Echipa:
Albu A. Mihai
Cojocaru Alexandra-Elena
Bodescu Constantin-Vlad

ChatBot Simplu Amuzant

Noi prezentam este un chatbot simplu și amuzant care poate sa converseze cu utilizatorul cu ajutorul unor întrebări predefinite. Codul este realizat în Python și este user friendly.

ReadMe.md:

🤖 Simple Python Chatbot

This is a beginner-friendly, rule-based chatbot built using Python. It responds to predefined questions using simple logic, making it a great starting point for anyone new to programming or chatbot development.

💡 Features

Handles basic conversations (greetings, farewells, etc.)
Answers predefined questions (e.g., "What's your name?", "What can you do?")
100% written in Python — no external libraries required
Great for learning the basics of conditionals, functions, and user input

🛠️ How It Works

The chatbot listens for user input and responds based on a predefined set of questions and answers. It's not powered by AI or machine learning, but by logic you define yourself!

1. Implementarea

ChatBoard.py:

```
"""
Board .py file where the bot will be implemented
"""
// Importă modulele necesare din PySide6 pentru GUI
from PySide6 import QtCore
from PySide6.QtWidgets import (QWidget, QLabel, QPushButton, QLineEdit,
QVBoxLayout, QHBoxLayout, QScrollArea,
                               QSizePolicy)
from PySide6.QtCore import Qt

// Importă clasele locale necesare
from Design import Design
from Options import OptionsDialog
from Parser import Parser
import json //pentru încărcarea fișierului Q&A


class Board(QWidget): //moștenește fereastra principală PySide6
    def __init__(self):
        super().__init__() //inițializează QWidget
        self.data = None //aici se va stoca conținutul JSON
```

```
        # initialize data from Q&A json
        initjson(self) //încarcă datele din fișierul Q&A

        # create parser to format the input
        self.parser = Parser() //creează un parser pentru text

        # labels and buttons
        // configurări de bază ale ferestrei
        self.showFullScreen() //afișează fereastra pe tot ecranul
        self.setMinimumSize(600, 600) //dimensiune minimă
        self.setWindowTitle("Funny Chatbot") //titlu fereastră
        self.label = QLabel("Bot: Cum te pot face să zâmbești?") //etichetă
inițială
        self.label.setObjectName("greetingLabel") //nume pentru CSS
        self.input = QLineEdit() //câmp de introducere text
        self.input.setSizePolicy(QSizePolicy.Policy.Expanding,
QSizePolicy.Policy.Fixed) //setări de mărime
        self.ok = QPushButton("OK") //buton OK
        self.cancel = QPushButton("Cancel") //buton cancel
        self.cancel.setObjectName("cancel")  //IMPORTANT: setează ID-ul
pentru CSS
        self.options = QPushButton("Options") //buton pentru comenzi
disponibile

        // layout pentru butoane (orizontal)
        buttons = QHBoxLayout()
        buttons.addWidget(self.ok)
        buttons.addWidget(self.cancel)
        buttons.addWidget(self.options)

        self.scroll_area = QScrollArea() //zonă scrollabilă pentru mesaje
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
//fără scrollbar vizibil

        self.messages_widget = QWidget() //widget care va conține toate
mesajele
        self.messages_layout = QVBoxLayout() //layout vertical pentru mesaje
        self.messages_layout.setAlignment(Qt.AlignTop) //aliniere sus
        self.messages_widget.setLayout(self.messages_layout)

        self.scroll_area.setWidget(self.messages_widget) //conectează zona
scroll cu widgetul de mesaje

        # whole board layout style (V)
        //layout-ul principal al aplicației
        self.layout = QVBoxLayout()
        self.layout.addWidget(self.label)
        self.layout.addWidget(self.scroll_area)
        self.layout.addWidget(self.input)
        self.layout.addLayout(buttons)
```

```
        self.setLayout(self.layout) //aplică layout-ul ferestrei
        self.button_pressed() //conectează butoanele la funcții

        self.design = Design(self) //aplică design/stil aplicației

    # function for when we press a button
    def button_pressed(self):
        self.ok.clicked.connect(self.update_text) //la click pe OK -> trimite
mesajul
        self.cancel.clicked.connect(self.close) //la click pe Cancel ->
închide aplicația
        self.options.clicked.connect(self.show_commands) //click pe Options
-> afișează dialogul cu comenzi

    def update_text(self):
        # Get input
        text = self.input.text().strip() //ia textul introdus
        if not text:
            return  # Skip if empty //nu face nimic dacă e gol

        user_msg = QLabel(f"You: {text}") //creează mesajul utilizatorului
        self.design.style_message_label(user_msg, 'user') //stilizează
mesajul

        # user message
        user_container = QWidget() //creează container pentru mesaj
        user_layout = QHBoxLayout()
        user_layout.addStretch(1) //împinge mesajul spre dreapta
        user_layout.addWidget(user_msg)
        user_container.setLayout(user_layout)
        self.messages_layout.addWidget(user_container) //adaugă mesajul în
layout

        # bot response
        matched_key = self.parser.match_fuzzy(text, self.data.keys()) //caută
întrebarea în baza de date
        bot_response = self.data[
            matched_key] if matched_key else "Îmi pare rău, nu știu răspunsul
la această întrebare." //răspunsul botului

        # bot message
        bot_msg = QLabel(f"Bot: {bot_response}") //etichetă cu răspunsul
botului
        self.design.style_message_label(bot_msg, 'bot') //stil bot

        # Setup bot message container
        bot_container = QWidget()
        bot_layout = QHBoxLayout()
        bot_layout.addWidget(bot_msg)
        bot_layout.addStretch(1)  # Push left //împinge botul spre stânga
        bot_container.setLayout(bot_layout)
        self.messages_layout.addWidget(bot_container)
```

```
        self.input.clear() //golește câmpul de input

        # Force UI update
        self.messages_widget.updateGeometry() //actualizează layout-ul
        self.messages_layout.update()

        # Auto-scroll with delay //scrollează automat în jos cu întârziere
        QtCore.QTimer.singleShot(50, lambda: self.scroll_to_bottom())

    def scroll_to_bottom(self):
        scroll_bar = self.scroll_area.verticalScrollBar()
        scroll_bar.setValue(scroll_bar.maximum()) //setează scroll la maxim
        QtCore.QTimer.singleShot(10, lambda:
scroll_bar.setValue(scroll_bar.maximum())) //fix pentru unele glitch-uri

    def keyPressEvent(self, event):
        if event.key() == Qt.Key_Return or event.key() == Qt.Key_Enter:
            self.update_text() //trimite mesajul
        elif event.key() == Qt.Key_Escape:
            self.close() //închide aplicația
        else:
            event.ignore() //ignoră alte taste

    def show_commands(self):
        if not self.data:
            return //ieși dacă nu există date

        dialog = OptionsDialog(commands=self.data.keys(), parent=self)
//creează dialogul
        dialog.exec() //afișează-l


# json initialization function
def initjson(self):
    with open('./docs/Q&A.json', 'r', encoding='utf-8') as file:
        self.data = json.load(file)
//deschidem fișierul cu întrebări și răspunsuri și îl salvăm în self.data.
```

Main.py:

```
"""
The main .py file where the board will be called
"""
import sys
from PySide6.QtWidgets import QApplication
from ChatBoard import Board

app = QApplication(sys.argv)
window = Board()
window.show()
sys.exit(app.exec())
//aici începe toată aplicația. Se creează fereastra și se arată pe ecran.
```

Parser.py:

```
from fuzzywuzzy import process
import re


class Parser:
    def __init__(self):
        pass

    def parse_txt(self, text):
        text = text.lower()
        text = re.sub(r'[^\w\s]', '', text)
        text = re.sub(r'\s+', ' ', text).strip()
        return text

    def match_fuzzy(self, user_input, options, cutoff=70):
        parsed_input = self.parse_txt(user_input)
        match, score = process.extractOne(parsed_input, options)
        if score >= cutoff:
            return match
        return None
//ia întrebarea, o transformă în litere mici, scoate semnele de punctuație și spațiile duble. Astfel, botul înțelege întrebările chiar dacă le scrii cu greșeli sau simboluri aiurea.
```

Design.py:

```
from PySide6.QtGui import QFont //fonturi personalizate
from PySide6.QtCore import Qt //constante de aliniere
from PySide6.QtWidgets import QSizePolicy //controlează mărimea widgeturilor


class Design:
    def __init__(self, board):
        self.board = board //referință la fereastra principală
        self.apply_styles() //aplică CSS-ul
        self.setup_fonts() //setează fonturile
        self.setup_layout_spacing() //spațiere modernă

    def apply_styles(self): //CSS aplicat la întreaga fereastră
        self.board.setStyleSheet("""
                QWidget {
                    background: qlineargradient(x1:0, y1:0, x2:1, y2:1,
                        stop:0 #0f0f23, stop:1 #1a1a2e);
                    color: #f8f9fa;
                    font-family: 'Inter', 'SF Pro Display', -apple-system,
BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
                    font-size: 16px;
                    font-weight: 500;
                }

                QLabel#botLabel {
                    color: #64ffda;
                    font-weight: 600;
```

```
            text-shadow: 0 0 10px rgba(100, 255, 218, 0.3);
        }

        QLineEdit {
            background: rgba(255, 255, 255, 0.05);
            border: 1px solid rgba(100, 255, 218, 0.2);
            border-radius: 12px;
            padding: 12px 16px;
            color: #f8f9fa;
            font-size: 16px;
            backdrop-filter: blur(10px);
            transition: all 0.3s ease;
            white-space: nowrap;
            overflow: hidden;
            text-overflow: ellipsis;
            white-space: nowrap;
            qproperty-alignment: 'AlignVCenter | AlignLeft';
        }

    QLineEdit:focus {
        border: 2px solid #64ffda;
        background: rgba(255, 255, 255, 0.08);
        box-shadow: 0 0 20px rgba(100, 255, 218, 0.2);
    }

    QLineEdit:hover {
        border: 1px solid rgba(100, 255, 218, 0.4);
        background: rgba(255, 255, 255, 0.07);
    }

    QPushButton {
        background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
            stop:0 #64ffda, stop:1 #4fc3f7);
        border: none;
        border-radius: 10px;
        padding: 10px 20px;
        font-weight: 600;
        color: #0f0f23;
        min-width: 90px;
        font-size: 15px;
        letter-spacing: 0.5px;
    }

    QPushButton:hover {
        background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
            stop:0 #4ecdc4, stop:1 #29b6f6);
        transform: translateY(-1px);
        box-shadow: 0 4px 15px rgba(100, 255, 218, 0.3);
    }

    QPushButton:pressed {
        background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
            stop:0 #26a69a, stop:1 #0288d1);
```

```
                transform: translateY(0px);
            }

            QPushButton#cancel {
                background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 #ff4757, stop:1 #c44569);
                color: #ffffff;
                border: 2px solid #ff3742;
                font-weight: 700;
            }

            QPushButton#cancel:hover {
                background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 #ff3742, stop:1 #b33951);
                color: #ffffff;
                border: 2px solid #ff2832;
                box-shadow: 0 6px 20px rgba(255, 71, 87, 0.5);
                transform: translateY(-1px);
            }

            QPushButton#cancel:pressed {
                background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 #e73c4e, stop:1 #a73244);
                transform: translateY(0px);
                box-shadow: 0 2px 8px rgba(255, 71, 87, 0.3);
            }

            QScrollArea {
            background: transparent;
            border: none;
            }

            QScrollArea > QWidget > QWidget {
                background: transparent;
            }

            QLabel#greetingLabel {
                background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 rgba(100, 255, 218, 0.7), stop:1 rgba(79, 195, 247,
0.7));
                color: #0f0f23;
                border-radius: 18px 18px 18px 4px;
                padding: 14px 18px;
                margin: 20px 0;
                font-weight: 600;
                font-size: 18px;
                line-height: 1.4;
                box-shadow: 0 4px 15px rgba(100, 255, 218, 0.3);
                border-left: 4px solid #64ffda;
            }
        """)

    def setup_fonts(self):
```

```
        # Modern font hierarchy with better typography
        font_label = QFont("Inter", 18, QFont.Weight.DemiBold) //font pentru
etichete
        font_input = QFont("Inter", 14, QFont.Weight.Normal) //font pentru
input
        font_buttons = QFont("Inter", 13, QFont.Weight.DemiBold) //font
pentru butoane

        # Enable font smoothing //se setează fonturile pe componentele
principale
        font_label.setStyleHint(QFont.StyleHint.System)
        font_input.setStyleHint(QFont.StyleHint.System)
        font_buttons.setStyleHint(QFont.StyleHint.System)

        self.board.label.setFont(font_label)
        self.board.input.setFont(font_input)
        self.board.ok.setFont(font_buttons)
        self.board.cancel.setFont(font_buttons)

    def style_message_label(self, label, sender='bot'):
        label.setWordWrap(True) //textul se rupe pe mai multe rânduri

        if sender == 'bot':
            label.setStyleSheet(""" //stil pentru mesajele botului
                background: qlineargradient(x1:0, y1:0, x2:1, y2:0,
                    stop:0 rgba(100, 255, 218, 0.9), stop:1 rgba(79, 195, 247,
0.9));
                color: #0f0f23;
                border-radius: 18px 18px 18px 4px;
                max-width: 800px;
                qproperty-wordWrap: true;
                qproperty-minimumWidth: 300px;
                padding: 12px 16px;
                margin: 8px 8px 8px 16px;
                font-weight: 500;
                font-size: 16px;
                line-height: 1.4;
                box-shadow: 0 2px 10px rgba(100, 255, 218, 0.2);
            """)
            label.setAlignment(Qt.AlignmentFlag.AlignLeft |
Qt.AlignmentFlag.AlignVCenter) //aliniere stânga
        else:
            label.setStyleSheet(""" //stil pentru mesajele userului
                background: rgba(255, 255, 255, 0.08);
                color: #f8f9fa;
                max-width: 800px;
                qproperty-wordWrap: true;
                qproperty-minimumWidth: 300px;
                border-radius: 18px 18px 4px 18px;
                padding: 12px 16px;
                margin: 8px 16px 8px 8px;
                font-weight: 600;
                font-size: 16px;
```

```
                line-height: 1.4;
                border: 1px solid rgba(255, 255, 255, 0.1);
                backdrop-filter: blur(10px);
            """)
            label.setAlignment(Qt.AlignmentFlag.AlignRight |
Qt.AlignmentFlag.AlignVCenter) //aliniere dreapta

        label.setSizePolicy(QSizePolicy.Policy.Preferred,
QSizePolicy.Policy.Fixed)

    def setup_layout_spacing(self):
        # Modern spacing with better visual hierarchy
        //setează spațierea între elemente
        self.board.layout.setSpacing(20)
        self.board.layout.setContentsMargins(24, 24, 24, 24)
        self.board.messages_layout.setSpacing(6)
        self.board.messages_layout.setContentsMargins(16, 16, 16, 16)

    def add_glassmorphism_effect(self, widget):
        # Add glassmorphism effect to any widget
        //adaugă efect de transparență + blur (glassmorphism)
        widget.setStyleSheet(widget.styleSheet() + """
            background: rgba(255, 255, 255, 0.05);
            backdrop-filter: blur(20px);
            border: 1px solid rgba(255, 255, 255, 0.1);
            border-radius: 16px;
        """)

    def add_subtle_animation_class(self):
        # CSS-like transition effects (can be used with QPropertyAnimation)
        //returnează o tranziție animată ca în CSS
        return """
            transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
        """

    def get_theme_colors(self):
        # Modern color palette for consistent theming
        //returnează o paletă de culori pentru temă
        return {
            'primary': '#64ffda',
            'primary_hover': '#4ecdc4',
            'primary_pressed': '#26a69a',
            'secondary': '#4fc3f7',
            'background_primary': '#0f0f23',
            'background_secondary': '#1a1a2e',
            'surface': 'rgba(255, 255, 255, 0.05)',
            'surface_hover': 'rgba(255, 255, 255, 0.08)',
            'text_primary': '#f8f9fa',
            'text_secondary': '#b0bec5',
            'border': 'rgba(255, 255, 255, 0.1)',
            'shadow': 'rgba(100, 255, 218, 0.2)'
        }
```

Options.py

```
from PySide6.QtWidgets import QDialog, QVBoxLayout, QLabel, QPushButton,
QListWidget, QHBoxLayout
from PySide6.QtGui import QFont


class OptionsDialog(QDialog):
   def __init__(self, commands, parent=None):
       super().__init__(parent)
       self.setWindowTitle("Chatbot commands") //titlu fereastră
       self.setMinimumSize(400, 300) //dimensiune minimă

       layout = QVBoxLayout() //layout vertical

       label = QLabel("Available commands:") //etichetă
       label.setFont(QFont("Arial", 12, QFont.Bold)) //font bold
       layout.addWidget(label) //adaugă eticheta în layout

       self.list_widget = QListWidget() //listă cu comenzile disponibile
       self.list_widget.addItems(sorted(commands)) //adaugă comenzile în
listă
       layout.addWidget(self.list_widget) //adaugă lista în layout

       button_layout = QHBoxLayout() //layout orizontal pentru buton
       close_btn = QPushButton("Close window") //buton închidere
       close_btn.clicked.connect(self.close) //acțiune la click
       button_layout.addStretch() //împinge butonul spre dreapta
       button_layout.addWidget(close_btn)
       layout.addLayout(button_layout) //adaugă layout-ul butonului în
layout principal

       self.setLayout(layout) //aplică layout-ul principal
```

Structura logica:

Main.py
↓
ChatBoard.py (Board - fereastra principală)
↓
Design.py → stiluri și fonturi moderne
Parser.py → înțelegerea întrebărilor (fuzzy match)
Options.py → fereastră cu comenzile disponibile
Q&A.json → întrebări și răspunsuri salvate

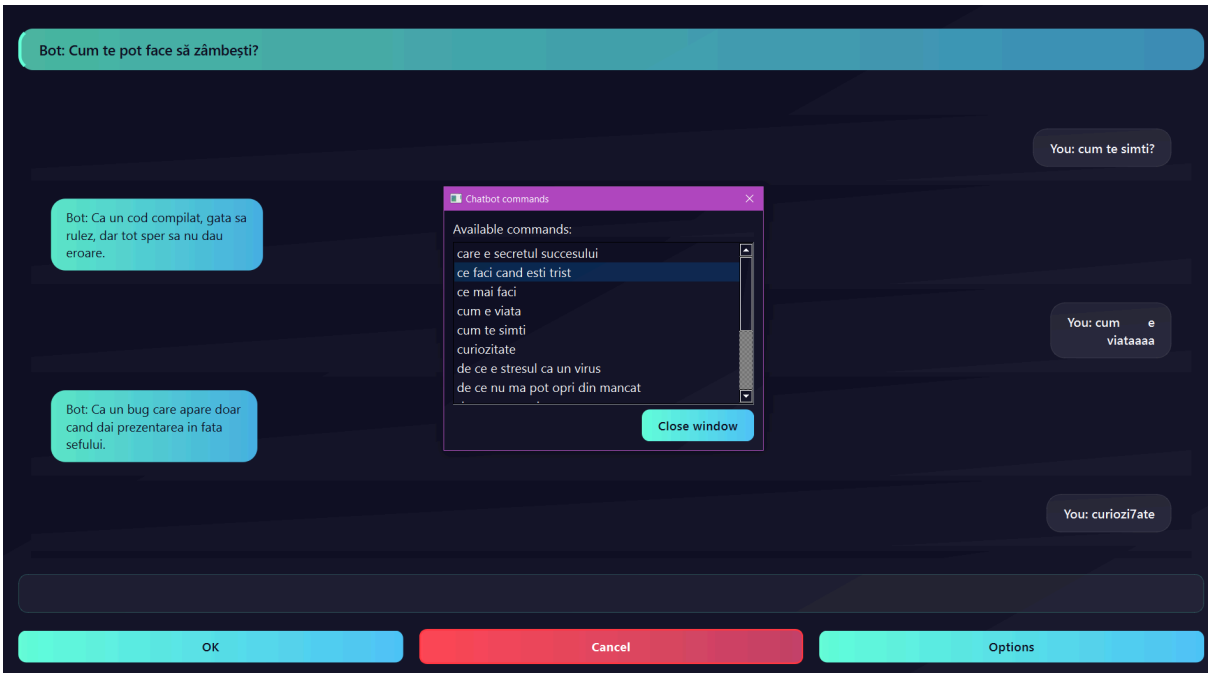
Fișier	Rol
Main.py →	Punctul de pornire – creează aplicația și deschide fereastra chatbotului
ChatBoard.py →	Clasa principală (`Board`) – gestionează UI-ul și logica de conversație
Parser.py →	Transformă întrebarea userului și găsește răspunsul cel mai apropiat
Design.py →	Aplică stiluri CSS moderne, fonturi și efecte vizuale (inclusiv mesaje stilizate)
Options.py →	Deschide o fereastră care arată toate comenzile posibile (cheile din Q&A)
Q&A.json →	Fișierul cu toate întrebările și răspunsurile botului

2. Functionalitate

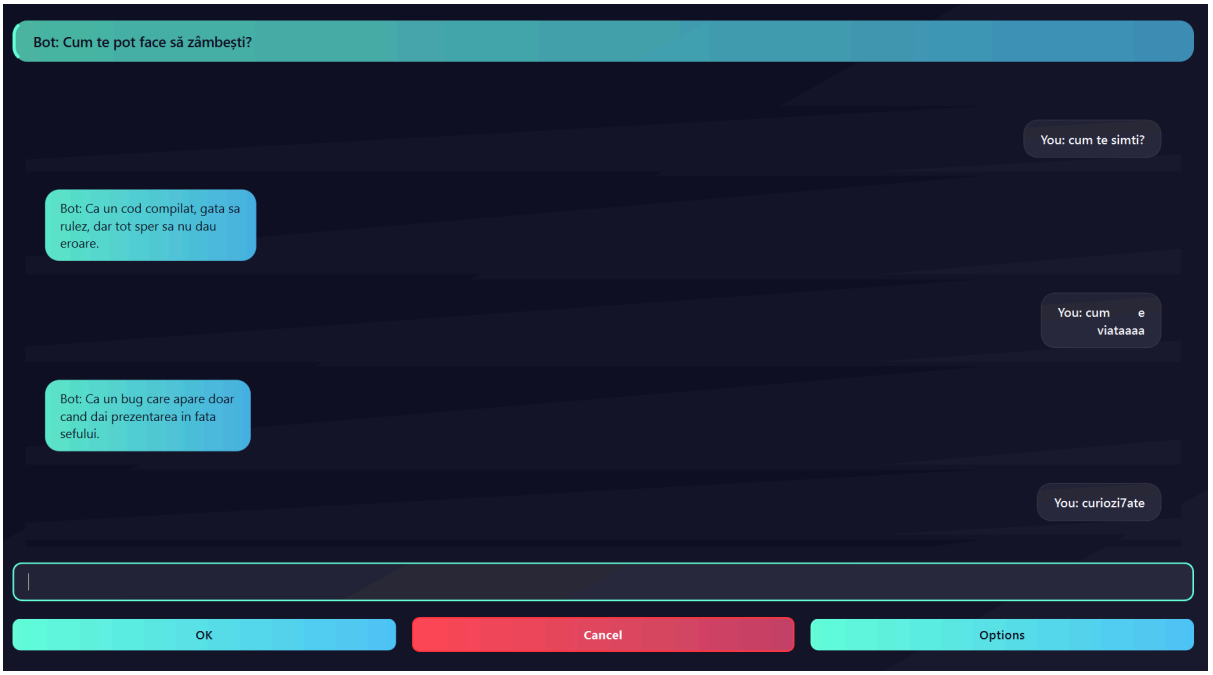
Cum funcționează botul?

- Utilizatorul scrie o întrebare în câmpul de text.
- Când apasă butonul „OK” sau tasta Enter, întrebarea este preluată și curățată (fără semne și spații în plus).
- Botul compară această întrebare cu o listă de întrebări predefinite dintr-un fișier (Q&A.json) folosind o potrivire aproximativă (fuzzy).
- Dacă găsește o întrebare asemănătoare în fișier, botul afișează răspunsul corespunzător.
- Dacă nu găsește, botul spune: „Îmi pare rău, nu știu răspunsul la această întrebare.”
- Mesajele (ale utilizatorului și ale botului) apar în ordine în fereastră, ca într-un chat real.
- Utilizatorul poate apăsa pe „Options” ca să vadă lista de întrebări disponibile.
- Butonul „Cancel” sau tasta Escape închide aplicația.

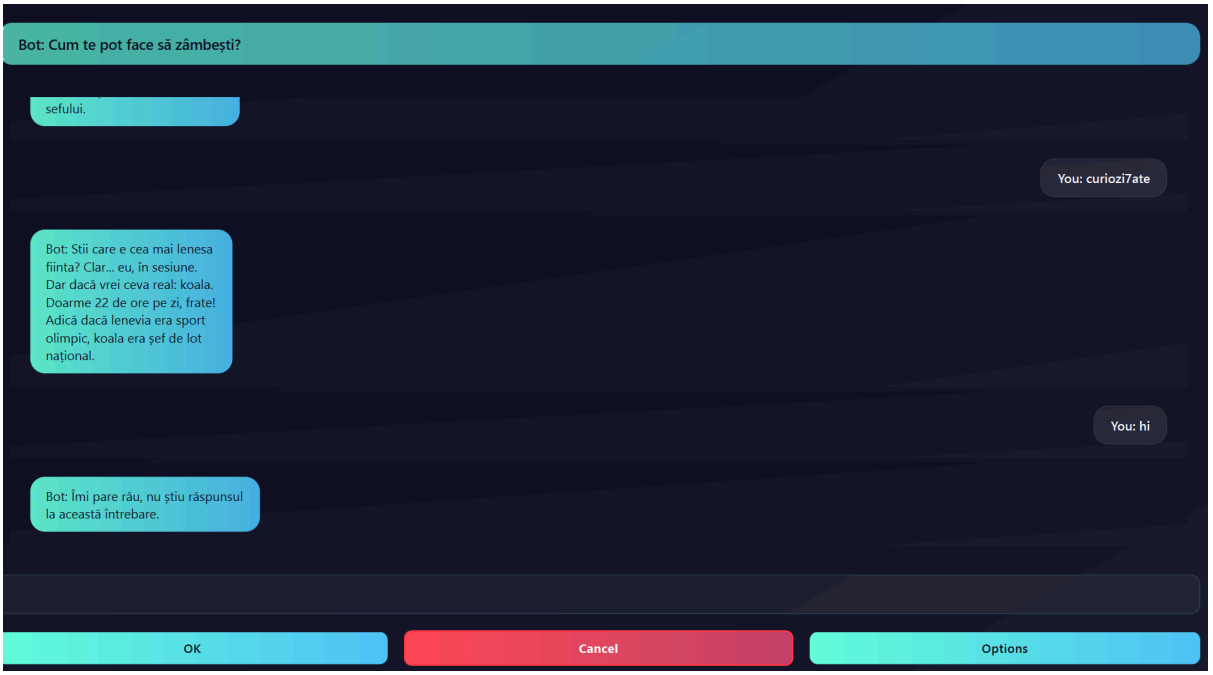
Aceasta este interfata principala a aplicatiei, totul incepe cu un mesaj predefinit al robotului la rulare. Dupa cum putem observa, avem 3 butoane si un spatiu pentru a tasta intrebarile noastre. Butonul ok este pentru a trimite intrebarea noastra ca dupa robotul sa ne raspunda. Butonul de optiuni de va arata cateva exemple de intrebari, iar butonul de cancel opreste rulare si inchide bot-ul.



Aici avem cateva optiuni de intrebari pentru robotul nostru.



Dupa cum se poate observa, robotul raspunde la intrebarile noastre prin raspunsuri predefinite si acesta ignora spati sau caractere tastate gresit si ia cuvantul corect dupa care raspunde.



Daca punem o intrebare care nu se incadreaza in lista de intrebari predefinite, robotul raspunde cu un mesaj predefinit care anunta utilizatorul ca ‘nu stie raspunsul’.