can I do for you?

My friends want to go out this saturday, but i'm not too fond of that idea

We can learn a lot about ourselves and others through our dislikes and preferences.

Figure 8. User Interface Design

The UI of the application consists (1) the Chat interface where you can chat with Luntian (2) is the calendar feature where the user can set their mood for that day and the user could check why they are feeling that certain mood for that day (3) Third is the diary part of that day wherein the user could input why they are feeling that way so they could go back to that day to know whether why did they feel that way. (4) Is where the Users Profile is set and the notification for that certain time to input the user's feelings.

9. Codes

The following lines are codes for Luntian’s **Message View**:

```
import 'package:luntian/Model/emotion.dart';
import 'package:luntian/Model/statsmodel.dart';
import 'package:luntian/View/colors.dart';
import 'package:luntian/main.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:syncfusion_flutter_charts/charts.dart';

class StatsView extends StatelessWidget {
  const StatsView({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final myStats = Stats(context);
    final chartData = [
      ChartData(Emotion.happy.getEmoji(), myStats.getHappyDays()),
      ChartData(Emotion.calm.getEmoji(), myStats.getCalmDays()),
      ChartData(Emotion.crying.getEmoji(), myStats.getCryingDays()),
      ChartData(Emotion.angry.getEmoji(), myStats.getAngryDays()),
      ChartData(Emotion.bad.getEmoji(), myStats.getBadDays()),
      ChartData(Emotion.loved.getEmoji(), myStats.getLovedDays()),
      ChartData(Emotion.sick.getEmoji(), myStats.getSickDays()),
      ChartData(Emotion.devil.getEmoji(), myStats.getDevilDays()),
    ];

    return Scaffold(
      backgroundColor: myBackgroundColor,
      appBar: AppBar(
        elevation: 0,
        title: Text(
          systemLocales.first.toString() == "es_ES" ? "Configuración" : "Stats",
          style: GoogleFonts.aboreto(
            textStyle: const TextStyle(
              fontWeight: FontWeight.bold,
              letterSpacing: 1,
            ),
          ),
        ),
        backgroundColor: myBackgroundColor,
        automaticallyImplyLeading: false,
      ),
      body: Container(
        height: maxheight,
        width: maxwidth,
        decoration: const BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.only(topLeft: Radius.circular(25)),
        ),
        child: Center(
          child: Padding(
            padding: const EdgeInsets.symmetric(vertical: 20.0, horizontal: 10),
            child: Container(
              padding: const EdgeInsets.only(right: 20),
              decoration: const BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.all(Radius.circular(20)),
            ),
            child: SfCartesianChart(
              title: ChartTitle(
                text: myStats.getYear(),
                textStyle: GoogleFonts.aboreto(
                  textStyle: const TextStyle(
                    color: Colors.green,
                    fontWeight: FontWeight.bold,
                    fontSize: 20,
                  ),
                ),
              ),
              plotAreaBorderColor: Colors.white,
              primaryXAxis: CategoryAxis(
                isInversed: true,
                axisLine: const AxisLine(color: myBackgroundColor),
                majorTickLines: const MajorTickLines(width: 0),
                majorGridLines: const MajorGridLines(width: 0),
              ),
              primaryYAxis: NumericAxis(
                isVisible: true,
                axisLine: const AxisLine(color: myBackgroundColor),
                majorTickLines: const MajorTickLines(width: 0),
                interval: 1,
                majorGridLines: const MajorGridLines(width: 0),
              ),
              series: <ChartSeries>[
                BarSeries<ChartData, String>(
                  onCreateRenderer: (ChartSeries<ChartData, String> series) =>
                      _CustomColumnSeriesRenderer(),
                  gradient: const LinearGradient(
                    begin: Alignment.topRight,
                    end: Alignment.bottomLeft,
                    colors: [myBackgroundColor, Colors.green],
                  ),
                  color: myBackgroundColor,
                  width: 0.2,
                  borderRadius: const BorderRadius.only(
                    topRight: Radius.circular(5),
                    bottomRight: Radius.circular(5),
                  ),
                  dataSource: chartData,
                  xValueMapper: (ChartData data, _) => data.position,
                  yValueMapper: (ChartData data, _) => data.value,
                ),
              ],
            ),
          ),
        ),
      ),
    ),
  );
 }
}

class ChartData {
  String position;
  int value;
  ChartData(this.position, this.value);
}

class _CustomColumnSeriesRenderer extends BarSeriesRenderer {
  _CustomColumnSeriesRenderer();

  @override
  BarSegment createSegment() => _ColumnCustomPainter();
}

class _ColumnCustomPainter extends BarSegment {
  final colorList = [
    happyColor,
    calmColor,
    cryingColor,
    angryColor,
    badColor,
    lovedColor,
    sickColor,
    devilColor,
  ];

  @override
  Paint getFillPaint() {
    final customerFillPaint = Paint()
      ..isAntiAlias = false
      ..color = colorList[currentSegmentIndex!]
      ..style = PaintingStyle.fill;
    return customerFillPaint;
  }
}
```

The following lines are codes for Luntian's **Calendar View**:

```
import 'package:luntian/Controller/controller.dart';
import 'package:luntian/Model/dayyearcalculator.dart';
import 'package:luntian/View/colors.dart';
import 'package:luntian/View/dayview.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:intl/intl.dart';
import 'package:scrollable_positioned_list/scrollable_positioned_list.dart';

import '../main.dart';

class CalendarView extends StatefulWidget {
  final Color backgroundColor, dotsColor, monthColor;
  final double calendarMinimizedHeight;
  final ItemScrollController itemScrollController = ItemScrollController();
  final ItemPositionsListener itemPositionsListener =
      ItemPositionsListener.create();

  CalendarView(this.backgroundColor, this.dotsColor, this.monthColor,
      this.calendarMinimizedHeight, {Key? key}) : super(key: key);

  @override
  State<CalendarView> createState() => _CalendarViewState();
}

class _CalendarViewState extends State<CalendarView> {
  Box? box;

  @override
  void initState() {
    super.initState();
    box = Hive.box(DateTime.now().year.toString());
  }

  int weekPosition(DateTime d) {
    switch (DateFormat('EEEE').format(d)) {
      case "Monday":
        return 0;
      case "Tuesday":
        return 1;
      case "Wednesday":
        return 2;
      case "Thursday":
        return 3;
      case "Friday":
        return 4;
      case "Saturday":
        return 5;
      case "Sunday":
        return 6;
      default:
        return 0;
  }
}

Widget myWeekDays(double maxwidth) {
  final dayNames = [
    "Lun",
    "Mar",
    "Mie",
    "Jue",
    "Vie",
    "Sab",
    "Dom",
  ];
  return SizedBox(
    width: maxwidth - maxwidth * 0.14,
    child: Row(
      children: dayNames.map((dayName) => SizedBox(
        width: maxwidth / 11,
        child: Text(
          systemLocales.first.toString() == "es_ES" ? dayName : dayName.substring(0, 3),
          textAlign: TextAlign.center,
          style: GoogleFonts.aboreto(
            textStyle: const TextStyle(
              color: containerColor,
              fontWeight: FontWeight.bold,
              fontSize: 12,
            ),
          ),
        ),
      )).toList(),
    ),
  );
}

Widget myGrid(int monthDays, int lastDays, double maxwidth, double maxheight, monthNumber) {
  CalendarController controller = CalendarController();
  DateTime firstDay = DateTime(int.parse(controller.getRenderedYear(context)), monthNumber, 1);
  int wPosition = weekPosition(firstDay);

  return SizedBox(
    width: maxwidth,
    height: maxheight,
    child: ValueListenableBuilder(
      valueListenable: box!.listenable(),
      builder: (context, box, child) {
        return GridView.count(
          crossAxisCount: 7,
          crossAxisSpacing: 0,
          mainAxisSpacing: 0,
          physics: const NeverScrollableScrollPhysics(),
          childAspectRatio: 1 / 1,
          shrinkWrap: true,
          children: List.generate(
            monthDays + wPosition,
            (index) => index < wPosition
```

```
                ? const SizedBox()
                : Center(
                    child: box.get(index - wPosition + lastDays + 1) != null
                        ? DayView(
                            box.get(index - wPosition + 1 + lastDays),
                            widget.calendarMinimizedHeight,
                            widget.dotsColor)
                        : Container(
                            width: 35,
                            height: 35,
                            decoration: BoxDecoration(
                              border: Border.all(color: Colors.white, width: 2),
                              color: containerColor,
                              borderRadius: BorderRadius.circular(10),
                            ),
                            child: Column(
                              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                              children: [
                                Row(
                                  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                                  children: [
                                    Container(
                                      height: 6,
                                      width: 6,
                                      decoration: BoxDecoration(
                                        color: widget.dotsColor,
                                        borderRadius: BorderRadius.circular(20),
                                      ),
                                    ),
                                    Container(
                                      height: 6,
                                      width: 6,
                                      decoration: BoxDecoration(
                                        color: widget.dotsColor,
                                        borderRadius: BorderRadius.circular(20),
                                      ),
                                    ),
                                  ],
                                ),
                                Text(
                                  (index - wPosition + 1).toString(),
                                  style: GoogleFonts.aboreto(),
                                ),
                              ],
                            ),
                          ),
                  ),
            ),
          );
        },
      ),
    );
}

@override
```

```
Widget build(BuildContext context) {
  CalendarController controller = CalendarController();
  box = Hive.box(controller.getRenderedYear(context));
  bool isLeapYear = DayYearCalculator(DateTime.now()).isLeapYear();

  var maxwidth = (MediaQuery.of(context).size.width);
  var maxheight = (MediaQuery.of(context).size.height);

  final monthNames = [
    "Enero",
    "Febrero",
    "Marzo",
    "Abril",
    "Mayo",
    "Junio",
    "Julio",
    "Agosto",
    "Septiembre",
    "Octubre",
    "Noviembre",
    "Diciembre",
  ];

  List<Widget> items = List.generate(
    monthNames.length,
    (monthIndex) => Column(
      children: [
        const SizedBox(height: 15),
        Center(
          child: Text(
            systemLocales.first.toString() == "es_ES"
                ? monthNames[monthIndex]
                : DateFormat('MMMM').format(DateTime(0, monthIndex + 1)),
            style: GoogleFonts.aboreto(
              textStyle: const TextStyle(
                fontWeight: FontWeight.bold,
                color: Colors.white,
                fontSize: 25,
                letterSpacing: 2,
              ),
            ),
          ),
        ),
        const SizedBox(height: 15),
        myWeekDays(maxwidth),
        myGrid(
          DateTime(0, monthIndex + 1, 0).day,
          List<int>.generate(monthIndex, (i) => DateTime(0, i + 1, 0).day).reduce((a, b) => a + b),
          maxwidth * 0.9,
          maxheight <= 600 ? maxheight * 0.5 : maxheight * 0.4,
          monthIndex + 1,
        ),
      ],
    ),
  );
```

```
      int currentMonth = controller.getCurrentDate(context).getMonth();

      return Container(
        clipBehavior: Clip.hardEdge,
        width: maxwidth,
        height: maxheight <= 600 ? maxheight * 0.7 : maxheight * 0.50,
        decoration: const BoxDecoration(
          color: Colors.transparent,
        ),
        child: ScrollablePositionedList.builder(
          initialScrollIndex: (currentMonth - 1) * 2,
          scrollDirection: Axis.horizontal,
          itemCount: items.length,
          itemBuilder: (context, index) => items[index],
          itemScrollController: widget.itemScrollController,
          itemPositionsListener: widget.itemPositionsListener,
        ),
      );
    }
}
```

The following lines are codes for Luntian’s **Statistics View**:

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:syncfusion_flutter_charts/charts.dart';

import '../main.dart';
import 'colors.dart';
import '../Model/emotion.dart';
import '../Model/statsmodel.dart';

class StatsView extends StatelessWidget {
  const StatsView({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Stats myStats = Stats(context);
    final List<ChartData> chartData = [
      ChartData(Emotion.happy.getEmoji(), myStats.getHappyDays()),
      ChartData(Emotion.calm.getEmoji(), myStats.getCalmDays()),
      ChartData(Emotion.crying.getEmoji(), myStats.getCryingDays()),
      ChartData(Emotion.angry.getEmoji(), myStats.getAngryDays()),
      ChartData(Emotion.bad.getEmoji(), myStats.getBadDays()),
      ChartData(Emotion.loved.getEmoji(), myStats.getLovedDays()),
      ChartData(Emotion.sick.getEmoji(), myStats.getSickDays()),
      ChartData(Emotion.devil.getEmoji(), myStats.getDevilDays()),
    ];
    return Scaffold(
      backgroundColor: myBackgroundColor,
      appBar: AppBar(
        elevation: 0,
        title: Text(
          systemLocales.first.toString() == "es_ES" ? "Configuración" : "Stats",
          style: GoogleFonts.aboreto(
            textStyle: const TextStyle(
              fontWeight: FontWeight.bold,
              letterSpacing: 1,
            ),
          ),
        ),
        backgroundColor: myBackgroundColor,
        automaticallyImplyLeading: false,  ),
      body: Container(
        height: MediaQuery.of(context).size.height,
        width: MediaQuery.of(context).size.width,
        decoration: const BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.only(topLeft: Radius.circular(25)),),
        child: Center(
          child: Padding(
            padding: const EdgeInsets.symmetric(vertical: 20.0, horizontal: 10),
            child: Container(
              padding: const EdgeInsets.only(right: 20),
              decoration: const BoxDecoration(
                color: Colors.white,
            borderRadius: BorderRadius.all(Radius.circular(20)),
          ),
          child: SfCartesianChart(
            title: ChartTitle(
              text: myStats.getYear(),
              textStyle: GoogleFonts.aboreto(
                textStyle: const TextStyle(
                  color: Colors.green,
                  fontWeight: FontWeight.bold,
                  fontSize: 20,
                ),
              ),
            ),
            plotAreaBorderColor: Colors.white,
            primaryXAxis: CategoryAxis(
              isInversed: true,
              axisLine: const AxisLine(color: myBackgroundColor),
              majorTickLines: const MajorTickLines(width: 0),
              majorGridLines: const MajorGridLines(width: 0),
            ),
            primaryYAxis: NumericAxis(
              isVisible: true,
              axisLine: const AxisLine(color: myBackgroundColor),
              majorTickLines: const MajorTickLines(width: 0),
              interval: 1,
              majorGridLines: const MajorGridLines(width: 0),
            ),
            series: <ChartSeries>[
              BarSeries<ChartData, String>(
                onCreateRenderer: (ChartSeries<ChartData, String> series) =>
                    _CustomColumnSeriesRenderer(),
                gradient: const LinearGradient(
                  begin: Alignment.topRight,
                  end: Alignment.bottomLeft,
                  colors: [myBackgroundColor, Colors.green],
                ),
                color: myBackgroundColor,
                width: 0.2,
                borderRadius: const BorderRadius.only(
                  topRight: Radius.circular(5),
                  bottomRight: Radius.circular(5),
                ),
                dataSource: chartData,
                xValueMapper: (ChartData data, _) => data.position,
                yValueMapper: (ChartData data, _) => data.value,
              ),   ],
          ),
        ),
      ),
    ),
  ),
 ); }}
class ChartData {
  String position;
  int value;
```

```
    ChartData(this.position, this.value);
  }
  class _CustomColumnSeriesRenderer extends BarSeriesRenderer {
    _CustomColumnSeriesRenderer();
    @override
    BarSegment createSegment() {
      return _ColumnCustomPainter();
    }
  }
  class _ColumnCustomPainter extends BarSegment {
    final colorList = [
      happyColor,
      calmColor,
      cryingColor,
      angryColor,
      badColor,
      lovedColor,
      sickColor,
      devilColor, ];
    @override
    Paint getFillPaint() {
      final Paint customerFillPaint = Paint();
      customerFillPaint.isAntiAlias = false;
      customerFillPaint.color = colorList[currentSegmentIndex!];
      customerFillPaint.style = PaintingStyle.fill;
      return customerFillPaint;
}}
```

During the development of the Luntian interface, the researchers carefully partitioned the dashboard into three distinct components, each serving a specific purpose. Dashboard One was designed to facilitate seamless user interactions with the chatbot, providing a platform for meaningful conversations. Dashboard two was dedicated to empowering users with the ability to input data into a calendar system, edit and manage their entries, and access previous data inputs. Dashboard three focused on providing visualizations and summaries of user data, as well as offering user profile configuration options.

These functionalities were purposefully designed to establish Luntian as a chatbot companion that promotes self-expression and encourages users to freely communicate. Moreover, they were strategically aligned with the study's objectives, ensuring that Luntian's features and capabilities directly contribute to the research goal.

10. Results

Table 3. Functionality

Functionality	Mean	SD	Implementation
CRITERIA:			
1. Can the Chatbot perform the task assigned?	2.08	1.065	Agree
2. I expect a companion Chatbot to help me express my emotions.	2.10	0.994	Agree
3. Is the software equipped with acceptable security measures?	2.16	1.090	Agree
4. Does the Chatbot meet existing requirements?	2.06	1.076	Agree

In the Functionality test, the respondents agree that (1) the chatbot can perform the task assigned having a 2.08 mean of having an "agree" interpretation,(2) the chatbot helped them in expressing their emotion having a 2.1 mean of "agree" interpretation,(3) that the software is equipped with acceptable security measures and lastly received 2.16 mean and an "agree" interpretation (4) that the chatbot meets its existing requirements having received the lowest mean of 2.06 and an "agree" interpretation. With an overall interpretation of "agree" the respondents agree about the software's functionality.

Table 4. Reliability

Reliability	Mean	SD	Implementation
CRITERIA:			
1. Can most faults be eliminated overtime?	1.98	0.9365	Agree
2. Can the software handle errors?	2.02	0.9791	Agree
3. Can the software resume work and restore data?	2.00	0.9035	Agree
4. Does the software meet existing reliability standards?	1.76	1.1250	Agree

While the responses of the respondents when it comes to the reliability of the software. For the first question (1) can most faults be eliminated over the responses generated a mean of 1.98 which is which resulted in an "agree" interpretation and (2) can the software handle errors received a mean of 2.02 which is an "agree" interpretation and highest mean in this table. (3) Can the software resume work and restore data receive a 2.0 mean which is interpreted as an "agree" and (4) does the software meet the existing reliability standards having the lowest mean and interpreted as "strongly agree" The responses show that most of the respondents agreed about the reliability of the software except for question 4 which resulted in strongly agree.

Table 5. Usability

Reliability	Mean	SD	Implementation
CRITERIA:			
1. Can the software be understood easily?	1.72	1.0100	Strongly Agree
2. Can the software be learned easily?	1.76	1.0606	Strongly Agree
3. Can the software be created with minimal effort?	2.28	1.0542	Agree
4. Is the interface of the software appealing?	1.82	1.0577	Agree
5. Does the software meet existing usability standards?	2.06		Agree

The data gathered about the usability of the software. Question (1) is Can the software be understood easily resulting in a 1.72 mean receiving the lowest in the usability category which is interpreted to “strongly agree” (2) Can the software can be easily learned received a 1.72 mean interpreted to “strongly agree”. (3) Can the software be created with minimal effort and receive the highest mean 2.28 mean which is interpreted to “agree”? (4) Does the interface of the software appealing receive a mean of 1.82 and an interpretation of “agree” and last (5) Does the software meet existing usability standards get a 2.06 mean which is interpreted to “agree”.

Table 6. Efficiency

Efficiency	Mean	SD	Implementation
CRITERIA:			
1. Does the software behave promptly?	2.00	1.0301	Agree
2. Does the software meet existing efficiency standards?	1.98	1.0971	Agree

The Efficiency data gathered is mixed of both agree and strongly agree meaning the respondents strongly agree for questions 1 and 2 in the usability and agree for questions 3 to 5. While the responses of the efficiency category of the software. In question (1) does the software behave promptly and receive a mean of 2.0 which transcribes to “agree”? Question two (2) Does the software meet the existing efficiency standard and receive a 1.98 mean which is interpreted as “agree”? The results for the efficiency table are all “agree” which means that the respondents agree on the efficiency of the application.

Table 7. Portability

Reliability	Mean	SD	Implementation
CRITERIA:			
1. Can the software be adapted easily?	2.06	1.1241	Agree
2. Can the software be installed easily?	2.10	1.0350	Agree
3. Can the software work with existing software systems?	1.94	1.0768	Agree
4. Can the software be replaced with a similar product?	2.12	1.0999	Agree
5. Does the software meet existing portability standards?	2.00	1.0301	Agree

The portability of the criteria is about the portability of the software. For question (1) can the software be adapted easily? Received a mean of 2.06 which is interpreted to “agree”. (2) Can the software be installed easily? Received a mean of 2.1 which is interpreted to “agree”. (3) Can the software work with existing software systems? Received a mean of 1.94 and an “agree” interpretation. (4) Can the software be replaced with a similar product? Received a mean of 1.94 and an “agree interpretation. Lastly (5) does the software meet existing portability standards received a 2.0 mean and was interpreted as “agree”. For Portability, the interpretations of the questions are all agreed which means that all the respondents agree with the portability feature of the application.

11. Conclusion

Based on the indicated findings, the following conclusions were drawn: The respondents indicated on the question that they “agreed” to the functionality, efficiency, and portability criteria of the chatbot application. While Reliability criteria, and Usability criteria, indicate that students answered "strongly agree" on some questions stated. The result of the table implies the software was able to perform its task regarding functionality, reliability, usability, efficiency, and portability. With this, the researchers are successful in doing their objectives of the study which are to:

- Designed and developed the machine learning algorithm to produce the chatbot,
- Trained the chatbot using a defined data set, and
- Determined user acceptance and adoption of LUNTIAN using the Software Process and Product ISO Standards Survey

Using machine learning and natural language processing, the researchers were able to apply and develop the algorithm to generate responses and interact with users successfully to accomplish the first goal. By using a defined data set to train the chatbot, the second goal was achieved. To teach the algorithm patterns, discern intentions, and provide acceptable answers, a sizable amount of labeled data has been provided. The researchers used the Software Process and Product ISO Standards Survey to ascertain user approval and adoption of LUNTIAN. This survey is a trusted and established technique for assessing user acceptability and satisfaction with software programs.

12. Recommendations

To further improve the chatbot application, however, there is room for further work and enhancements. Some potential areas for investigation include:

- Expansion of the dataset of the AI for more accurate conversation-friendly applications.
- Enhancement of the dialogue conversation flow pattern to allow more nonlinear user and chatbot conversation patterns.
- Evaluation using a larger group to determine if the same findings will be established.

References

Al-Qutaish, R., & Al-Sarayreh, K. (2005). Software Process and Product ISO Standards: A Comprehensive Survey. Retrieved from https://www.researchgate.net/publication/237104465_Software_Process_and_Product_ISO_Standards_A_Comprehensive_Survey/citation/download

Anjara, S. G. (2020, August 10). Using the GHQ-12 to screen for mental health problems among primary care patients: psychometrics and practical considerations - International Journal of Mental Health Systems. International Journal of Mental Health Systems, 14(1), 45. https://doi.org/10.1186/s13033-020- 00397-0

Areàn, P., Ly, K. H., & Andersson, G. (2022). Mobile technology for mental health assessment. Current Directions in Psychological Science, 18(2), 115-119. https://doi.org/10.31887/DCNS.2016.18.2/parean

Arnrich, B., Osmani, V., & Bardram, J. E. (2011). Mental health and the Impact of ubiquitous technologies. In Proceedings of the 13th International Conference on Ubiquitous Computing (Ubicomp '11) (pp. 317-326). ACM. https://doi.org/10.1145/2030112.2030151

Baikie, K. (2018, January). Emotional and physical health benefits of expressive writing. Advances in Psychiatric Treatment, 24(2), 216-224. https://doi.org/10.1192/apt.bp.107.004030

Bishop, E. (2021, July 6). Artificial Intelligence Companions for the Young and Lonely. DataDrivenInvestor. Retrieved from https://medium.datadriveninvestor.com/artificial-intelligence-companions-for-theyoung-and-lonely-13ab12916354

Bizzaco, M. (2021, March 30). What is Amazon's Alexa, and what can it do? Digital Trends. Retrieved from https://www.digitaltrends.com/home/what-is-amazonsalexa-and-what-can-it-do/

Boydell, K., Hodgins, M., Pignatiello, A., & Teshima, J. (2014). Using Technology to Deliver Mental Health Services to Children and Youth: A Scoping Review. Journal of the Canadian Academy of Child and Adolescent Psychiatry, 23(2), 87- 99. Retrieved from: https://pubmed.ncbi.nlm.nih.gov/24872824

Choudhury, N., & Islam, A. (2020). Mobile apps for Mental Health: a content analysis. In Proceedings of the 9th International Conference on Software and Information Engineering (ICSIE '20) (pp. 19-23). IEEE. https://doi.org/10.1109/ICSIE48614.2020.00007

Conversation theory (Gordon Pask). (n.d.). InstructionalDesign.org. Retrieved October 7, 2022, from https://www.instructionaldesign.org/theories/conversationtheory/

de Alva, F. E. M., Wadley, G., & Lederman, R. (2015, December 7). It feels different from real life: Users' Opinions of Mobile Applications for Mental Health. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15) (pp. 467-476). ACM. https://doi.org/10.1145/2702123.2702195

Doherty, G., Coyle, D., & Matthews, M. (2010). Design and evaluation guidelines for mental health technologies. International Journal of Human-Computer Studies, 68(4), 223-236. https://doi.org/10.1016/j.ijhcs.2009.10.006

Dudkin, I. (2019, June 4). How Does Siri Work: Technology and Algorithm. Skywell Software. Retrieved from: https://skywell.software/blog/how-does-siri-worktechnology-and-algorithm/

Elokence. (2017). Everything you've always wanted to know about Akinator. Retrieved September 10, 2022, from https://en.akinator.com/content/6/everythingyou-ve-always-wanted-to-know-about-akinator

Explained: What is SimSimi? (n.d.). www.webwise.ie. Retrieved September 11, 2022, From: https://www.webwise.ie/parents/explained-what-is-simsimi/

Fedewa, J. (2021, January 21). What Is Google Assistant, and What Can It Do? www.howtogeek.com. Retrieved September 10, 2022, From: https://www.howtogeek.com/692895/what-is-google-assistant-and-what-can-itdo/

Fowler, G. (2022, March 16). A digital companion for the elderly: Does it solve loneliness or make it worse? The Washington Post. Retrieved from https://www.washingtonpost.com/technology/2022/03/16/lonely-elderlycompanion-ai-device/

independentleft. (2021, April 15). The social impact of artificial intelligence. Independentleft.ie. Retrieved March 28, 2023, from https://independentleft.ie/aiand-society/

Jainish Patel, Prittesh Patel (2019). Consequences of Repression of Emotion: Physical Health, Mental Health and General Well Being. International Journal of Psychotherapy Practice and Research, 1(3), 16-21. https://openaccesspub.org/ijpr/article/999

Joseph H. Puyat, Ma. Cecilia Gastardo-Conaco, Josefina Natividad, Mariah Allyson Banal (2021). Depressive symptoms among young adults in the Philippines: Results from a nationwide cross-sectional survey. Journal of Affective Disorders Reports, 3, 100073. https://doi.org/10.1016/j.jadr.2020.100073

J;, H.P.T. (2017 January). Artificial Intelligence in medicine, Metabolism: clinical and experimental. U.S. National Library of Medicine. Available at: https://pubmed.ncbi.nlm.nih.gov/28126242/

Koç, M. (2019, September). Investigation of Emotional Expression as a Predictor of Psychological Symptoms. International Journal of Psychology and Educational Studies, 6(3), 158-164. https://doi.org/10.17220/ijpes.2019.03.015

Kogan, L., et al. (2021, July). The Psychosocial Influence of Companion Animals on Positive and Negative Affect during the COVID-19 Pandemic. Journal of Veterinary Behavior, 42, 35-41. https://doi.org/10.1016/j.jveb.2021.04.017

Lichtenegger, F. (2017, July 12). Warum kann der "Akinator" noch immer unsere Gedanken lesen? Retrieved September 13, 2022, from https://www.vice.com/de/article/vbmz7m/warum-kann-der-akinator-auch-10- jahre-spater-noch-unsere-gedanken-lese.

Lim, L. T. S., Regencia, Z. J. G., Dela Cruz, J. R. C., Ho, F. D. V., Rodolfo, M. S., Ly-Uson, J., et al. (2022). Assessing the effect of the COVID-19 pandemic, shift to online learning, and social media use on the mental health of college students in the Philippines: A mixed-method study protocol. PLoS ONE, 17(5), e0267555. https://doi.org/10.1371/journal.pone.0267555

Lindberg, S. (2022). What Is Talk Therapy? Healthline. Retrieved July 10, 2023, from https://www.healthline.com/health/mental-health/talk-therapy#types

Luxton, D. D., McCann, R. A., Bush, N., & Mishkind, M. C. (2011). mHealth for Mental Health: Integrating Smartphone Technology in Behavioral Healthcare. Professional Psychology: Research and Practice, 42(6), 505-512. https://doi.org/10.1037/a0024485

Malhotra, S. (2020, March 16). Healthcare Chatbot Development with Dialogflow: Reshaping Diagnosis. AI Oodles. Retrieved from https://artificialintelligence.oodles.io/blogs/healthcare-chatbot-development-withdialogflow/#!

Martinez, A. B. (2020, August 20). Filipino help-seeking for mental health problems and associated barriers and facilitators: a systematic review. Social Psychiatry and Psychiatric Epidemiology, 55(1), 11-22. https://doi.org/10.1007/s00127-020- 01937-2

Merrill Jr., K., Kim, J., & Collins, C. (2022). AI companions for lonely individuals and the role of social presence. Journal of Mental Health, 1-9. https://doi.org/10.1080/08824096.2022.2045929

IJRP.ORG
International Journal of Research Publications
ISSN: 2708-3578 (Online)

Appendix A. Chatbot Application Video

For more information on how the Chatbot application works and what are functions of this application you could visit this link:

https://www.youtube.com/watch?v=rrj0WKRBShg