Cybersecurity Awareness Chatbot (Telegram Bot)

An interactive Telegram bot to spread cybersecurity awareness, provide learning resources, and engage users with quizzes and tips.

1. Features

- **Cybersecurity Tips & Awareness:** Sends random best practices to help users stay secure online.
- **FAQ Module:** Answers common questions about firewalls, account security, and more.
- **Interactive Quiz:** Ask users cybersecurity-related multiple-choice questions.
- **Learning Resources:** Recommends free, budget, and paid cybersecurity course platforms.
- **Certification Guidance:** Shares popular certifications like CEH, Security+, CISSP, etc.
- **Programming for Cybersecurity:** Lists important languages for infosec roles.
- **Natural Text Triggers:** Responds to user messages intelligently.
- **Fully Async** and built with python-telegram-bot 20.x.

2. Tech Stack

Component	Description
Python 3	Main programming language
python-telegram-bot	Telegram API wrapper
asyncio + nest_asyncio	To allow async bot logic with polling
Telegram Bot Platform	Frontend for interacting with users

3. Steps to Create a Telegram Bot and Get Its Token:

1. **Open Telegram** (mobile or desktop app).
2. **Search for @BotFather**
   This is the official Telegram bot to create and manage other bots.
3. **Start a chat** and type:
4. **Create a new bot** by typing:
5. **Follow the prompts**:
   - Give your bot a name (can be anything, like CyberAwareBot)
   - Give it a **unique username** (must end in bot, e.g., CyberAwareBot)
6. BotFather will respond with your new bot token

4. Code:

```
import logging
import random
import nest_asyncio
import asyncio
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Application, CommandHandler, MessageHandler,
CallbackQueryHandler, filters, CallbackContext

# Apply nest_asyncio to allow nested event loops
nest_asyncio.apply()

# Logging setup
logging.basicConfig(format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
level=logging.INFO)
logger = logging.getLogger(__name__)

# Bot token
TOKEN =  # Replace with your telegram bot token

# Content for tips, awareness, quizzes, and resources
TIPS = [
    "🔐 Keep your software up-to-date to avoid security vulnerabilities.",
    "🛡️ Use multi-factor authentication wherever possible.",
    "🔑 Don't use weak passwords; opt for long, complex ones.",
    "📧 Be cautious with links in emails or messages.",
    "💾 Back up important files regularly.",
    "🧠 Never reuse passwords across accounts.",
    "📲 Install apps only from trusted sources.",
    "🌐 Use a VPN when browsing on public Wi-Fi."
]

AWARENESS = [
    "🔒 Cybersecurity awareness is crucial in the digital age.",
    "🎣 Beware of phishing! Don't click unknown links or share personal info.",
    "⚠️ Malware can be disguised as legit software—download from trusted sources."
]

QUIZZES = [
    {"question": "What does VPN stand for?",
     "options": ["Virtual Private Network", "Very Private Network", "Virtual Public Network"],
     "answer": "Virtual Private Network"},
    {"question": "What is phishing?",
     "options": ["A type of cyber attack", "A type of fishing", "A form of encryption"],
```

```
    "answer": "A type of cyber attack"}
]

FAQS = [
    {"question": "What is a firewall?",
     "answer": "A firewall monitors and controls network traffic to protect your system."},
    {"question": "How to secure online accounts?",
     "answer": "Use strong passwords, MFA, and keep your software updated."}
]


# Handlers

async def start(update: Update, context: CallbackContext):
    user = update.effective_user
    welcome_text = (
        f"👋 Hello {user.first_name}, welcome to *CyberSecBot*!\n\n"
        "I can assist you with cybersecurity awareness, learning, and fun quizzes.\n\n"
        "*Choose an option below to get started:*"
    )

    # Adding Inline Buttons
    keyboard = [
        [InlineKeyboardButton("📌 Tips", callback_data="tips")],
        [InlineKeyboardButton("🧠 Awareness", callback_data="awareness")],
        [InlineKeyboardButton("📚 Learning", callback_data="learn")],
        [InlineKeyboardButton("🧩 Quiz", callback_data="quiz")],
        [InlineKeyboardButton("❓ FAQ", callback_data="faq")],
        [InlineKeyboardButton("🎓 Certifications", callback_data="certifications")],
        [InlineKeyboardButton("💻 Programming Languages",
callback_data="programming_languages")]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)

    await update.message.reply_text(welcome_text, reply_markup=reply_markup,
parse_mode="Markdown")


async def tips(update: Update, context: CallbackContext):
    await update.message.reply_text(random.choice(TIPS))


async def awareness(update: Update, context: CallbackContext):
    await update.message.reply_text(random.choice(AWARENESS))
async def faq(update: Update, context: CallbackContext):
```

```
    entry = random.choice(FAQS)
    await update.message.reply_text(f"❓ {entry['question']}\n✅ {entry['answer']}")


async def quiz(update: Update, context: CallbackContext):
    q = random.choice(QUIZZES)
    text = f"🧠 *Question:* {q['question']}\n" + "\n".join([f"{i + 1}. {opt}" for i, opt in
enumerate(q['options'])])
    context.user_data['quiz_answer'] = q['answer']
    await update.message.reply_text(text, parse_mode="Markdown")


async def answer_quiz(update: Update, context: CallbackContext):
    if 'quiz_answer' not in context.user_data:
        await update.message.reply_text("Please start a quiz first using `quiz`.")
        return
    user_answer = update.message.text.strip()
    correct = context.user_data['quiz_answer']
    if user_answer.lower() == correct.lower():
        await update.message.reply_text("✅ Correct! Well done.")
    else:
        await update.message.reply_text(f"❌ Wrong! The correct answer was: {correct}")


# Inline learning menu
async def show_learning_options(update: Update, context: CallbackContext):
    query = update.callback_query  # Get the callback query from the update
    await query.answer()  # Acknowledge the callback

    # Send the learning options message as a reply to the callback query
    keyboard = [
        [InlineKeyboardButton("🆓 Free Courses", callback_data="free_courses")],
        [InlineKeyboardButton("💸 Budget-Friendly", callback_data="budget_courses")],
        [InlineKeyboardButton("💰 Paid Platforms", callback_data="paid_courses")]
    ]

    await query.message.reply_text("📚 Choose a category to explore cybersecurity resources:",
 reply_markup=InlineKeyboardMarkup(keyboard))


# Callback function to handle inline buttons
async def button_callback(update: Update, context: CallbackContext):
    query = update.callback_query
    await query.answer()
```

```
    # Check which callback button was pressed and respond accordingly
    if query.data == "free_courses":
        text = (
            "*🆓 Free Courses:*\n"
            "- [TryHackMe](https://tryhackme.com): A hands-on learning platform to teach cybersecurity skills from scratch.\n"
            "- [PortSwigger Academy](https://portswigger.net/web-security): Learn web application security with free resources.\n"
            "- [Coursera (Free audit)](https://www.coursera.org): Free courses in cybersecurity and networking (audit option).\n"
            "- [Cybrary](https://cybrary.it): Cybersecurity learning platform with free resources.\n"
            "- [OpenSecurityTraining](https://opensecuritytraining.info): Free security training resources for beginners to advanced learners."
        )
    elif query.data == "budget_courses":
        text = (
            "*💸 Budget-Friendly:*\n"
            "- [Udemy](https://udemy.com) – Courses priced from $10–$20 during sales, a great deal for foundational cybersecurity topics.\n"
            "- [Skillshare](https://www.skillshare.com) – Offers a free trial for learning cybersecurity skills on a budget."
        )
    elif query.data == "paid_courses":
        text = (
            "*💰 Paid Platforms:*\n"
            "- [LetsDefend](https://letsdefend.io) – Practical blue team labs and SOC tools.\n"
            "- [HTB Academy](https://academy.hackthebox.com) – Offers penetration testing and ethical hacking courses.\n"
            "- [INE](https://ine.com) – Comprehensive platform for cybersecurity training, from beginner to advanced certifications.\n"
            "- [eLearnSecurity](https://elearnsecurity.com) – Offers specialized courses for advanced ethical hackers."
        )
    elif query.data == "certifications":
        text = (
            "*🎓 Cybersecurity Certifications:*\n"
            "- [CompTIA Security+](https://www.comptia.org/certifications/security): An entry-level cert, usually costs around $370.\n"
            "- [CISSP (Certified Information Systems Security Professional)](https://www.isc2.org/cissp): A certification for experienced professionals, costing around $699.\n"
            "- [Certified Ethical Hacker (CEH)](https://www.eccouncil.org/programs/certified-ethical-hacker-ceh/): A certification focused on ethical hacking, usually costing around $1,199.\n"
            "- [AWS Certified Security – Specialty](https://aws.amazon.com/certification/certified-security-specialty/): Cloud security certification by AWS, costing around $300.\n"
            "- [Google Cloud Professional Cloud Security
```

```
Engineer](https://cloud.google.com/certification/cloud-security-engineer): Google Cloud security
cert, costing around $200.\n"
            "- [Microsoft Certified: Azure Security Engineer Associate](https://learn.microsoft.com/en-
us/certifications/azure-security-engineer/): Cloud security certification for Azure, around $165."
        )
    elif query.data == "programming_languages":
        text = (
            "*Useful Programming Languages for Cybersecurity:*\n"
            "- *Python*: Widely used for automation, script writing, and penetration testing. It's easy to
learn and very versatile.\n"
            "- *C/C++*: Important for understanding low-level operations and working with exploits
and vulnerabilities.\n"
            "- *JavaScript*: Useful for understanding web application vulnerabilities, especially related
to XSS and other browser-based attacks.\n"
            "- *Bash/Shell Scripting*: A must for working in Linux environments and automating tasks
during penetration testing.\n"
            "- *Ruby*: Known for the Metasploit framework, Ruby is often used in security tools and
exploit development."
        )
    elif query.data == "tips":
        text = random.choice(TIPS)
    elif query.data == "awareness":
        text = random.choice(AWARENESS)
    elif query.data == "quiz":
        q = random.choice(QUIZZES)
        text = f"🧠 *Question:* {q['question']}\n" + "\n".join([f"{i + 1}. {opt}" for i, opt in
enumerate(q['options'])])
        context.user_data['quiz_answer'] = q['answer']
    elif query.data == "faq":
        entry = random.choice(FAQS)
        text = f"❓ {entry['question']}\n✅ {entry['answer']}"
    elif query.data == "learn":
        return await show_learning_options(update, context)  # reuse learning button UI
    else:
        text = "Option not recognized."

    # Edit the original message to display the response
    await query.edit_message_text(text, parse_mode="Markdown",
disable_web_page_preview=True)

# Text trigger handler
async def handle_text(update: Update, context: CallbackContext):
    text = update.message.text.lower()
    if "tip" in text:
        await tips(update, context)
    elif "aware" in text:
```

```
        await awareness(update, context)
    elif "quiz" in text:
        await quiz(update, context)
    elif "faq" in text or "question" in text:
        await faq(update, context)
    elif "learn" in text or "resource" in text or "course" in text:
        await show_learning_options(update, context)
    elif "cert" in text or "certification" in text:
        await button_callback(update, context)  # Show certifications
    elif "programming" in text:
        await button_callback(update, context)  # Show programming languages
    else:
        await answer_quiz(update, context)  # fallback for quiz answer


# Main bot function
async def main():
    application = Application.builder().token(TOKEN).build()

    # Command handlers
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("tips", tips))
    application.add_handler(CommandHandler("awareness", awareness))
    application.add_handler(CommandHandler("quiz", quiz))
    application.add_handler(CommandHandler("faq", faq))
    application.add_handler(CommandHandler("resources", show_learning_options))

    # Inline callback handler
    application.add_handler(CallbackQueryHandler(button_callback))

    # Text message handler
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_text))

    # Run the bot
    await application.run_polling()

# Run the bot
if __name__ == '__main__':
    asyncio.run(main())
```

5. Output

1. **Install dependencies:**

   ```
   pip install python-telegram-bot nest_asyncio
   ```

2. **Run the bot:**

   ```
   python bot.py
   ```

Sample Interactions

/start

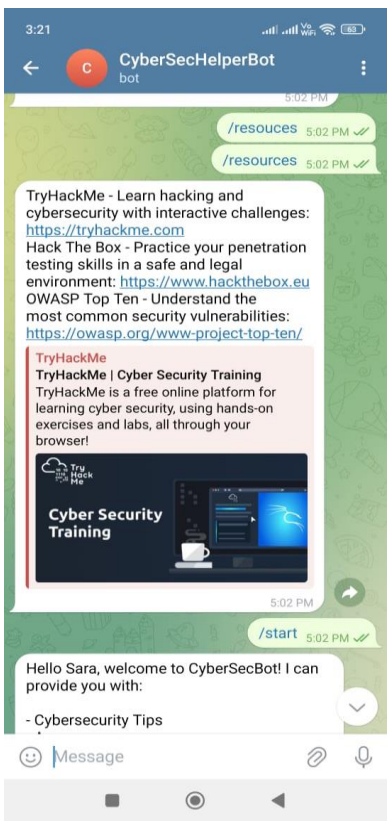
Hello Saranya, welcome to CyberSecBot!

I can assist you with cybersecurity awareness, learning, and fun quizzes.

Choose an option:

- Tips
- Awareness
- Quiz
- FAQ
- Learning
- Certifications
- Programming Languages

6. Screenshots



7. Summary

This bot features interactive buttons, intelligent responses to text input, and an async architecture using python-telegram-bot. Ideal for educating users about digital safety in a conversational format.