Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем і технологій

Лабораторна робота №3

з дисципліни «Основи WEB-технологій»

Тема: «Створення телеграм бота з меню та запитом до ChatGPT»

Виконав:
студент групи ІА-11
Воробей Антон Олегович

Перевірив:
Альбрехт Й. О.

Київ 2024

Завдання:

Створити telegram-бот з меню та задеплоїти його на сервісі https://pythonanywhere.com/.

Структура меню:

- Студент (прізвище, група)
- IT-технології (….)
- Контакти (тел., e-mail)
- Prompt ChatGPT

Посилання на вихідний код проекта: https://github.com/tonujet/WEB-basics/tree/main/Lab3

Хід роботи

Для виконання даної лабораторної роботи я скористаюся мовою Rust

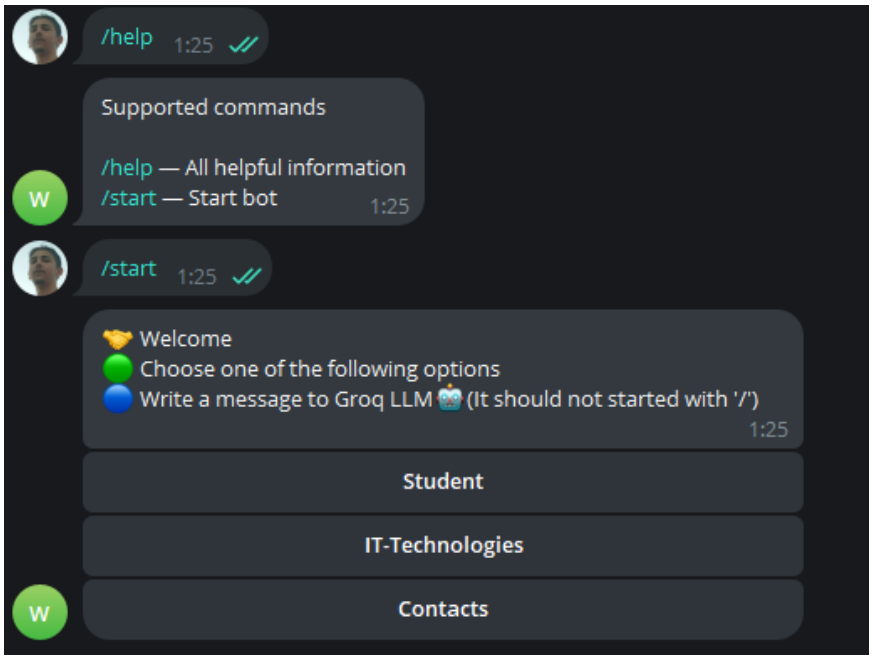
Створимо телеграм бота за допомогою бібліотеки teloxide

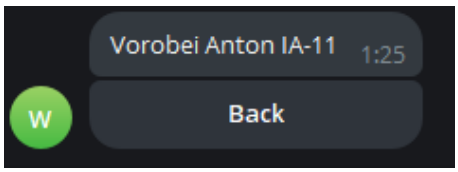
```rust
#[derive(Default, Debug, Clone)]
pub struct State {
    last_menu_reply: Arc<RwLock<HashMap<UserId, MessageId>>>,
    http_client: reqwest::Client,
}

#[tokio::main]
async fn main() -> Result<(), Box<dyn Error>> {
    pretty_env_logger::init();
    log::info!("Starting buttons bot...");

    let bot = Bot::new(config().TELEGRAM.TOKEN.as_str());
    let state = State::default();

    let handler = dptree::entry()
        .branch(Update::filter_message().endpoint(message_handler))
        .branch(Update::filter_callback_query().endpoint(callback_handler));

    Dispatcher::builder(bot, handler)
        .dependencies(dptree::deps![state])
        .enable_ctrlc_handler()
        .build()
        .dispatch()
        .await;
    Ok(())
}

async fn message_handler(
    bot: Bot,
    msg: Message,
    me: Me,
```

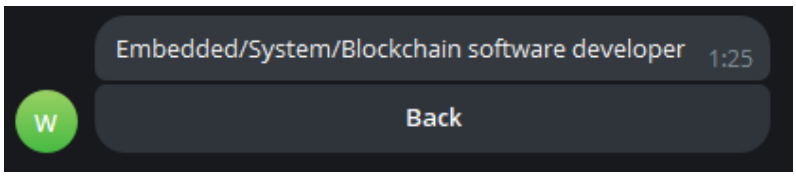
```
    state: State,
) -> Result<(), Box<dyn Error + Send + Sync>> {
    if msg.text().is_none() {
        return Ok(());
    }

    let text = msg.text().unwrap();
    let is_menu_sent = { state.last_menu_reply.read().await.contains_key(&me.user.id)
};

    if is_menu_sent {
        let msg_id = state
            .last_menu_reply
            .write()
            .await
            .remove(&me.user.id)
            .unwrap();
        bot.delete_message(msg.chat.id, msg_id).await?;
    }

    if !is_menu_sent || text.starts_with('/') {
        command_handler(bot.clone(), msg.clone(), me.clone(), state.clone()).await?;
    } else {
        let groq_reply = send_prompt_to_groq(text, state.clone()).await?;
        let msg = bot
            .parse_mode(ParseMode::Html)
            .send_message(msg.chat.id, format!("<b>Groq reply🤘</b>
\n\n{groq_reply}"))
            .reply_markup(keyboard::back_keyboard())
            .await?;

        state.last_menu_reply.write().await.insert(me.id, msg.id);
    }
    Ok(())
}

// **IMPORTANT**: do not send privacy-sensitive data this way!!!
// Anyone can read data stored in the callback button.
async fn callback_handler(
    bot: Bot,
    q: CallbackQuery,
    state: State,
) -> Result<(), Box<dyn Error + Send + Sync>> {
    if q.data.is_none() {
        return Ok(());
    }

    bot.answer_callback_query(q.id.clone()).await?;
    let cb_data = q.data.clone().unwrap();
    let button = KeyboardButton::from(cb_data.as_str());
    button.press(bot.clone(), q, state).await
}



#[derive(BotCommands)]
#[command(
    rename_rule = "lowercase",
    description = "Supported commands"
)]
enum Command {
    #[command(description = "All helpful information")]
    Help,
    #[command(description = "Start bot")]
```

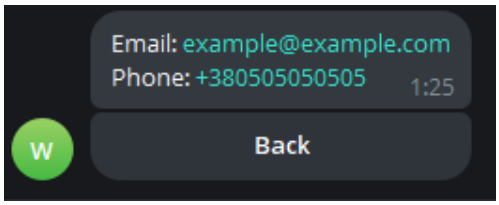
```
    Start,
}

async fn command_handler(bot: Bot, msg: Message, me: Me, state: State) -> Result<(),
Box<dyn Error + Send + Sync>>{
    if msg.text().is_none() {
        return Ok(());
    }
    let text = msg.text().unwrap();

    match BotCommands::parse(text, me.username()) {
        Ok(Command::Help) => {
            bot.send_message(msg.chat.id, Command::descriptions().to_string())
                .await?;
        }
        Ok(Command::Start) => {
            let msg = bot.send_message(
                msg.chat.id,
                "🤝 Welcome \n🟢 Choose one of the following options \n🔵 Write a
message to Groq LLM🤖(It should not started with '/')",
            )
                .reply_markup(keyboard::menu_keyboard())
                .await?;
            state
                .last_menu_reply
                .write()
                .await
                .insert(me.user.id, msg.id);
        }
        Err(_) => {
            bot.send_message(msg.chat.id, "Command not found!").await?;
        }
    }

    Ok(())
}
```

Перевіримо роботу бота зі статичною інформацією

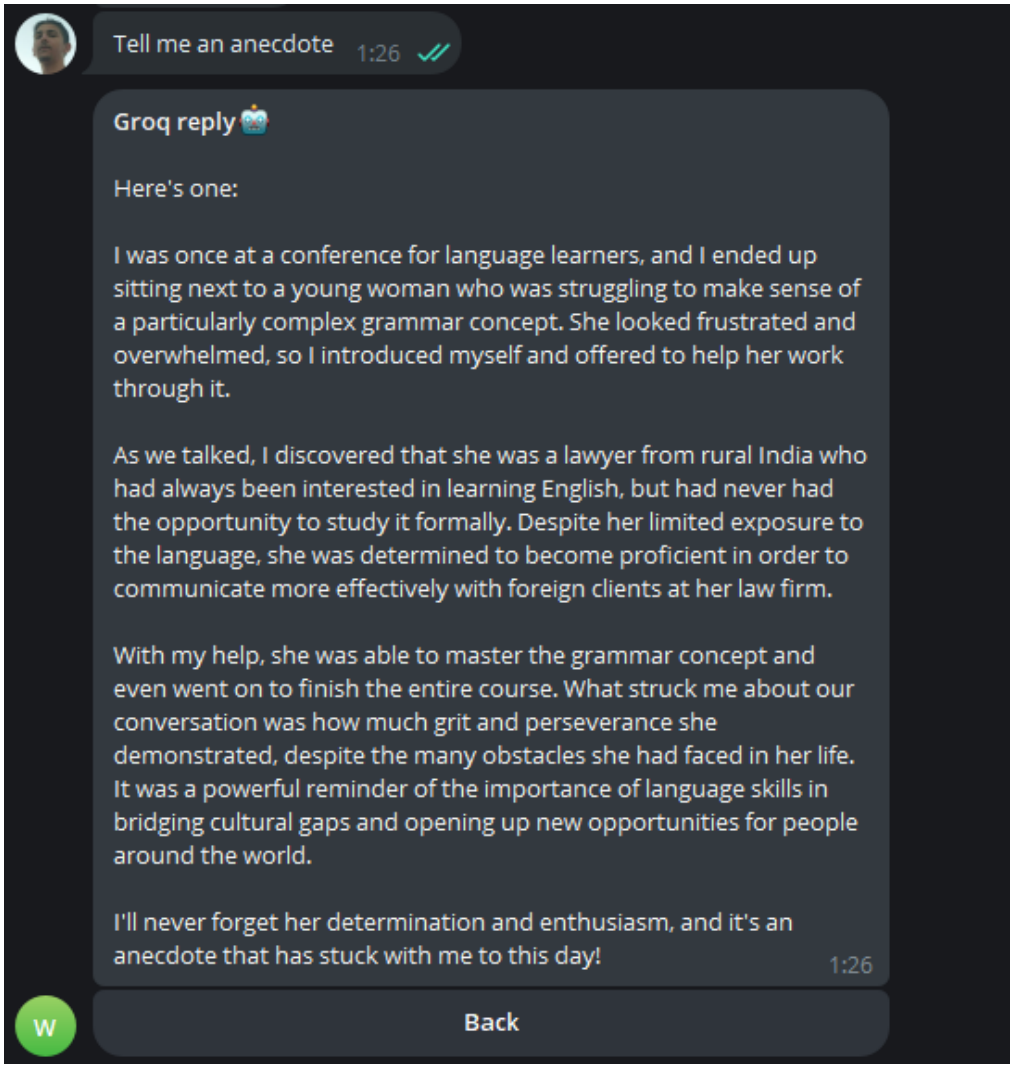








Перевіримо роботу бота зі звернення до Groq LLM



Висновок: На цій лабораторні роботі я навчився створювати телеграм ботів та створив свого власного у відповідності до завдання, що надав викладач