**Руководство пользователя**

**1.1 Назначение**

Программа предназначена для решения широкого спектра задач от ведения планов до психологической помощи.

- Функция тайм-менеджмента и планирования, бот может выступать в качестве личного секретаря: записывать идеи, составлять списки дел, регулировать время пользователя и держать его в курсе событий, которые он запланировал.

- Функция «иллюзия» собеседника; имитация общения с живым человеком посредством анализа данных пользователя, использования заданной темы для разговора.

- Функция личного психолога, работая в режиме которой бот легко сможет оказывать мотивационную поддержку, давать советы пользователям на основании их личных жизненных целей, текущей проблемы, настроения, а также психологического опыта и врачебной практики.

**1.2 Условия выполнения программы:**

Для выполнения программы необходима заполненная база данных и ЭВМ соответствующая следующим характеристикам:

Intel Core i3 2.0 Ghz

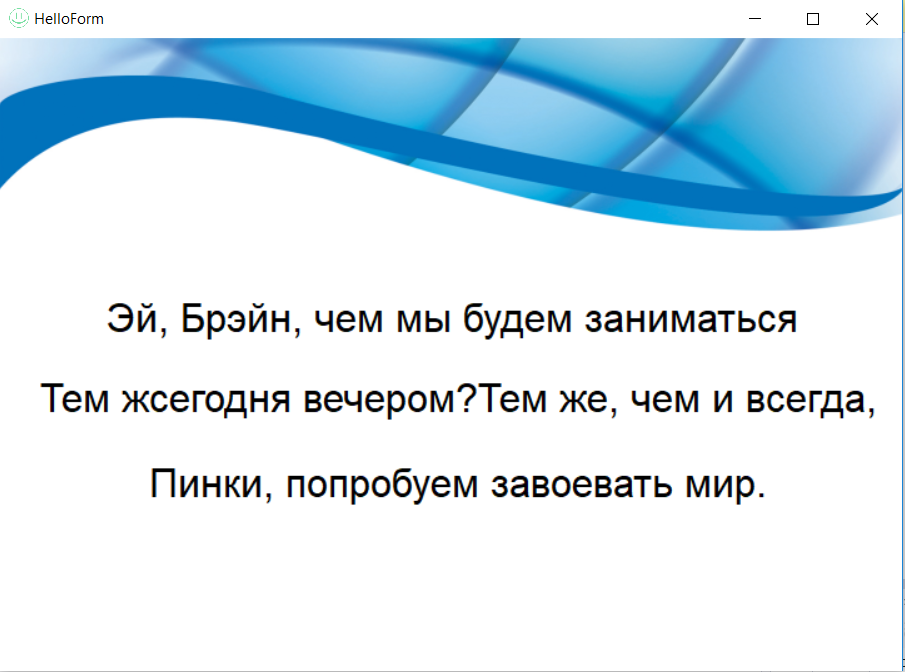
4гб ОЗУ

Стандартный корректно работающий встроенный микрофон.

В любом ином случае работа с данным программным продуктом невозможна.

**1.3 Выполнение программы:**

Перед непосредственной работой в приложении бот приветствует пользователя:



После того, как бот прочитает приветствие, программа откроет главное меню приложения.



Главная форма программы

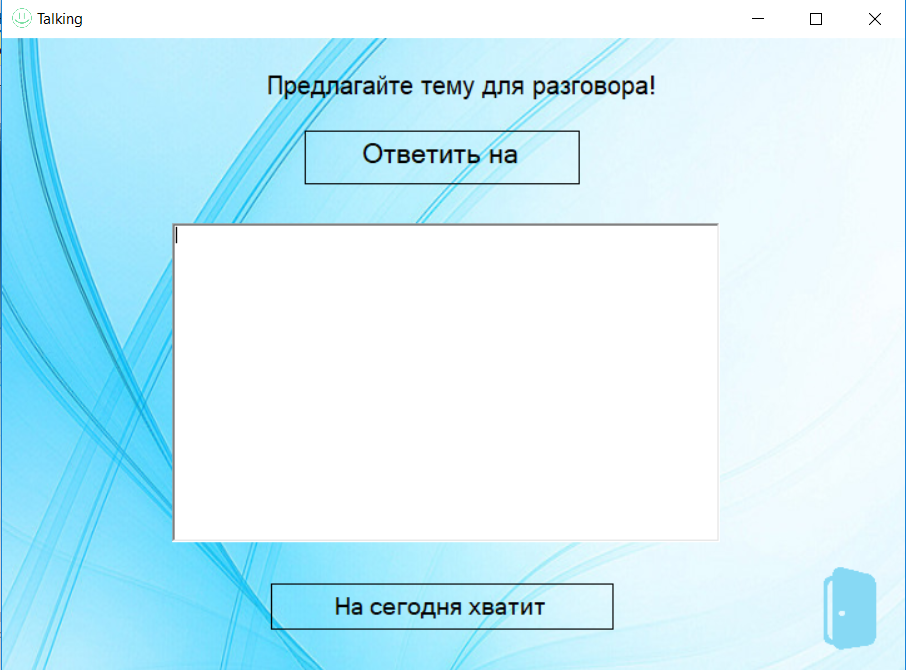
С этой формы доступен переход ко всем основным функциям программы:

1. Переключение приложения для работы в фоновом режиме
2. Вызов расписания на любую дату
3. Переход в режим бота-психолога
4. Переход в режим интерактивного диалога с пользователем
5. Переход в режим личного секретаря – для редактирования списка дел, расписания и т.п.
6. Выход из приложения

Также отдельно стоит выделить голосовые функции, которые сигнализируют боту выполнить то или иное действие: «Хочу поговорить» - действие аналогично пункту 4 выше, «Мне плохо», «Нужен совет» - действие аналогично пункту 3 выше, «Бот, можешь отдохнуть» действие аналогично пункту 1 выше, «Бот, список дел» - действие аналогично пункту 2 выше, «Бот, запиши в расписание» - действие аналогично пункту 5 выше.

На любые другие голосовые команды в данном режиме бот будет отвечать, что «не совсем понял, что пользователь хотел сказать».

Если бота оскорбить, то он отреагирует обидой и «уйдет» - немедленно прекратит выполнение программы и всех ее процессов.



Форма интерактивного диалога с пользователем – имитация общения.

После выбора темы для разговора бот начнет диалог – для этого следует текстом написать в поле для ввода тему для разговора, которой бот будет придерживаться или же воспользоваться голосовым вводом и просто вслух назвать боту тему для разговора.

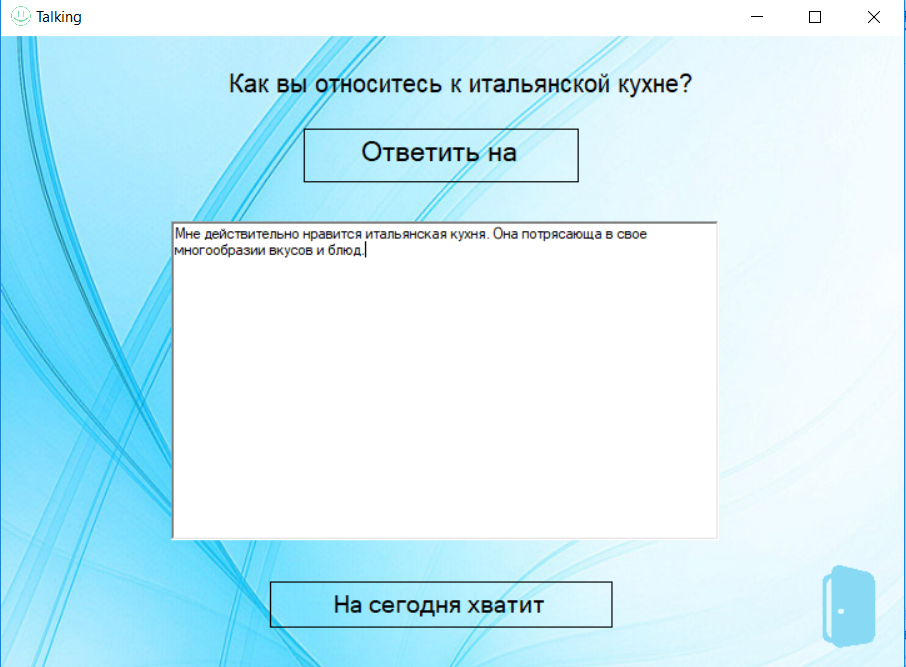
Форма непринужденной беседы включает в себя следующие кнопки:

- Ответить на – предназначена для разговора с ботом, ввода данных от пользователя.

- На сегодня хватит – предназначена для возвращения к форме главного меню.

- Дверь – при нажатии на данную кнопку произойдет выход из программы.

При показе формы бот предложит пользователю выбрать тему для разговора.



Форма психологической беседы с пользователем включает в себя следующие кнопки:

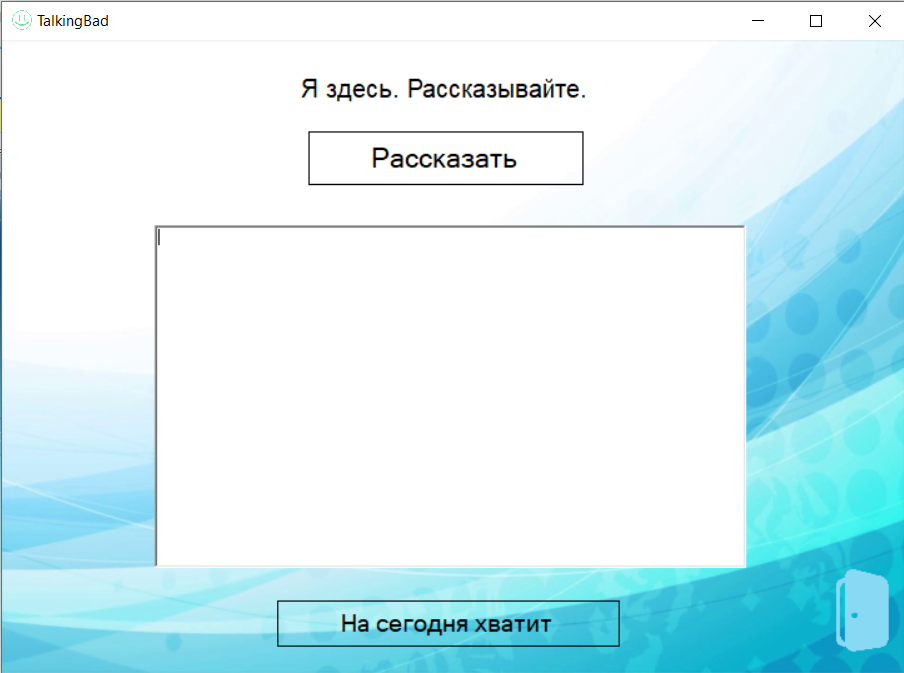
- Рассказать – предназначена для разговора с ботом, ввода данных от пользователя.

- На сегодня хватит – предназначена для возвращения к форме главного меню.

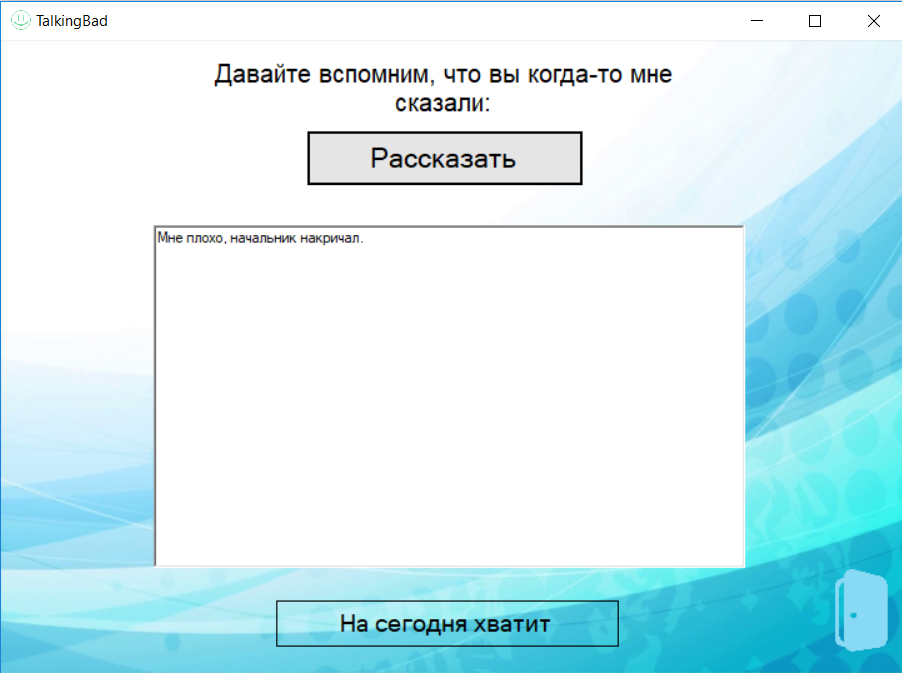
- Дверь – при нажатии на данную кнопку произойдет выход из программы.

Также на данной форме есть текстовое поле, посредством которого пользователь может обращаться к боту, если не желает использовать голосовой ввод.

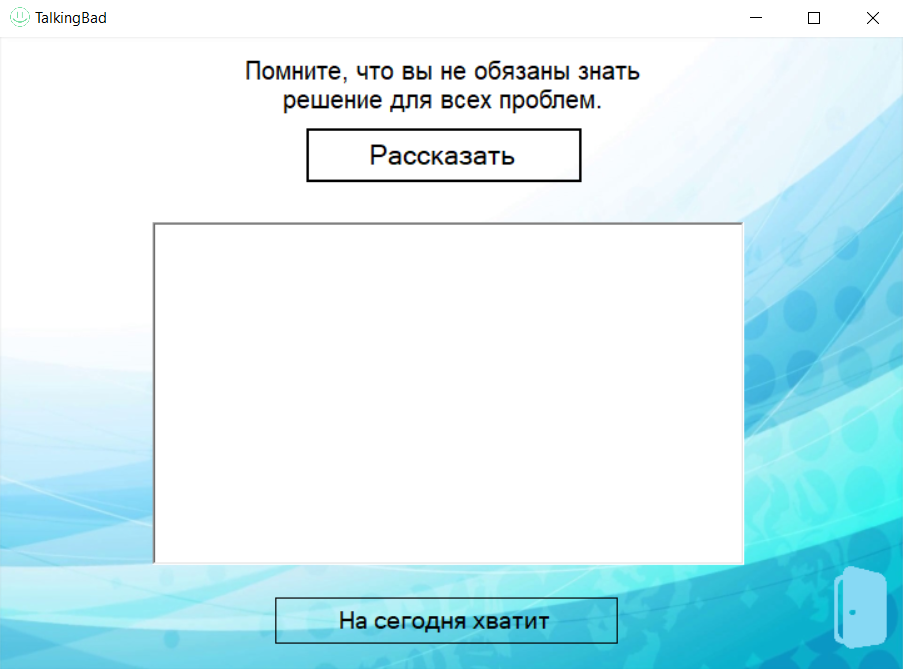
При открытии данной формы бот выразит готовность начать диалог с пользователем(Рис.10).



После того, как пользователь введет данные, расскажет боту о проблеме, будет проведен анализ сказанного и бот будет напоминать пользователю о его целях в жизни, давать психологические советы(Рис.11).



Психологический сеанс с ботом будет заканчиваться некой мудрой мыслью по поводу жизни, решения проблем и т.п. (Рис.12)



Форма редактирования расписания включает в себя следующий функционал:

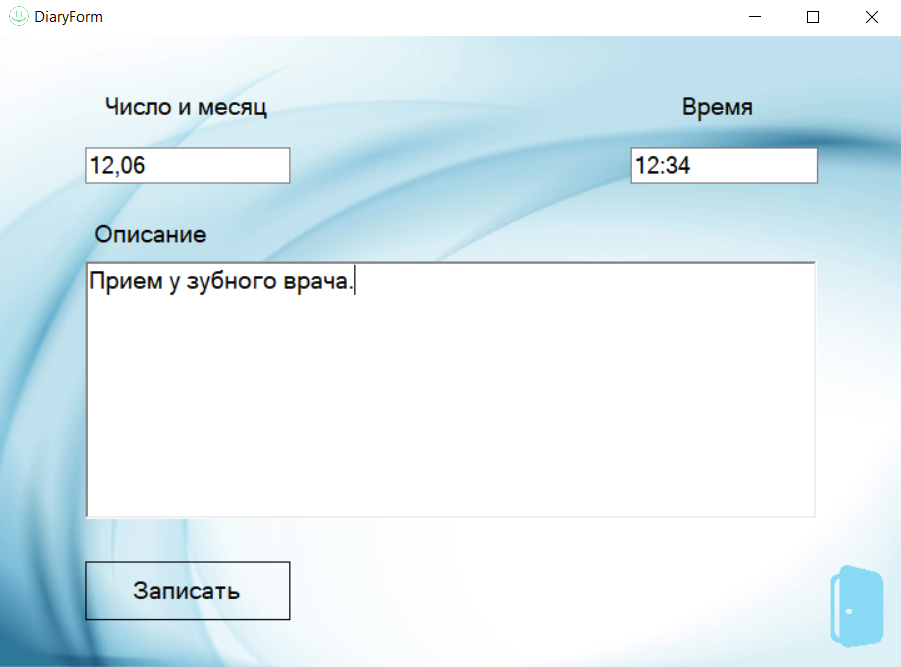
- Кнопка Записать – кнопка предназначена для записи нового пункта в расписание.

- Кнопка Дверь – при нажатии на данную кнопку произойдет выход из программы.

- Текстовое поле для ввода даты – предназначено для ввода даты события.

- Текстовое поле для ввода времени – предназначено для ввода времени события.

- Текстовое поле для ввода описания – предназначено для ввода описания расписания.



**1.4 Сообщения пользователю**

Программа предусматривает 2 вид сообщений для пользователя:

1. Сообщение о том, что пользователь ввел слишком много непонятной и несвязной информации – бот при помощи голоса сообщит пользователю следующую фразу «Я не волшебник, я только учусь. Боюсь, я не могу понять вас полностью, не могли бы вы упростить сказанное и рассказать мне то же самое немного проще», после чего пользователю будет предложено упростить текст.
2. Сообщение «Боюсь, я не совсем вас понял» - в случае, если бот не сможет распознать никаких из существующих команд или слов его грамматического словаря, он будет при помощи голоса выводить эту фразу.
3. Сообщение «Вы не выбрали дату. Я не могу записать в расписание дело, которое не ясно когда нужно выполнить» - в случае, если пользователь попытается внести новый пункт в расписание, не выставив дату.

**РУКОВОДСТВА ПРОГРАММИСТА**

1. **Аннотация**

Приводится руководство программиста программного обеспечения для психологической поддержки людям, имитации собеседника для пациентов различных клиник, которым постоянно нужен собеседник в силу определенных причин на примере центра помощи людям, которые хотят совершить самоубийство.

Программный продукт должен быть разработан для пункта оказания психологической помощи пациентам психиатрической клиники в целях улучшения самочувствия пациентов, повышения их способности к саморегенерации на фоне отвлечения от болезней и проблем, создания дружественной и благоприятной атмосферы для успешного выздоровления больных.

В руководстве программиста рассматриваются назначения, характеристики, условия, необходимые для выполнения программы.

1. **Назначение программы**

Программный продукт будет использоваться в психиатрической клинике №2 г. Невель, а также центральной больнице общей терапии №4 г. Валенсия после приема пациентов, при бодрствовании пациентов по их желанию или желанию лечащего врача. Программа должна придерживаться следующих принципов: создание дружественной пациентам атмосферы, настрой людей на позитивный лад, постоянная мотивация и поддержка.

Данным продуктом смогут воспользоваться все желающие пациенты или пациенты, которым необходимо постоянное общение по мнению врача.

1. **Условия, необходимые для выполнения программы**

Для работы программного продукта необходима следующая программно-аппаратная конфигурация:

* Windows 7,Windows Server 2003 Service Pack 2,Windows Server 2008,Windows Server 2008 R2,Windows Vista, Windows Vista Service Pack 1,Windows XP Service Pack 2,Windows XP Service Pack 3;
* 32-разрядные системы: компьютер, оборудованный процессором Intel или совместимым процессором с тактовой частотой 2 ГГц или выше (рекомендуется 2,4 ГГц или выше, поддерживается только один процессор);
* 64-разрядные системы: процессор с тактовой частотой 2 ГГц или выше (рекомендуется 2,4 ГГц или выше, поддерживается только один процессор);
* минимальный объем ОЗУ 1ГБ (рекомендуется 2 ГБ или более);
* 1 ГБ свободного места на диске;

**Характеристики программы**

* 1. **Режим работы программы**

Программа представляет собой Desktop приложение.

* 1. **Средства проверки правильности выполнения программы**

Проверка правильности работы осуществляется при выполнении конкретных примеров. Программа выдаёт голосовые сообщения о неправильном использовании каких-либо элементов интерфейса или голосовых команд. Пользователю при неправильной эксплуатации будет выдано соответствующее голосовое предупреждение, после чего будет предложено скорректировать свои действия определенным образом.

**4.3. Функционирование программы после сбоев**

В случае аварийного прекращения работы программы, вызванного внешними причинами, программа при устранении причины сбоя загружается вновь работоспособной. Уникальной особенностью бота является постоянный анализ данных, вводимых пользователем. В случае сбоя информация, которая была проанализирована и успешно обработана сохранится, а та часть, которая не успеет, соответственно – нет.

1. **Обращение к программе**

Для запуска программы необходимо произвести двойной клик по ярлыку, который обозначает программу.

1. **Входные и выходные данные**

К входной информации относится:

1. Речь пользователя – голосовой или текстовый ввод: информация о различных сферах и аспектах жизни пользователя, как:

- хобби

- цели

- любимая музыка

- любимое кино

- и т.п.

1. Данные о настроении пользователя.

- хорошее

- плохое

- нейтральное

1. Текущее расписание пользователя.
2. База вопросов: ввод новой темы для разговоров с пользователем посредством задания темы и некоторого кол-ва стандартных вопросов.
3. Имя бота-помощника.

К выходной информации относится:

1. Советы бота-помощника, выдаваемые пользователю в процессе общения в форме голосового монолога, если он о таком попросит.
2. Динамически изменяющееся в процессе работы индивидуальное расписание пользователя.
3. Справочная информация пользователя в форме обращения «Вопрос-Ответ», например:

«- Бот, что ты умеешь?»

«- Я умею разговаривать, анализировать данные.»

«- А что еще?»

«- Я могу составлять ваше индивидуальное расписание, давать советы.»

1. Любые другие входные данные, проанализированные и преобразованные искусственным интеллектом в советы, аналитическую статистику ответов пользователя или базу вопросов, тем для общения.

Разработанный программный продукт имеет следующую структуру:

* HelloForm.cs – форма для приветствия пользователя при запуске приложения.
* MainForm.cs – основная форма приложения, с которой доступен весь основной функционал программы.
* Talking.cs – форма для ведения с пользователем непринужденных бесед.
* TalkingBad.cs – форма для ведения с пользователем психологической беседы.
* DiaryForm.cs – форма для создания и редактирования индивидуального интерактивного расписания пользователя.
* Hobby.txt, Food.txt, Life.txt, Music.txt, Films.txt – файлы, предназначенные для хранения тем для разговора с ботом по умолчанию.
* Like.txt, Dontlike.txt – файлы, предназначенные для хранения проанализированной информации о пользователе, о его предпочтениях, интересах и т.п.
* Talking.txt – файл, предназначенный для анализа речи пользователя. В случае сбоев в работе системы может послужить для восстановления несохраненных данных.
* Psyhologist – файл-база психологических способностей бота. Лучше не редактировать по избежание сбоев в функциях бота-психолога.

1. **Описание основных функций**

Для составления чека программа должна выполнять следующие функции:

* Анализ данных, вводимых пользователем
* Формирование мотивационного совета пользователю
* Выдача данных о планах пользователя
* Заполнение расписания пользователя

**Функция анализа данных пользователя:**

Основная задача данной функции – при непринужденной беседе с ботом или в случае психологической беседы бот постоянно анализирует данные пользователя, поэтому необходимо постоянно менять схему разговора в зависимости от появления новых данных. Также предназначена для поиска ключевых слов с целью создания аналитической картины настроения и состояния пользователя и сохранения этих данных.

Эта функция имеет следующий код реализации:

switch (richTextBox1.Text)

{

case "поговорим о жизни":

theme = "жизнь";

dywt = true;

StreamReader read = new StreamReader("life.txt", Encoding.Default);

while (!read.EndOfStream)

{

string buf = read.ReadLine();

qw.Add(buf);

}

nextqw = r.Next(0, qw.Count);

string qwest = qw[nextqw];

string correctqw = "";

for (int u = 0; u < qwest.Length; u++)

{

if (qwest[u] != '1' && qwest[u] != '2' && qwest[u] != '3' && qwest[u] != '4' && qwest[u] != '5' && qwest[u] != '6' && qwest[u] != '7' && qwest[u] != '8' && qwest[u] != '9' && qwest[u] != '0' && qwest[u] != '.')

{

correctqw += qwest[u];

}

}

label2.Text = correctqw;

// Initialize a new instance of the SpeechSynthesizer.

SpeechSynthesizer synth = new SpeechSynthesizer();

synth.SetOutputToDefaultAudioDevice();

synth.Speak(label2.Text);

read.Close();

break;

case "поговорим об играх":

theme = "игры";

dywt = true;

StreamReader read5 = new StreamReader("games.txt", Encoding.Default);

while (!read5.EndOfStream)

{

string buf = read5.ReadLine();

qw.Add(buf);

}

nextqw = r.Next(0, qw.Count);

string qwest5 = qw[nextqw];

string correctqw5 = "";

for (int u = 0; u < qwest5.Length; u++)

{

if (qwest5[u] != '1' && qwest5[u] != '2' && qwest5[u] != '3' && qwest5[u] != '4' && qwest5[u] != '5' && qwest5[u] != '6' && qwest5[u] != '7' && qwest5[u] != '8' && qwest5[u] != '9' && qwest5[u] != '0' && qwest5[u] != '.')

{

correctqw5 += qwest5[u];

}

}

label2.Text = correctqw5;

// Initialize a new instance of the SpeechSynthesizer.

SpeechSynthesizer synth5 = new SpeechSynthesizer();

synth5.SetOutputToDefaultAudioDevice();

synth5.Speak(label2.Text);

read5.Close();

break;

case "поговорим о музыке":

theme = "музыка";

dywt = true;

StreamReader read6 = new StreamReader("music.txt", Encoding.Default);

while (!read6.EndOfStream)

{

string buf = read6.ReadLine();

qw.Add(buf);

}

nextqw = r.Next(0, qw.Count);

string qwest6 = qw[nextqw];

string correctqw6 = "";

for (int u = 0; u < qwest6.Length; u++)

{

if (qwest6[u] != '1' && qwest6[u] != '2' && qwest6[u] != '3' && qwest6[u] != '4' && qwest6[u] != '5' && qwest6[u] != '6' && qwest6[u] != '7' && qwest6[u] != '8' && qwest6[u] != '9' && qwest6[u] != '0' && qwest6[u] != '.')

{

correctqw6 += qwest6[u];

}

}

label2.Text = correctqw6;

// Initialize a new instance of the SpeechSynthesizer.

SpeechSynthesizer synth6 = new SpeechSynthesizer();

synth6.SetOutputToDefaultAudioDevice();

synth6.Speak(label2.Text);

read6.Close();

break;

case "поговорим о хобби":

theme = "хобби";

dywt = true;

StreamReader read9 = new StreamReader("hobby.txt", Encoding.Default);

while (!read9.EndOfStream)

{

string buf = read9.ReadLine();

qw.Add(buf);

}

nextqw = r.Next(0, qw.Count);

string qwest9 = qw[nextqw];

string correctqw9 = "";

for (int u = 0; u < qwest9.Length; u++)

{

if (qwest9[u] != '1' && qwest9[u] != '2' && qwest9[u] != '3' && qwest9[u] != '4' && qwest9[u] != '5' && qwest9[u] != '6' && qwest9[u] != '7' && qwest9[u] != '8' && qwest9[u] != '9' && qwest9[u] != '0' && qwest9[u] != '.')

{

correctqw9 += qwest9[u];

}

}

label2.Text = correctqw9;

// Initialize a new instance of the SpeechSynthesizer.

SpeechSynthesizer synth9 = new SpeechSynthesizer();

synth9.SetOutputToDefaultAudioDevice();

synth9.Speak(label2.Text);

read9.Close();

break;

case "поговорим о еде":

theme = "еда";

dywt = true;

StreamReader read1 = new StreamReader("food.txt", Encoding.Default);

while (!read1.EndOfStream)

{

string buf = read1.ReadLine();

qw.Add(buf);

}

nextqw = r.Next(0, qw.Count);

string qwest2 = qw[nextqw];

string correctqw2 = "";

for (int u = 0; u < qwest2.Length; u++)

{

if (qwest2[u] != '1' && qwest2[u] != '2' && qwest2[u] != '3' && qwest2[u] != '4' && qwest2[u] != '5' && qwest2[u] != '6' && qwest2[u] != '7' && qwest2[u] != '8' && qwest2[u] != '9' && qwest2[u] != '0' && qwest2[u] != '.')

{

correctqw2 += qwest2[u];

}

}

label2.Text = correctqw2;

// Initialize a new instance of the SpeechSynthesizer.

SpeechSynthesizer synth2 = new SpeechSynthesizer();

synth2.SetOutputToDefaultAudioDevice();

synth2.Speak(label2.Text);

read1.Close();

break;

case "поговорим о кино":

theme = "кино";

dywt = true;

StreamReader read13 = new StreamReader("films.txt", Encoding.Default);

while (!read13.EndOfStream)

{

string buf = read13.ReadLine();

qw.Add(buf);

}

nextqw = r.Next(0, qw.Count);

string qwest3 = qw[nextqw];

string correctqw3 = "";

for (int u = 0; u < qwest3.Length; u++)

{

if (qwest3[u] != '1' && qwest3[u] != '2' && qwest3[u] != '3' && qwest3[u] != '4' && qwest3[u] != '5' && qwest3[u] != '6' && qwest3[u] != '7' && qwest3[u] != '8' && qwest3[u] != '9' && qwest3[u] != '0' && qwest3[u] != '.')

{

correctqw3 += qwest3[u];

}

}

label2.Text = correctqw3;

// Initialize a new instance of the SpeechSynthesizer.

SpeechSynthesizer synth13 = new SpeechSynthesizer();

synth13.SetOutputToDefaultAudioDevice();

synth13.Speak(label2.Text);

read13.Close();

break;

default:

string[] words = { };

if (richTextBox1.Text != "")

{

StreamWriter wr = new StreamWriter("talking.txt", false);

foreach (string s in richTextBox1.Lines)

{

wr.WriteLine(s);

}

wr.WriteLine("%");

wr.Close();

int nespos = 0;

StreamReader re = new StreamReader("talking.txt", true);

string buf = "";

buf = re.ReadLine();

if (buf[buf.Length-1]!='.')

{

buf += ".";

}

re.Close();

bool trig = false;

words = buf.Split(new char[] { ' ', ',', ';', ':', '(', ')', '-', '\_' }, StringSplitOptions.RemoveEmptyEntries);

while (trig == false)

{

for (int i = 0; i < words.Length; i++)

{

string check = words[i];

for (int j = 0; j < check.Length; j++)

{

if (check[j] == '.' || check[j] == '!' || check[j] == '?')

{

Array.Resize(ref words, words.Length + 1);

words[words.Length - 1] = Convert.ToString(check[j]);

MessageBox.Show(words[words.Length - 1]);

check = check.Substring(0, check.Length - 1);

words[i] = check;

MessageBox.Show(words[i]);

trig = true;

break;

}

}

if (trig == true)

{

break;

}

//MessageBox.Show(words[i]);

}

}

for (int i = 0; i < good.Length; i++)

for (int j = 0; j < words.Length; j++)

{

if (words[j] == good[i])

{

nespos = j;

break;

}

}

for (int i = 0; i < words.Length; i++)

{

//MessageBox.Show(words[i]);

}

ilike = "";

dontlike = "";

bool bad= false;

for (int i=0; i < nespos; i++)

{

if (words[i]=="не")

{

bad = true;

break;

}

}

if (bad == true)

{

for (int i = 0; i < words.Length; i++)

{

if (words[i] != "." && words[i] != "!" && words[i] != "?")

{

//MessageBox.Show(ilike);

dontlike += words[i]+" ";

}

else

{

break;

}

}

StreamWriter w = new StreamWriter("dontlike.txt", true);

w.WriteLine(dontlike);

w.Close();

}

else

{

for (int i = nespos - 1; i < words.Length; i++)

{

if (words[i] != "." && words[i] != "!" && words[i] != "?")

{

//MessageBox.Show(ilike);

ilike += words[i] + " ";

}

else

{

break;

}

}

StreamWriter w = new StreamWriter("like.txt", true);

w.WriteLine(ilike);

w.Close();

}

if (ilike!="" || dontlike != "")

{

qw.RemoveAt(nextqw);

}

if (qw.Count > 0)

{

nextqw = r.Next(0, qw.Count);

string qwest1 = qw[nextqw];

string correctqw1 = "";

for (int u = 0; u < qwest1.Length; u++)

{

if (qwest1[u] != '1' && qwest1[u] != '2' && qwest1[u] != '3' && qwest1[u] != '4' && qwest1[u] != '5' && qwest1[u] != '6' && qwest1[u] != '7' && qwest1[u] != '8' && qwest1[u] != '9' && qwest1[u] != '0' && qwest1[u] != '.')

{

correctqw1 += qwest1[u];

}

}

label2.Text = correctqw1;

// Initialize a new instance of the SpeechSynthesizer.

SpeechSynthesizer synth1 = new SpeechSynthesizer();

synth1.SetOutputToDefaultAudioDevice();

synth1.Speak(label2.Text);

}

else

{

SpeechSynthesizer synth1 = new SpeechSynthesizer();

synth1.SetOutputToDefaultAudioDevice();

synth1.Speak("Мне надоело говорить на эту тему. Давайте поговорим о чем-нибудь другом или закончим пока беседу.");

vib = r.Next(1, 4);

if (vib == 1)

{

label2.Text = "О чем поговорим?";

label2.Visible = true;

SpeechSynthesizer synth11 = new SpeechSynthesizer();

// Configure the audio output.

synth11.SetOutputToDefaultAudioDevice();

// Speak a string.

synth11.Speak(label2.Text);

}

if (vib == 2)

{

label2.Text = "На какую тему будем общаться?";

label2.Visible = true;

SpeechSynthesizer synth11 = new SpeechSynthesizer();

// Configure the audio output.

synth11.SetOutputToDefaultAudioDevice();

// Speak a string.

synth11.Speak(label2.Text);

}

if (vib == 3)

{

label2.Text = "Предлагайте тему для разговора!";

label2.Visible = true;

SpeechSynthesizer synth11 = new SpeechSynthesizer();

// Configure the audio output.

synth11.SetOutputToDefaultAudioDevice();

// Speak a string.

synth11.Speak(label2.Text);

}

}

}

break;

}

richTextBox1.Clear();

**Функция формирования мотивационного совета:**

Основная задача данной функции – формирование структуры диалога с пользователем в режиме бота-психолога.

Эта функция имеет следующий код реализации:

private void TalkingBad\_Load(object sender, EventArgs e)

{

beginDiag = new string [] {"Здравствуйте. Что случилось?", "Приветствую. В чем дело?", "Вижу, вы хотите о чем-то рассказать. Я слушаю.", "Хотите чем-то поделиться? Я готов.", "Я здесь. Рассказывайте." };

Random r = new Random();

int phrase\_Numb = r.Next(0,beginDiag.Length);

label1.Text = beginDiag[phrase\_Numb];

interv = 0;

StreamReader rea = new StreamReader("like.txt", Encoding.Default);

SpeechSynthesizer synth3 = new SpeechSynthesizer();

synth3.SetOutputToDefaultAudioDevice();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

SpeechSynthesizer synth3 = new SpeechSynthesizer();

StreamReader read = new StreamReader("like.txt", Encoding.Default);

while (!read.EndOfStream)

{

motiv.Add(read.ReadLine());

}

Random ran = new Random();

int motii = ran.Next(0,motiv.Count);

synth3.SetOutputToDefaultAudioDevice();

StreamReader re = new StreamReader("helloPsychologist.txt", Encoding.Default);

List<string> hello = new List<string>();

while (!re.EndOfStream)

{

hello.Add(re.ReadLine());

}

Random ry = new Random();

re.Close();

label1.Text = hello[r.Next(0,qw.Count)];

synth3.Speak(label1.Text);

synth3.Speak(motiv[motii]);

StreamReader rep = new StreamReader("psychologist.txt", Encoding.Default);

List<string> ps = new List<string>();

while (!rep.EndOfStream)

{

ps.Add(rep.ReadLine());

}

rep.Close();

label1.Text = ps[r.Next(0, ps.Count)];

synth3.Speak(label1.Text);

richTextBox1.Clear();

}

private void button2\_Click(object sender, EventArgs e)

{

MainMenu m = new MainMenu(this);

m.Show();

this.Hide();

}

private void TalkingBad\_Shown(object sender, EventArgs e)

{

timer1.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

interv++;

if (interv==1)

{

SpeechSynthesizer synth3 = new SpeechSynthesizer();

synth3.SetOutputToDefaultAudioDevice();

synth3.Speak(label1.Text);

}

}

private void button4\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

**Функция выдачи данных о расписании пользователя:**

Основная задача данной функции – выдача данных о расписании пользователя на заданный день.

chisl = true;

l.Text = e.Result.Text;

timedate = l.Text;

//l.Text = e.Result.Text;

string[] mas = timedate.Split();

MessageBox.Show(Convert.ToString(mas.Length));

if (mas.Length == 3)

{

mas[0] += " " + mas[1];

mas[1] = mas[2];

Array.Resize(ref mas, mas.Length - 1);

}

MessageBox.Show(Convert.ToString(mas.Length));

string months = "";

string days = "";

for (int i = 0; i < day.Length; i++)

for (int j = 0; j < mas.Length; j++)

{

if (day[i] == mas[j])

{

days = Convert.ToString(i + 1);

break;

}

}

for (int i = 0; i < month.Length; i++)

for (int j = 0; j < mas.Length; j++)

{

if (month[i] == mas[j])

{

months = Convert.ToString(i + 1);

break;

}

}

timedate = "";

timedate += days + ",";

timedate += months;

if (timedate.Length == 3)

{

timedate = timedate.Insert(0, "0");

MessageBox.Show(Convert.ToString(timedate.Length));

timedate = timedate.Insert(3, "0");

}

bool flag = false;

int h = 0;

string buf = "";

MessageBox.Show(timedate);

StreamReader read = new StreamReader("timetable.txt", Encoding.Default);

while (!read.EndOfStream)

{

if (flag == true)

{

buf = read.ReadLine();

if (buf != "%")

{

tt.Add(buf);

}

else

{

break;

}

}

else

{

buf = read.ReadLine();

if (buf == timedate)

{

flag = true;

}

}

}

foreach (string l in tt)

{

MessageBox.Show(l);

}

read.Close();

SpeechSynthesizer synth5 = new SpeechSynthesizer();

// Configure the audio output.

synth5.SetOutputToDefaultAudioDevice();

// Speak a string.

string said = "";

foreach (string s in tt)

{

said += s + " ";

}

if (said == "")

{

synth5.Speak("У вас на этот день ничего не запланировано. Вы полностью свободны. Пока что.");

}

else

{

synth5.Speak("Диктую расписание на " + l.Text);

synth5.Speak(said);

said = "";

}

break;

**Функция заполнения индивидуального расписания:**

Основная задача данной функции – заполнении и редактирование расписания дня пользователя.

private void button1\_Click(object sender, EventArgs e)

{

StreamWriter wr = new StreamWriter("timetable.txt", true);

wr.WriteLine("Дата: " + maskedTextBox1.Text);

wr.WriteLine("Время: " + maskedTextBox2.Text);

foreach(string s in richTextBox1.Lines)

{

wr.WriteLine(s);

}

wr.WriteLine("%");

wr.Close();

richTextBox1.Clear();

}

public void sre\_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)

{

if (e.Result.Confidence > 0.60)

{

// l.Text = e.Result.Text;

if (e.Result.Text == "запиши в расписание" || e.Result.Text == "можно записывать" || e.Result.Text == "измени расписание ")

{

StreamWriter wr = new StreamWriter("timetable.txt", true);

string d1 = maskedTextBox1.Text;

string date = "";

for (int i=0; i<d1.Length; i++)

{

if(d1[i]!='0' && d1[i] != '.')

{

date += d1[i];

}

}

wr.WriteLine(date);

wr.WriteLine("Время: " + maskedTextBox2.Text);

foreach (string s in richTextBox1.Lines)

{

wr.WriteLine(s);

}

wr.WriteLine("%");

wr.Close();

SpeechSynthesizer synth = new SpeechSynthesizer();

// Configure the audio output.

synth.SetOutputToDefaultAudioDevice();

// Speak a string.

synth.Speak("Записано.");

maskedTextBox1.Clear();

maskedTextBox2.Clear();

richTextBox1.Clear();

}

if (e.Result.Text=="я закончил")

{

MainMenu m = new MainMenu(this);

m.Show();

this.Hide();

}

}