Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

Факультет математики та інформатики
(повна назва інституту/факультету)

Кафедра математичного моделювання
(повна назва кафедри)

Створення Telegram-бота для допомоги студенту вищого навчального закладу

Кваліфікаційна робота
Рівень вищої освіти - другий (магістерський)

Виконав :
студент 6 курсу, групи 607
спеціальності 124 – Системний аналіз
(назва спеціальності)
Чорноус Олександр Віталійович
(прізвище, ім'я та по-батькові)
Керівник доцент, канд. фіз.-мат. наук,
Юрченко І.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

До захисту допущено:
Протокол засідання кафедри № __
від „___" грудня 2021 р.
зав. кафедри _____ проф. Черевко І.М.

Чернівці – 2021

ЗМІСТ

Анотація

Дипломна робота присвячена ознайомленню та розробці телеграм бота за допомогою мови Python.

Вступ

Магістерська робота присвячена розробці Telegram-боту для допомоги студентам в навчанні та проведенні вільного часу.

Студент який тільки потрапляє в новий та цікавий період студенства губиться у вирі подій, мій бот розрахований на те, щоб полегшити адаптацію студента до нової ланки його життя. Тому в моєму боті передбачений такий функціонал як: розташування кафедр, можливість оплати за гуртожиток та університет, Telegram-канали з потрібною інформацією, новини з Instagram - каналу та сайту університету.

Друга частина бота - це вільний час студента та різні варіанти як він його може провести. В моєму боті передбачені такі функції: Вибір фільма та серіалу, за яким можна провести вечір, вибір з топ 400 бестселерів книжкового світу, та за можливою співпрацею з різними закладами для молоді, щоб сповіщати їх у разі якихось нових заходів приурочених святам.

Також в боті є функція гри, яка об'єднує в собі обидва пункти, і проведення вільного часу і допомога в навчанні. “Гра в питання” - це проста гра в питання-відповіді де студент може вибрати цікаву для нього тему, пройтись по питанням та підготуватись до тестувань чи екзаменів.

Розділ І Мова програмування Python. Мессенджер Telegram

1.1 Python

Python — найшвидше зростаюча мова програмування за останні кілька років. Про це свідчить дослідження система питань відповідей StackOverflow за 2019 рік.

Python підтримує структуроване, об'єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване програмування. Що дає розробнику місце для маневру. Основними архітектурними особливостями є динамічна типізація, автоматичне керування пам'яттю, повний самоаналіз, механізм обробки винятків, підтримка багатопоточних обчислень та обробка винятків високого рівня. Підтримка розподілу програм за модулями, які можна об'єднувати в пакети[1].

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування, метапрограмування та функціональне програмування. Завдання узагальненого програмування вирішуються рахунок динамічної типизації. Аспектно-орієнтоване програмування частково підтримується через декоратори, повноцінна підтримка забезпечується додатковими фреймворками. Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень. Основні архітектурні риси - динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопотокових обчислень з глобальним

[1] https://www.python.org/

блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які, своєю чергою, можуть поєднуватися в пакети.

1.2 Telegram

Telegram - це мультиплатформенний додаток(програма) на Android, IOS, Windows та інші операційні системи, основним функціоналом якого є спілкування в мережі, обмін мультимедійними даними, з недавнього часу, створення онлайн конференцій по аналогу Meet чи Zoom.

Додаток був створений російським програмістом та підприємцем Павлом Дуровим, розробником соціальної мережі ВКонтаткі.

На даний момент цей месенджер є одним з найпопулярніших месенджерів, топ-2 в категорії найбільш захищених месенджерів, та входить в топ-5 найбільш популярних месенджерів світу.

Цей месенджер я вибрав по декільком причинам:

1. Моє активне використання його в повсякденному житті і відповідно, знання його переваг та недоліків;
2. Прив’язка основних чатів БДМУ до цього месенджера;
3. Доступність до розробки ботів та обширний функціонал;

1.3 Боти та їх використання

Бот(програма) - спеціальна програма, що виконує автоматично та/або за заданим розкладом будь-які дії через інтерфейси, призначені для людей.

Приклади ботів можна зустріти як корисні програми: автовідповідач техпідтримки, бот довідник, захисний бот від спаму і подібні їм програми. А також є нечесні(зловмисні) програми-боти: спамери, DDOS, боти-скупщики, до негативних також можна додати ботів, які створюють видимість активності на трансляціях та в коментарях до статей чи відео.

Також до ботів можна віднести програми в комп'ютерних іграх які імітують поведінку людей(гравців) або істот запрограмованих допомагати чи заважати гравцеві просуватись по сюжету гри чи створювати перешкоди в багатокористувацьких режимах.

В магістерській описується тип боту - бот-довідник, з додатковими функціями для допомоги студенту адаптуватись до нової обстановки, в першу чергу бот розрахований на першокурсників, або студентів які тільки недавно перейшли в університет.

Для створення Telegram-bot, потрібно його зареєструвати в @BotFather – це бот який дозволяє створювати та реєструвати ботів на сервері Telegram. Йому потрібно написати повідомлення та вибрати відповідні пункти в меню, після успішної реєстрації видається токен для підключення до системи бота. Токен – це унікальний набір символів для підключення своїх налаштувань(коду) до серверів на яких знаходиться сам бот. Оскільки токен використовується для підключення до боту, то його не варто розповсюджувати з міркувань безпеки.

Розділ II Бібліотеки та функції @BDMU_bot

2.1 Функції та фішки бота

Початок роботи з ботом для користувача виглядає як привітання та прохання до користувача зареєструватись в боті.(Додаток 2.1, Рис. 1)

Після введення своєї пошти, йде перевірка користувачем на правильність введення, якщо все вірно то потрібно натиснути кнопку «Так», ваша адреса підв'яжиться до вашого TelegramId, в протилежному випадку натиснувши кнопку «Ні», бот попросить вас повторити спробу(Додаток 2.1, Рис. 2)[2]

Дані зберігаються в базі даних (Додаток 2.1, Рис. 3)

Також в боті є функція підписки, як перевіряє чи даний користувач підписаний на бота, підписка перевіряється в реальному часі (Додаток 2.1, Рис. 4)

Так виглядає база підписок (Додаток 2.1, Рис. 5)

Після реєстрації відкривається доступ до основних функцій бота які представлені таким меню кнопок (Додаток 2.1, Рис. 6)

Телеграм бот БДМУ має такі функції при використанні меню кнопок:

1. Актуальні новини з Instagram каналу університету та сторінка новин з офіційного сайту університету (Додаток 2.1, Рис. 7)
2. Інформація щодо теперішнього стану університету в період спалаху вірусу COVID-19 (Додаток 2.1, Рис. 8)

[2] https://www.youtube.com/watch?v=M8fhrtvedHA&list=PLwsbEQtav8RGTMuZTBcWLbFBggEbeqRd1&index=3

3. При натисканні на кнопку «Free time», з’являються рекомендації щодо проведення вільного часу студента які поділяються на (Рис. 9)
    a. «Фільм на вечір» та «Серіал на вечір» - Рекомендації щодо перегляду фільму та серіалу. База даних кінематографу підтягується автоматично з сайту всесвітнього відомого рейтингу IMDb за допомогою технології API, бот дає тільки рекомендацію щодо вибору фільму, API сайту IMDb є у відкритому доступу. За допомогою цієї технології підтягується актуальний топ-250 серіалів та фільмів\мультфільмів, який постійно оновлюється (Додаток 2.1, Рис. 10 \ 11)
    b. «Книга на вечір» - Рекомендації щодо вибору художньої та навчальної літератури, для проведення часу за книгою. Перелік літератури береться зі створеної мною бази даних, всі перечисленні автори з їхніми книгами взяті з топу найкращих книг світу який можна подивитись на сайті вікіпедії за відповідним запитом(Додаток 2.1, Рис. 12 \ 13)
    c. Пункт «Куди сходити вечором» - буде доповнюватись інформацією щодо проведення заходів для студентів, таких як тематичні зустрічі, вечірки або якісь акції для студентів. Дається коротка інформація та посилання, або картинка з афішою (Додаток 2.1, Рис. 14)
    d. Пункт «Гра в питання» - це одна з функцій Telegram яку вони ввели в 2016 році, ця функція дозволяє у внутрішньому браузері відкривати ігри написані за допомогою Web-технологій. У боті працює гра в питання (Додаток 2.1, Рис. 15) – це гра в якій на картці з’являється питання та студент має 3 секунди до то го моменту як на цій картці з’явиться відповідь на поставлене питання (Додаток 2.1, Рис. 16 / 17)
4. Кнопка «Для студента» має такий перелік функцій: (Додаток 2.1, Рис. 18)

a. «Журнал оцінок» - це кнопка повинна була відповідати за швидкий доступ до інформації ЕЖ(Електронний журнал БДМУ), перелік пропусків та список оцінок. Працювати все повинно було через eBSMU API – яка працює з ЕЖ по аналогу з API Instagram чи IMDb. Але через проблеми з ЕЖ ці функції не вдалось реалізувати. На даний момент дається тільки посилання на сайт ЕЖ де можна авторизуватись та працювати на сайті.
b. «Корпуси» - це перелік факультетів та кафедр (Додаток 2.1, Рис. 19 \ 20) з їхніми веб-сайтами(якщо вони існують), тут можна подивитись та скласти маршрут до кафедри на якій проводяться заняття або до гуртожитків. Якщо студенту потрібно створити маршрути до певної кафедри, він вибирає потрібний факультет, на цьому факультеті кафедру і йому виводиться інформація у вигляді посилання на сайт кафедри та мітка на карті, маршрут можна відкрити у любому зручному додатку на телефоні(наприклад Google Maps), (Додаток 2.1, Рис. 21). Те саме стосується і гуртожитків (Додаток 2.1, Рис. 22 \ 23).
c. «Розклад» - представлений посилання на сторінку офіційного сайту університету де розміщенні посилання на розклад студентів (Додаток 2.1, Рис. 24).
d. «Оплата за університет» - представлена кнопкою з посилання на сторінку кабінету університету в Приват банку, де студенти можуть здійснити оплату за послу чи навчання(Додаток 2.1, Рис. 25). У зв’язку з закриттям Приват банком функції оплати через ботів в месенджерах, робота по оплаті тепер здійснюється тільки таким чином.

2.2 Бібліотеки Python для роботи з ботом

Повномірний код представлений в Додатоку 2.2.

В магістерській роботі використовуються такі бібліотеки Python як:

1. Telebot – це бібліотека для Python, яка має повний функціонал для роботи з месенджером Telegram, тобто за допомогою цієї бібліотеки можна користуватись усіма функціями, такими як: надсилання фото, відео, аудіо файлів. Надсилати повідомлення в месенджер відповідати на запитання, якщо бот запрограмований на такі дії. Розпізнавання тексту чи фото, функціонал дуже обширний для роботи з цією бібліотекою.
2. Random – бібліотека, яка допомагає генерувати випадкові числа, вона використовується у роботі з API IMDb та в роботі базою даних для генерації випадкової id книги[3].
3. sqlightDB – бібліотека яка використовується для роботи з базою даних SQLight в мові програмуванні Python, з її допомогою можна повноцінно працювати із запитами до бази даних.
4. InstagramAPI – це бібліотека, яка дозволяє отримати доступ до API Instagram, брати інформацію про пости, про користувачів,
5. Datetime – бібліотека яка дозволяє використовувати класи по маніпулюванню датами та часом, вона використовується для моніторингу активності користувача, вона веде облік останньої активності користувача, та записує дату та час відправлення його останнього повідомлення. Хоча арифметика дати й часу підтримується, основна увага приділяється ефективному витягу атрибутів для форматування та маніпулювання результатами[4].

[3] https://docs.python.org/3/library/random.html
[4] https://docs.python.org/3/library/datetime.html

6. IMDb – це бібліотека, яка дозволяє працювати з API сайту IMDb, з якого запитуються дані для кнопок «Кіно на вечір» та «Серіал на вечір», окрім доступу до топ-рейтингу сайту, вона дозволяє підтягувати по API велику кількість даних, про фільм, про автора, про акторів, про режисерів та іншу інформацію що стосується якимось чином кіноіндустрії.

2.3 Розбір функцій

Використання кнопок та клавіатури

У магістерській роботі частіше всього використовувалась функція розпізнавання тексту, тобто: створюється меню кнопок, в яке додається певний текст,

```
btn_stud_kaf = telebot.types.KeyboardButton("Корпуси")
btn_stud_tmtb = telebot.types.KeyboardButton("Розклад")
btn_stud_jrl = telebot.types.KeyboardButton("Журнал оцінок")
btn_stud_std = telebot.types.KeyboardButton("Оплата за університет")
```

потім додається умова, що під час написання повідомлення бот на певну фразу повинен виконувати запрограмовану в нього дію: надсилати повідомлення, передавати посилання, відкривати клавіатуру кнопок, відкривати внутрішню клавіатуру, звертатись до бази даних чи інші доступні функції.

Наприклад при натисканні на кнопку «Корпуси» боту в чат надсилається повідомлення з текстом «Корпуси», реакцією на текст буде виклик меню-клавіатури та відправлення повідомлення з текстом: «Виберіть локацію яка вас цікавить»[5].

```
elif (message.text == "Корпуси"):
  bot.send_message(message.chat.id, text = 'Виберіть локацію яка вас
цікавить', reply_markup = (markup_loc_menu))
```

Інший приклад, коли під час реакції на повідомлення відкривається не меню текстових команд, а відкривається внутрішня клавіатура. При

[5] https://github.com/eternnoir/pyTelegramBotAPI

натисканні на меню-клавіатурі кнопки «Гуртожитки» в чат надсилається відповідне повідомлення з текстом кнопки та бот відкриває внутрішню клавіатуру в якій знаходиться список гуртожитків у вигляді інтерактивних кнопок (Додаток 2.1 Рис. 22), нижче описаний код виклику так званої inline-клавіатури - це клавіатура що розташовується в середині чата, після натискання на якусь з запропонованих кнопок, бот зчитує яку саме кнопку було натиснуто, звертається до бази даних передаючи потрібний id для запиту, база даних, в цьому випадку, передає такі параметри: назву гуртожитку та розташування його на карті передаючи координати(широту та довготу).

```
@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_hostel_1','loc_btn_hostel_2','loc_btn_hostel_3','loc_btn_hostel_4',
'loc_btn_hostel_5','loc_btn_hostel_6','loc_btn_hostel_7','loc_btn_hostel_8',]
)
def inline_location_hostels(call):
  try:
    if call.message:
      if call.data == 'loc_btn_hostel_1':
        lat = db.get_location_lat(id_loc = 4)
        long = db.get_location_long(id_loc = 4)
        name = db.get_location_name(id_loc = 4)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
```

Bot.send_message – це функція надсилання повідомлення в якій передається текст з бази даних в повідомлення для користувача. [6]

Bot.send_location – це функція передачі геоданих, в ній я повинен передати два параметри з бази широту та довготу, тому в коді виклик даних записаний двома окремими змінними **long** та **lat.** Після отримання даних, бот надсилає кінцевий результат запиту до користувача в чат, у вигляді повідомлення та посилання на локацію. Після закінчення роботи з внутрішньою клавіатурою вона ховається.

```
bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
```

За допомогою цього рядка, приховується внутрішню клавіатуру після відправки повідомлення в чат з її допомогою. **reply_markup** - відповідає за

[6] https://github.com/eternnoir/pyTelegramBotAPI

виклик чи приховання клавіатури, в лапках записується посилання на змінну яка відповідає за формування клавіатури, якщо це місце залишити пустим, то відповідно клавіатура буде прихованою.

В магістерській роботі клавіатури формуються заздалегідь таким кодом:

```
markup_main = telebot.types.ReplyKeyboardMarkup(resize_keyboard =
True).row(btn_news, btn_cov).row(btn_fr_tm, btn_stud)
```

Спочатку ми записуємо назву клавіатури, додаємо тип клавіатури, **ReplyKeyboardMarkup** відповідає за меню-клавіатуру, яку надсилає текстові команди в чат. За допомогою параметру **resize_keyboard = True** звичайна клавіатур перетворюється в адаптивну, яка підлаштовується під екран телефону або комп'ютера **row** відповідає за кількість кнопок в рядку, в прикладі наведеному вище в кожному рядку знаходиться по дві кнопки, також за допомогою **add** додається не рядок з кнопка, а один елемент. В прикладі наведеному нижче меню кнопок складається з рядків та окремих кнопок – це меню «Free time»[7].

```
markup_frtm = telebot.types.ReplyKeyboardMarkup(resize_keyboard =
True,).row(free_button1,
free_button2).add(free_button5).row(free_button3,free_button4).add(return_mai
n)
```

Розглянемо інший вид меню кнопок – це внутрішня клавіатура параметри, так звана **InlineKeyboard**. Цей тип клавіатури працює як звичайна кнопка, ніякий текст не відправляється в чат як у попередньому виді кнопок, ця клавіатура виникає у чаті у вигляді меню яке складається із запрограмованої кількості кнопок, в прикладі нижче це меню буде мати рядок з двох кнопок «Так» та «Ні», вони використовуються для підтвердження правильності введеної адреси користувачем для реєстрації в боті.

```
markup_YN = telebot.types.InlineKeyboardMarkup().row(inline_btn_Y,
inline_btn_N)
```

Для кнопок в цьому меню з'являється ще один параметр для прив'язки дії до неї, якщо в попередньому виді кнопок використовувалась назва кнопки

[7] https://github.com/eternnoir/pyTelegramBotAPI

яка відправлялась в чат для реакції на неї, то в цьому випадку при натисканні на кнопку відправляється запит до бота на реакцію яка прив'язана до кнопки.

```
inline_btn_Y = telebot.types.InlineKeyboardButton("Так",
callback_data="button_Y")
inline_btn_N = telebot.types.InlineKeyboardButton("Ні",
callback_data="button_N")
```

Приклад використання кнопок «Так» та «Ні»[8]

```
@bot.callback_query_handler(func = lambda call: call.data in ['button_Y',
'button_N'])
def email_accept(call):
  try:
    if call.message:
      if call.data == 'button_Y':
        bot.send_message(call.message.chat.id, "Пошта підтверджена.")
        db.update_email(call.message.chat.id, email = call.message.text)
      elif call.data == 'button_N':
        bot.send_message(call.message.chat.id, "Будь ласка введіть свою
університетську пошту:")
        bot.register_next_step_handler_by_chat_id(call.message.chat.id,
email)
    bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup='')
  except Exception as e:
    print(repr(e))
```

Бот перевіряє де використовується **callback_data** кнопок клавіатури і звертається до потрібної функції. В першому рядку йде перевірка чи дана функція стосується клавіатури(кнопки), якщо стосується, то виконуються відповідно запрограмовані команди. В даному випадку, йде перевірка пошти, користувачу надсилається повідомлення з введеною ним поштую якщо він погоджується з тим що пошта введена правильно, то пошта приписується до його TelegramID, якщо він помічає помилку при введені, він натискає кнопку «Ні» і йому пропонується заново ввести пошту, після введення алгоритм повторюється. Після закінчення дій клавіатура приховується. Приховання клавіатури виконується з двох причин, перша причина – це зайва інформація яка засмічує чат з ботом, друга – це якщо користувач багато раз буде натиска на кнопки клавіатури буде виникати збій у роботі бота, який приведе до його вимкнення. Функції внутрішньої клавіатури найчастіше зустрічаються у роботі з геоданими: факультети, кафедри та гуртожитки, виводяться з її допомогою.

[8] https://www.youtube.com/watch?v=M8fhrtvedHA&list=PLwsbEQtav8RGTMuZTBcWLbFBggEbeqRd1&index=3

Розділ ІІІ Використання SQLiteDB

3.1 SQLiteDB

SQLite — це бібліотека в процесі роботи, яка реалізує автономний, безсерверний, транзакційний механізм баз даних **SQL** з нульовою конфігурацією. Код для **SQLite** є загальнодоступним і, таким чином, безкоштовний для будь-яких цілей, комерційних чи приватних. **SQLite** — це найпоширеніша база даних у світі з більшою кількістю додатків, ніж ми можемо розрахувати, включно з кількома високопрофільними проектами[9].

SQLite — це вбудований механізм баз даних **SQL**. На відміну від більшості інших баз даних **SQL**, **SQLite** не має окремого серверного процесу. **SQLite** читає та записує безпосередньо на звичайні дискові носії. Повна база даних **SQL** з кількома таблицями, індексами, тригерами та представленнями міститься в одному файлі на диску. Формат файлу бази даних є кросплатформним – ви можете вільно копіювати базу даних між 32-розрядними та 64-розрядними системами або між архітектурами великого та малого порядків. Ці функції роблять **SQLite** популярним вибором як формат файлу програми.

SQLite - це компактна бібліотека. З увімкненими всіма функціями розмір бібліотеки може бути менше 600 КБ, залежно від цільової платформи та налаштувань оптимізації компілятора. (64-розрядний код більший. І деякі оптимізації компілятора, такі як агресивне вбудовування функцій і розгортання циклу, можуть призвести до значного збільшення об'єктного коду.) Існує компроміс між використанням пам'яті та швидкістю. **SQLite**, як правило, працює швидше, чим більше пам’яті ви даєте йому. Тим не менш, продуктивність зазвичай досить хороша навіть в середовищі з низьким рівнем

[9] https://www.sqlite.org/about.html

пам’яті. Залежно від того, як він використовується, **SQLite** може бути швидшим, ніж прямий ввід-вивід файлової системи.

В проекті, ця база даних використовується для реєстрації та моніторингу користувачі чат-боту, для зберігання інформації що стосується користувачів(**user_id**, остання дата активності, електронна адреса), також використовується для ведення бази книг та бази розташування кафедр.

3.2 Використання SQLiteDB для реєстрації користувача в боті

У проекті база **SQLite** використовується як база даних користувачів. Підключення до бази виконується за допомогою підключення через файл.[10]

Ця частину коду відповідає за вибір файлу в якому знаходиться база даних

```
db = SQLighter('db.db')
```

звертаємось до класу, в іншому файлі, в якому знаходяться всі запити до бази даних SQLighter – це клас із усіма запитаними.

```
def __init__(self, database_file):
  """Підключаємось до бази даних"""
  self.connection = sqlite3.connect(database_file, check_same_thread = False)
  self.cursor = self.connection.cursor()
```

Для роботи з базою користувача використовуються такі запити:

```
def get_subscriptions(self, status = True):
  """Отримуємо активних користувачів"""
  with self.connection:
    return self.cursor.execute("SELECT * FROM `subscriptions` WHERE `status`
= ?", (status,)).fetchall()
```

В цьому запиті проводиться перевірка статусу користувача, отримуючи певне значення (**True or False**), користувача повідомляється чи є в нього підписка на канал чи немає, і що потрібно зробити для того щоб її підключити.

```
def subscriber_exists(self, user_id):
  """Перевіряємо чи є юзер в базі"""
  with self.connection:
    result = self.cursor.execute("SELECT * FROM `subscriptions` WHERE
`user_id` = ?", (user_id,)).fetchall()
    return bool(len(result))
```

[10] https://www.youtube.com/watch?v=M8fhrtvedHA&list=PLwsbEQtav8RGTMuZTBcWLbFBggEbeqRd1&index=3

Відбувається додавання користувача в базу, в цьому випадку просто відбувається відмітка користувача що він хоч раз підключався до бота.

```
def add_subscriber(self, user_id, status = True, sub_date =
datetime.today()):
  """Додаємо нового користувача"""
  with self.connection:
    return self.cursor.execute("INSERT INTO `subscriptions` (`user_id`,
'status','sub_date') VALUES(?,?,?)",(user_id, status, sub_date,))
```

Через цей запит вже відбувається реєстрація користувача, і в базу передається 3 параметри: **user_id** – що відповідає за мітку користувача в телеграмі, **status** – показує чи користувач підписаний на канал та **sub_date** – що відповідає за дату підписки.

```
def update_subscription(self, user_id, status):
  """Оновлюємо статус підкписки користувача"""
  with self.connection:
    return self.cursor.execute("UPDATE `subscriptions` SET `status` = ? WHERE
`user_id` = ?", (status, user_id,))
```

Цей запит відповідає за оновлення статусу підписки в базі, при натисканні кнопки підписки, статус змінюється цей параметр дозволить відслідковувати чи надсилати цьому користувачу новини та інші оновлення.

```
def update_email (self, id_us, email, last_date= datetime.today()):
  """Додаємо пошту користувача в базу"""
  with self.connection:
    return self.cursor.execute("INSERT INTO `user_data` (email, id_us,
last_date)VALUES(?,?,?)", (email, id_us, last_date))
```

Тут відбувається прив’язка електронної пошти користувача до його **user_id** в телеграмі.

```
def update_date (self, id_us, last_date):
  """Перевріям коли користувач останній раз заходив"""
  with self.connection:
    return self.cursor.execute("UPDATE user_data SET last_date = ? WHERE
id_us = ?",( last_date, id_us,))
```

За допомогою цього запиту відбувається моніторинг активності користувача, таким чином можна відслідковувати неактивних користувачі та видаляти їх з бази, якщо їхня остання активність в чаті була дуже давно, або можна відправляти повідомлення з нагадуванням про те що вони не були активні довгий період часу.

```
def email_exists(self, user_id):
  """Перевіряємо чи є користувач в базі"""
  with self.connection:
    result = self.cursor.execute("SELECT * FROM `subscriptions` WHERE
```

```
`user_id` = ?", (user_id,)).fetchall()
    return bool(len(result))
```

Тут відбувається перевірка чи користувач вже є в базі, щоб не було створено багато однакових записів в базі.

Такий функціонал передбачений для роботи з користувачами та моніторингу їх активності.

3.3 Використання SQLiteDB для передавання інформації в функції бота

Також, база використовується для ведення інформації для пункту «Корпуси» який показує розташування кафедр, гуртожитків та основних корпусів університету. Та для пункту «Книга на вечір», який відповідає за базу даних художньої літератури.

Для цього використовуються та запити до бази даних:

```
def get_location_lat (self, id_loc):
  """Витягуємо широту"""
  with self.connection:
    return self.cursor.execute("SELECT latitude FROM locations WHERE id_loc =
? ", (id_loc,)).fetchone()

def get_location_long (self, id_loc):
  """витягуємо довготу"""
  with self.connection:
    return self.cursor.execute("SELECT longitude FROM locations WHERE id_loc
= ?", (id_loc,)).fetchone()
```

Тут описується запит, що відповідає за отримання даних геолокації, вибраних користувачем в меню кнопок, корпусів. Запити прийшлось розділити, щоб можна було передавати, окрема широту та окрему довготу, в бота.

```
def get_location_info(self, id_loc):
  """отримуємо посилання на сайт кафедри"""
  with self.connection:
    return self.cursor.execute("SELECT href FROM locations WHERE id_loc = ?",
(id_loc,)).fetchone()
```

Також, додається запит який передає посилання на сайт кафедри, якщо відповідний пункт бази даних є заповнений. Якщо поли посилання пусте, то бот нічого не додає в своєму повідомленні.

```
def get_location_name(self, id_loc):
  """Отримуємо назву локації"""
  with self.connection:
    return self.cursor.execute("SELECT loc_name FROM locations WHERE id_loc =
?", (id_loc,)).fetchone()
```

І відповідно, додається назва вибраної локації для кращого орієнтування в чаті.

```
def get_book_name(self, id_b):
  """Отримуємо назву книги """
  with self.connection:
    name = self.cursor.execute("SELECT name_b FROM books WHERE id_b = ?",
(id_b,)).fetchone()
    return (name)

def get_book_auther(self, id_b):
  """Отримуємо ПІБ автора книги"""
  with self.connection:
    auther = self.cursor.execute("SELECT auther_b FROM books WHERE id_b = ?",
(id_b,)).fetchone()
    return (auther)
```

Для роботи з базою книг, використовується всього два запити, перший - для отримання назви книги, а другий – для отримання автора книги. Випадково вибирається **id** книги і це **id** передається в запити.

```
def get_event(self, id):
  """Назву заходу """
  with self.connection:
    return self.cursor.execute("SELECT night_party FROM after_univ WHERE id =
?", (id,)).fetchone()

def get_event_img(self, id):
  """Посилання на захід"""
  with self.connection:
    return self.cursor.execute("SELECT url FROM after_univ WHERE id = ?",
(id,)).fetchone()
```

Також в базі даних знаходить пункт про проведення студентських заходів в якому є короткий опис, та посилання на зображення чи на сторінку заходу.

Розділ IV Використання технології API

4.1 Технологія API

API(Application Programming Interface) - це є програмний посередник, який дозволяє двом додаткам спілкуватися один з одним. Щоразу, коли ви використовуєте програму, як Facebook, надсилаєте миттєві повідомлення або перевіряєте погоду на своєму телефоні, ви використовуєте API[11].

Коли ви використовуєте програму на своєму мобільному телефоні, ця програма підключається до Інтернету та надсилає дані на сервер. Потім сервер отримує ці дані, інтерпретує їх, виконує необхідні дії та надсилає їх назад на ваш телефон. Потім програма інтерпретує ці дані та представляє вам необхідну інформацію в доступному для читання вигляді. Ось що таке API - все це відбувається через API.

Щоб краще пояснити це, наведемо знайомий приклад:

Уявіть, що ви сидите за столиком у ресторані з меню на вибір. Кухня є частиною «системи», яка підготує ваше замовлення. Чого не вистачає, так це важливого зв'язку, щоб передати ваше замовлення на кухню та доставити їжу назад до вашого столу. Ось тут на допомогу приходить офіціант або API. Офіціант — це месенджер або API, який приймає ваш запит чи замовлення і повідомляє кухні — системі — що робити. Потім офіціант повертає вам відповідь; в даному випадку це їжа.

Інший приклад може бути таким:

Якщо ви знайомі з процесом пошуку авіарейсів в Інтернеті. Як і в ресторані, у вас є різноманітні варіанти на вибір, включаючи різні міста, дати

[11] https://www.mulesoft.com/resources/api/what-is-an-api

відправлення та повернення тощо. Уявімо, що ви бронюєте рейс на веб-сайті авіакомпанії. Ви вибираєте місто і дату відправлення, місто і дату повернення, клас салону, а також інші змінні. Щоб забронювати рейс, ви взаємодієте з веб-сайтом авіакомпанії, щоб отримати доступ до їх бази даних і перевірити, чи вільні місця на ці дати та які можуть бути витрати.

4.2 API технологія для IMDb

У магістерський роботі використовується API IMDb для роботи з їхньою базою даних. Це був один із способі отримати дані з їхнього сайту для бота. Другий спосіб – це був парсинг сайту[12].

Отже, за допомогою API, для роботи функцій по вибору фільмів та серіалів в Telegram боті використовується такий код:

```
elif (message.text == "Фільм на вечір \U0001F3A5"):
  num = random.randrange(0, 249, 1)
  href = ia.get_top250_movies()
  href = href[num]
  id_href = ia.get_imdbID(href)
  bot.send_message(message.chat.id,text =
"https://www.imdb.com/title/tt"+id_href)
```

Записується реакція на текст в чаті, при натисканні на відповідні кнопки, виконується випадковий вибір номера від 0 до 249 – це пов'язано з тим, що рейтинг IMDb складається з 250 фільмів(Топ 250), починаємо відраховувати від 0 з кроком 1, отримавши випадкове значення, ми переходимо до функцій API. Щоб не записувати постійно рядок звернення до бібліотеки, він був скорочений.

```
ia = imdb.IMDb()
```

Через функцію **get_top250_movies**, ми отримуємо доступ до топу фільмів де ми можемо отримати назву фільму.

Після звернення ми перепризначуємо змінну **href**, але в же додаючи випадкове значення яке ми отримали раніше, за допомогою таких маніпуляцій ми отримуємо випадковий фільм. Кожен фільм на сайті має своє **id** і нам

[12] https://imdbpy.github.io/

потрібно його отримати щоб надати посилання користувачеві, тому ми звертаємось до **get_imdbID**, яка дозволить нам отримати **id** щоб додати його в посилання на фільм і вивести його користувачеві. Тому записуємо в дужках до функції ми вказуємо **href**, в якій знаходиться назва фільму, отримавши **id**, ми додаємо його до загального посилання на фільм і через бота передаєм це повне посилання, в кінці отримавши робоче посилання з коротким описом/

За таким самим принципом працює і вибір серіалу на вечір.

```
elif (message.text == "Серіал на вечір \U0001F4FA"):
  num = random.randrange(0, 249, 1)
  href = ia.get_top250_tv()
  href = href[num]
  id_href = ia.get_imdbID(href)
  bot.send_message(message.chat.id, text="https://www.imdb.com/title/tt" +
id_href)
```

Тільки з однією відмінністю, що вибирається з топ 250 серіалів, для нього використовується інша функція - **get_top250_tv**.

В магістерській роботі використовується тільки невелика частину функціоналу API IMDb, так як основна ціль бота зовсім інша. Але якщо створювати бота який більш вузько направлений, основною тематикою якого буде кінематограф, API IMDb дозволяє отримати обширну інформацію з баз сайту IMDb, можна отримати велику кількість інформації що стосується фільмів, акторів, персонажів чи будь кого хто якимось чином стосується кінематографу(режисери, відповідальних за спецефекти, монтажери, продюсери та інші).

4.3 API технологія для Instagram

Для пункту про новини, в боті передбачена робота з API Instagram, так як велику кількість новин та інформація яка стосується університету можна знайти саме в Instagram. Для роботи з Instagram було розглянуто два варіанти опрацювання даних, як і в роботі з IMDb, це був парсинг та робота з API. Для використання було обрано саме API, через обширний функціонал по роботі з Instagram.

Спосіб який використовується в магістерський роботі потребує авторизації користувача, для отримання даних – це пов'язано з тим що деякі сторінки користувачів можуть бути приховані для загального доступу, тому в окремому файлі, ми підтягуємо дані акаунту користувача для авторизації та скорочуємо код входу до однієї змінної.

```
api = InstagramAPI(pass_log.login,pass_log.password)
```

Після чого створюємо функцію **get_lastposts** в якій ми отримуємо код останнього запису користувача, в нашому випадку - це пост з каналу Буковинського державного медичного університету. В циклі **for** ми отримуємо всі пости які доступні на початковій сторінці тобто – 22 пости, але повертаємо тільки код першого, прописуючи **return[0]**[13].

```
def get_lastposts(us_id):
  api.getUserFeed(us_id)
  if 'items' in api.LastJson:
    info = api.LastJson['items']
    posts=[]
    for media in info:
      if (media['caption']!=None):
       #print(media['caption']['media_id'])
        posts.append(media['caption']['media_id'])
    return posts[0]
```

Таким чином ми отримуємо зашифрований код так званий **media_id**, який ми не зможемо просто використати як посилання для користувача, його потрібно дешифрувати. Дешифрування відбувається таким чином, ми отримуємо **media_id**, отримуємо остачу від ділення на 64, цю остачу ми віднімаємо від **media_id**, та цю ж остачу проганяємо по алфавіту (**alphabet**) для підбору по рядку символів, щоб знайти символ, який буде належати **id** поста. Ця дія повторюється поки не закінчиться рядок **media_id**, як тільки рядок **media_id** закінчується, ми генеруємо посилання на пост та відправляємо його користувачеві.

```
def getInstagramUrlFromMediaId(media_id):
  alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-
_'
  shortened_id = ''

  while media_id > 0:
    remainder = media_id % 64
```

[13] https://github.com/facebookarchive/python-instagram

```
    # dual conversion sign gets the right ID for new posts
    media_id = (media_id - remainder) // 64
    # remainder should be casted as an integer to avoid a type error.
    shortened_id = alphabet[int(remainder)] + shortened_id
  return 'https://instagram.com/p/' + shortened_id
```

Розділ V Додаткові технології та можливості в месенджері Telegram

5.1 Ігри в месенджері Telegram

З 2016 року в месенджері Telegram з'явилась функція для ботів «Ігри». Всі ігри для Telegram можуть бути написані за допомогою HTML-5, тобто це сайт-додаток, який відкривається у внутрішньому браузері месенджера, та може керуватись як всі сайти за допомогою сенсорного дисплею.

Дана функція доступна кожному, і будь хто може створити та розмістити свою гру в телеграмі, для цього потрібно створити відповідний продукт за допомогою web-мов та його на хостингу або в себе на сервері.

Складність ігор залежить тільки від вмінь та фантазії програміста який її створює.

Ігри можна використовувати не тільки у вигляді «Вбивці часу», а і в якості тренування для мозку чи пам'яті.

5.2 Використання функції ігор месенджеру Telegram в боті БДМУ

В магістерській роботі функція ігор в Telegram використовується як підготовчо-тренувальна програма для студентів, щоб дати їм змогу підготуватись до тестів, екзаменів або просто підвищити свій загальний рівень знань в якійсь із медичних галузей.

Принцип гри в боті достатньо простий, при натисканні на пункт меню «Гра в питання» у вкладці «Free time» відкривається меню для в ходу в гру (Додаток 5.2 Рис. 1), після натискання на кнопку «Play exam_game» користувача переключає на внутрішній браузер, якщо він працює з телефону, і відкриває нову в кладку в браузері, якщо робота з ботом відбувається на комп'ютері[14].

Правила гри дуже прості, натискаючи на кнопку старт, з відповідної бази питань підтягується одне випадкове і користувачеві дається від 3 до 5 секунд на відповідь (Додаток 1.3 Рис. 16), після закінчення відведеного часу на картці випливає відповідь на питання. (Додаток 1.3 Рис. 17). Таким чином, граючи в цю гру, користувач може підготуватись до тестування чи екзамену.

Для гри використовується одно сторінковий сайт який розвернутий на локальній машині за допомогою програми **LocalHsot**. База даних питань знаходиться в тій же програмі та працює за допомогою **phpMyAdmin**. До бази даних я під'єднуюсь за допомогою **php** коду.

```
$mysqli = new
mysqli();
$mysqli->options(MYSQLI_OPT_INT_AND_FLOAT_NATIVE, 1);
$mysqli->real_connect("127.0.0.1", "admin", "prince69", "admin_examquiz");
```

За допомогою двох запитів здійснюються всі функції гри. Для вибору запитання з бази я використовує запит, який дозволяє отримати **id** останнього елементу з бази, щоб можна було, випадковим чином обрати запитання з бази.

```
$sqlId = $mysqli->query("SELECT id FROM `answer_questioon` ORDER BY id DESC
LIMIT 1");
$resultId = mysqli_fetch_array($sqlId);
$sqlID = $resultId['id'];
$id = rand(0,$sqlID);
```

І за допомогою функції **rand** ми отримуємо випадковий елемент з діапазону від нуля **id** останнього питання в базі.

[14] https://tlgrm.ru/docs/bots/games

Використовуючи отриманий результат ми знову звертаємося до бази і за допомогою другого запиту отримуємо запитання з бази, критерієм вибору для якого є **id**.

```
$sql = $mysqli->query("SELECT `question`, `answer` FROM `answer_questioon`
WHERE id = $id");
$result = mysqli_fetch_array($sql);
$question = $result['question'];
$answer = $result['answer'];
```

Результати запиту ми передаємо у змінні, щоб потім вивести їх в текст на картці. Картка - є кнопкою, при натисканні на яку виконується код що заміняє текст на ній питаннями та відповідями з бази даних.

```
Const ques = "<?php echo $question; ?>";

button.innerHTML = "<h1>" + ques + "</h1>";
setTimeout(getQues, 3000);
});
function
getQues()
{
  const answ = "<?php echo $answer; ?>";
button.innerHTML = "<h1>" + answ + "</h1>";
}
```

За допомогою коду **JS** та **PhP** ми заміняємо текст в кнопці **button.innerHTML** – виконує заміну тексту та дозволяє додати потрібні атрибути до виведеного тексту, а за допомогою **setTimeout** ми робимо затримку перед викликом функції виводу відповіді для користувача.

Висновок

Отже, виконана магістерська робота дала мені змогу освоїти нові технології та навички що стосуються роботи з месенджерами та їхніми функціями.

До корисних навичок я можу віднести роботу з локальним сервером та розміщенням на ньому свого проекту. Робота з технологією API, та використання популярних API (Instagram, IMdB) для функцій бота. Робота з SQL базами даних в новому середовищі SQLight.

Також, додаткова робота з такими функціями месенджера як ігри, в якій використовувались технології веб розробки, як FrontEnd(HTML, CSS, JS) так і BackEnd (PHP). Також під’єднання гри до бази PhPMyAdmin, та роботі з запитами до бази даних.

Участь в науковій конференції та презентування мого проекту на ній, дала змогу проявити себе з боку доповідача, що частково є обов'язками Тімліда.

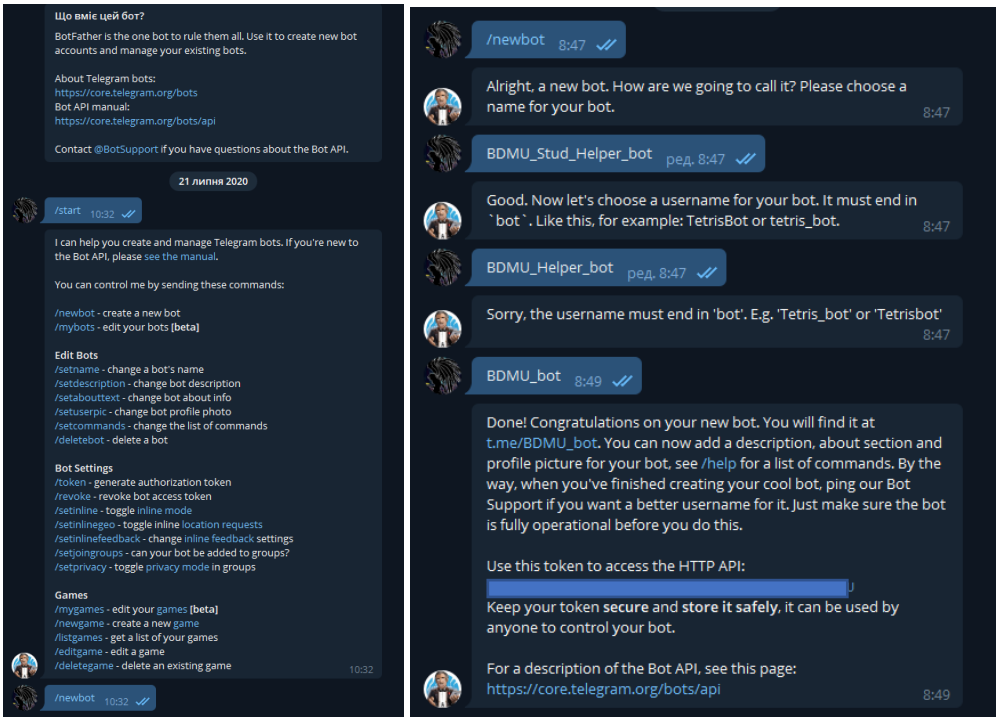
Використанні технології та набуті знання, під час праці над магістерською роботою, мені зможуть допомогти в подальшій роботі з месенджерами, автоматизації процесів через Telegram-бота, або інших месенджерів.

Список літератури

1. https://www.mulesoft.com/resources/api/what-is-an-api
2. https://www.sqlite.org/about.html
3. https://www.youtube.com/watch?v=M8fhrtvedHA&list=PLwsbEQtav8RGTMuZTBcWLbFBggEbeqRd1&index=3
4. https://imdbpy.github.io/
5. https://github.com/facebookarchive/python-instagram
6. https://docs.python.org/3/library/datetime.html
7. https://docs.python.org/3/library/random.html
8. https://www.python.org/
9. https://tlgrm.ru/docs/bots/games
10. Чорноус Олександр. Створення Telegram-бота для допомоги студенту вищого навчального закладу // Матеріали студентської наукової конференції Чернівецького національного університету (20–21 квітня 2021 року). Факультет математики та інформатики. – Чернівці: Чернівецький нац. ун-т ім. Ю. Федьковича, 2021.– С.76–77 (Науковий керівник – доц. Юрченко І.В.). https://drive.google.com/file/d/1aMb6oag04cMMNMWkzAmZHeTXFf7TJTl5/view https://archer.chnu.edu.ua/xmlui/handle/123456789/1779
11. https://github.com/eternnoir/pyTelegramBotAPI

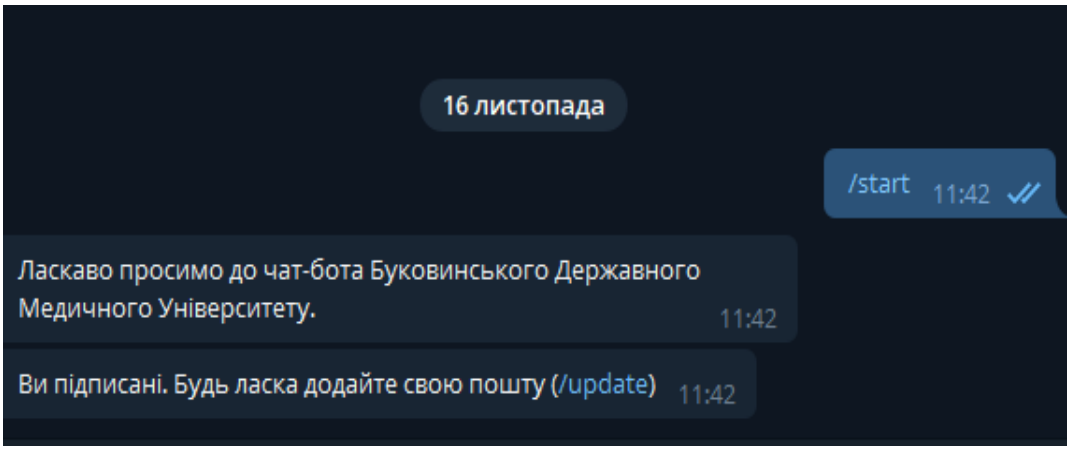
Додаток

Додаток 1.3

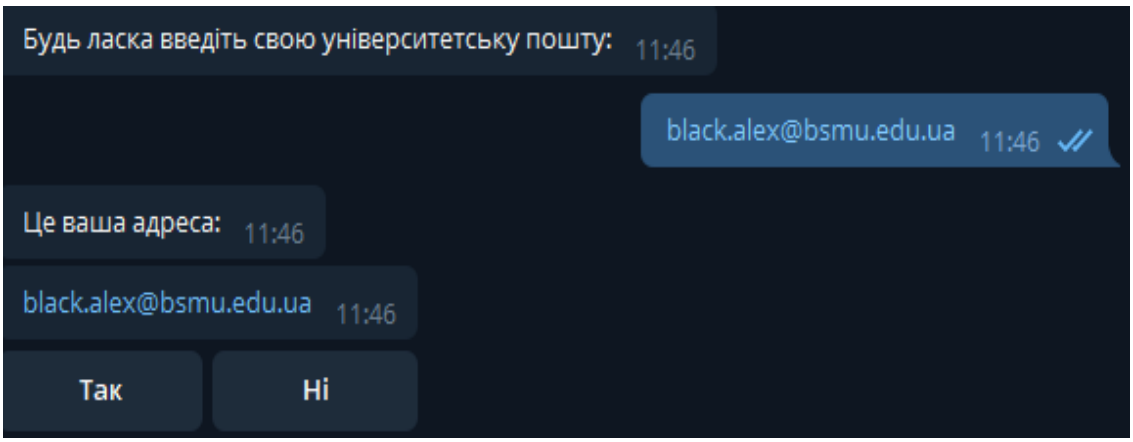


(Рис. 1\2)

Додаток 2.1



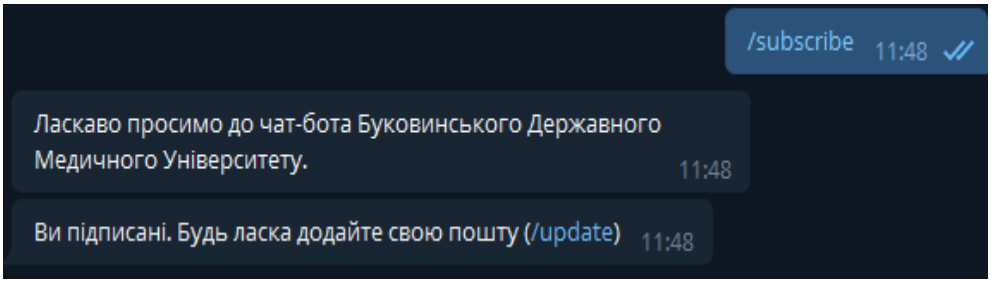
(Рис. 1)



(Рис. 2)

id_us	email	profetion	year	last_date
610423001	black.alex@bsmu.edu.ua	NULL	NULL	2021-11-16 11:40:15.124168

(Рис. 3)

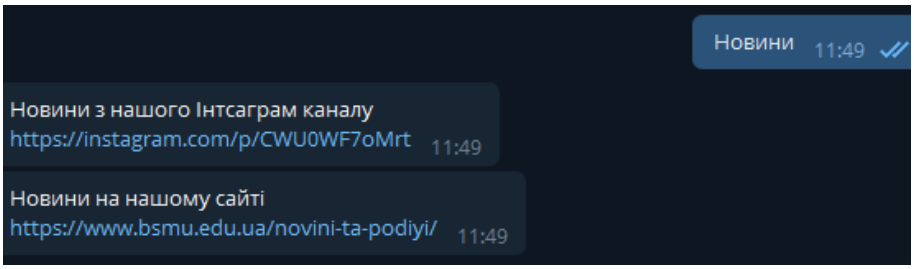


(Рис. 4)

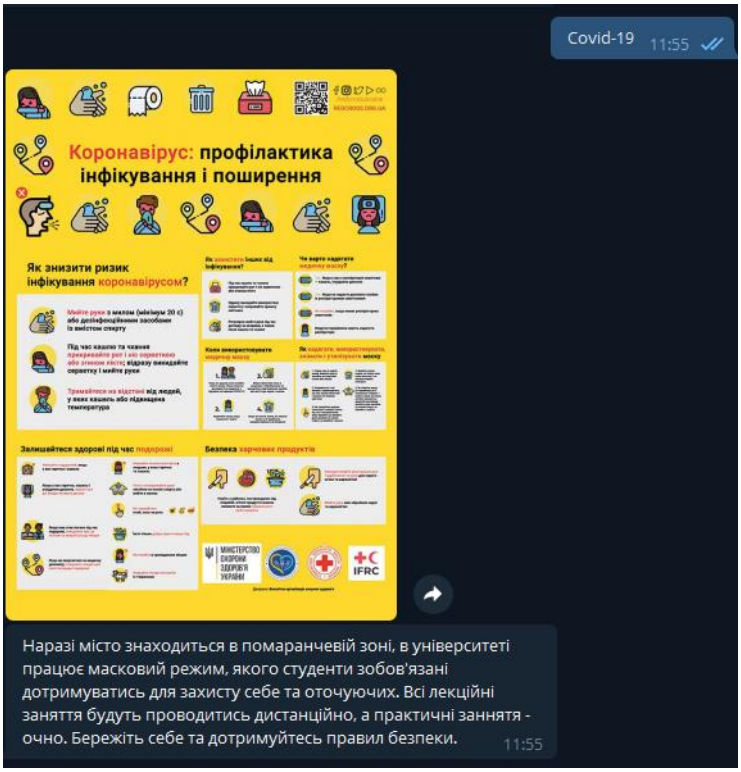
	id	user_id	status	sub_date
1	1	610423001	1	2020-12-16 11:11:19.432436
2	2	1446311787	1	2021-01-20 11:16:56.933196
3	3	135464283	1	2021-08-02 11:56:18.111265
4	4	684805479	1	2021-11-02 19:31:57.396229
5	5	996074700	1	2021-11-11 10:43:44.549577

(Рис. 5)

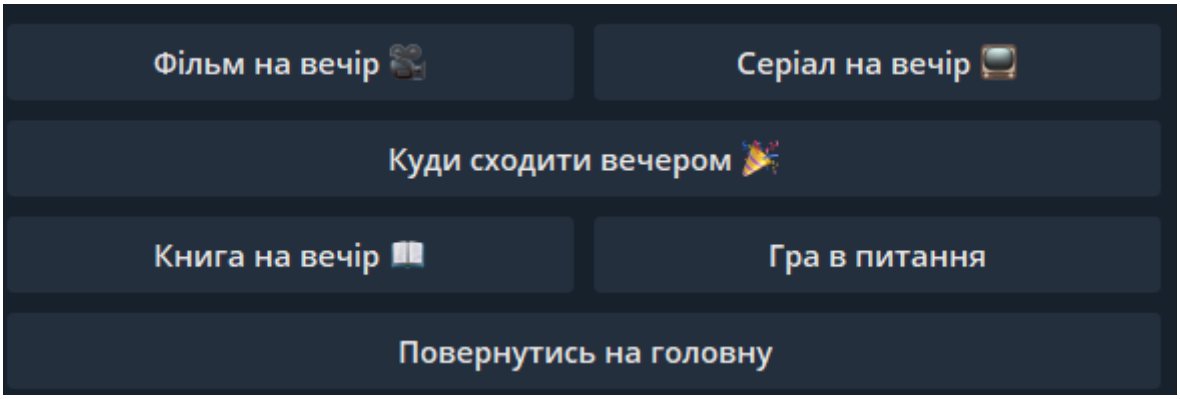
(Рис. 6)



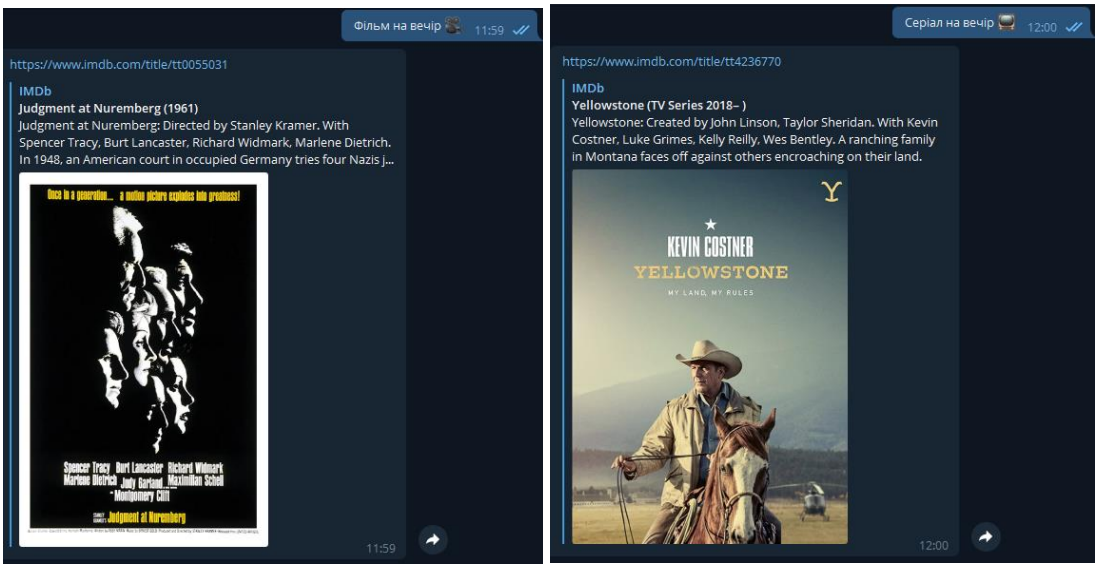
(Рис. 7)



(Рис. 8)



(Рис. 9)



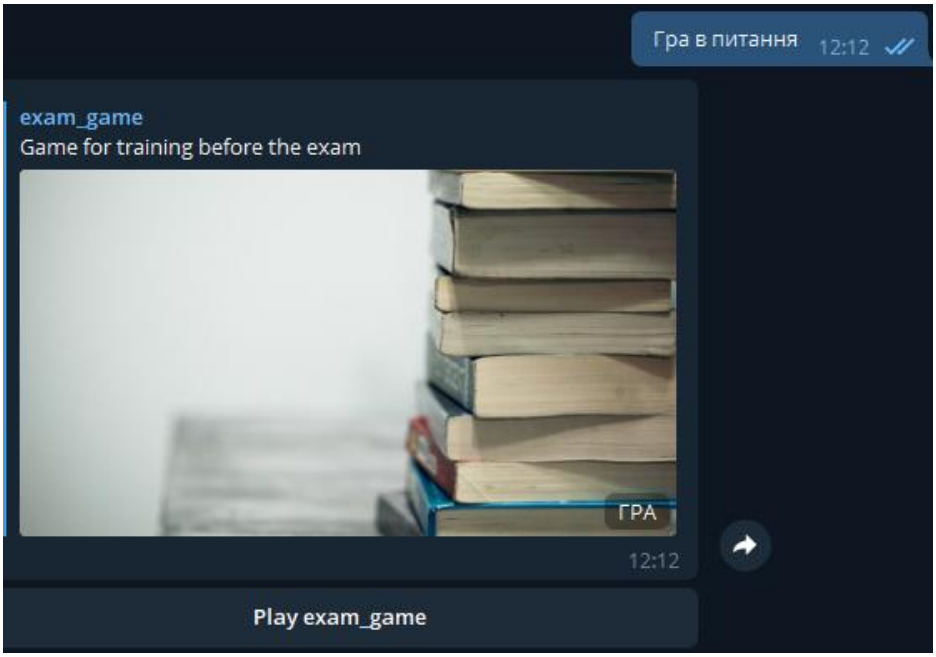
(Рис. 10 \ 11)

1	Сторонній	Альбер Камю
2	У пошуках утраченого часу	Марсель Пруст
3	Процес	Франц Кафка
4	Маленький принц	Антуан де Сент-Екзюпері
5	Умови людського існування	Андре Мальро
6	Подорож на край ночі	Луї-Фердінан Селін
7	Грона гніву	Джон Стейнбек
8	По кому подзвін	Ернест Хемінгуей

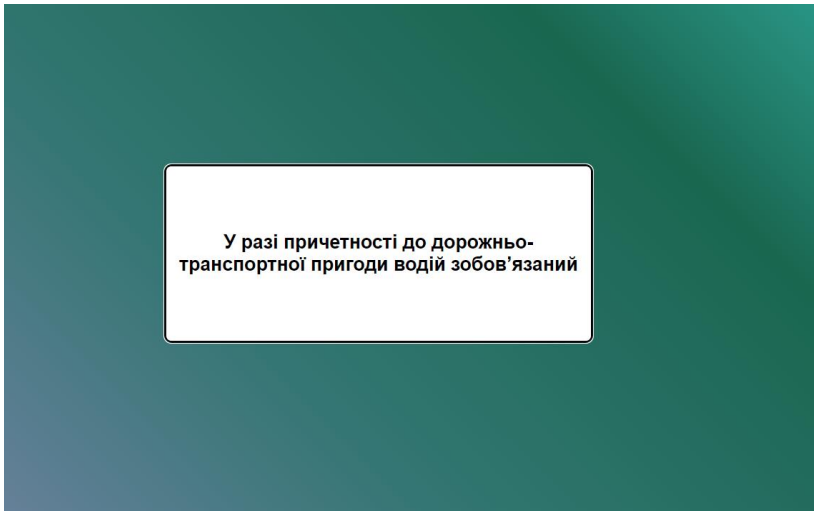
(Рис. 12 \ 13)

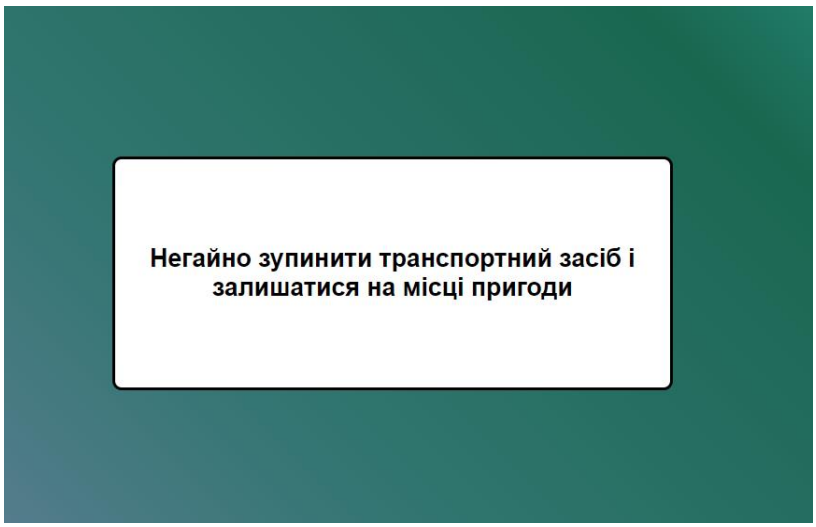

Куди сходити вечером
12:03
PLACEHOLDER
Тут могла бути ваша реклама.
12:03

(Рис. 14)

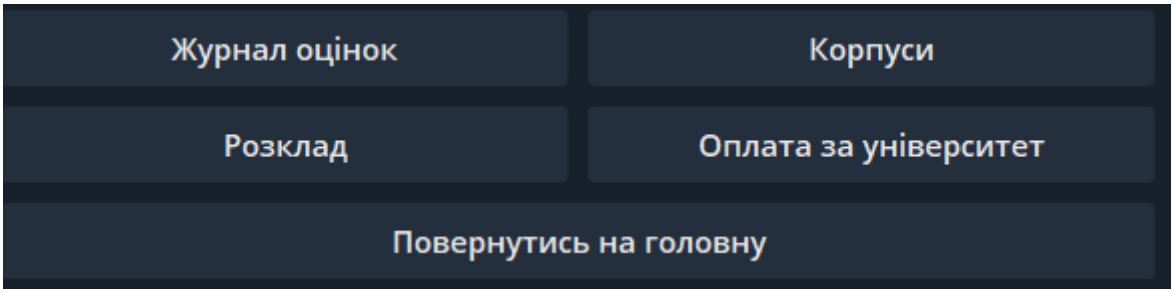

Гра в питання
12:12
exam_game
Game for training before the exam
ГРА
12:12
Play exam_game

(Рис. 15)

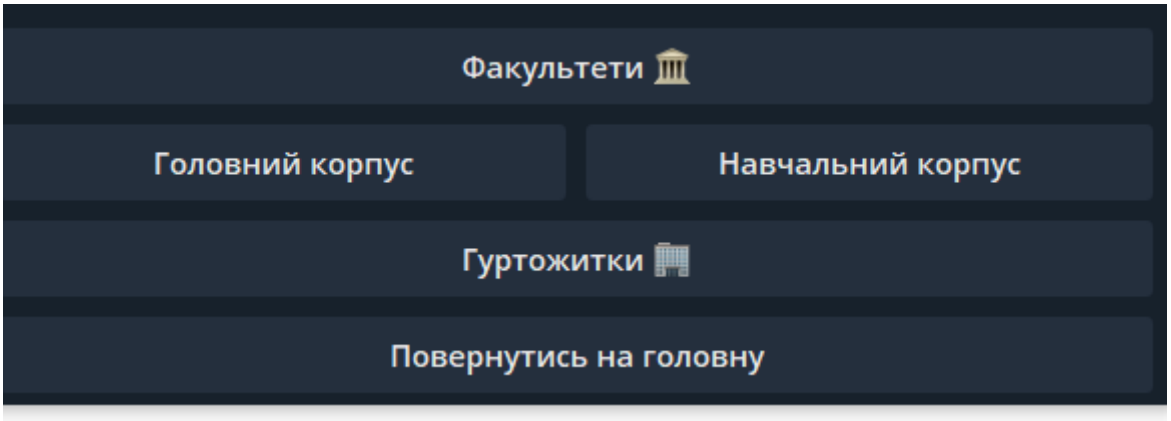

У разі причетності до дорожньо-транспортної пригоди водій зобов'язаний

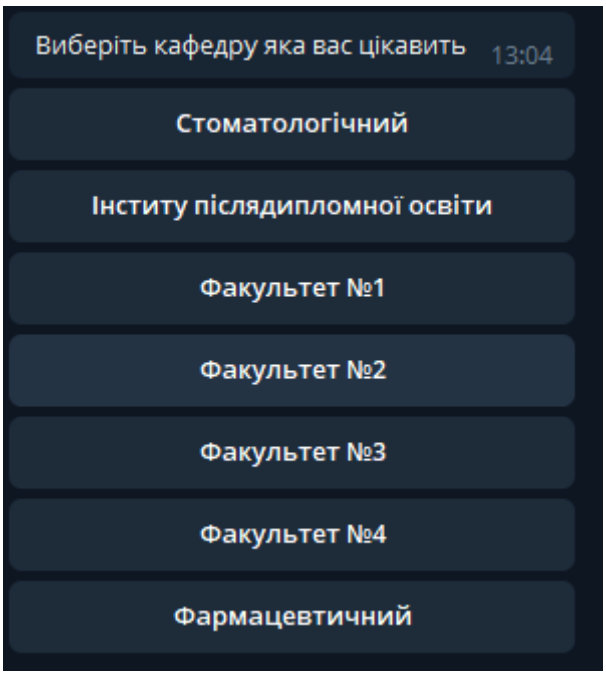


(Рис. 16/17)

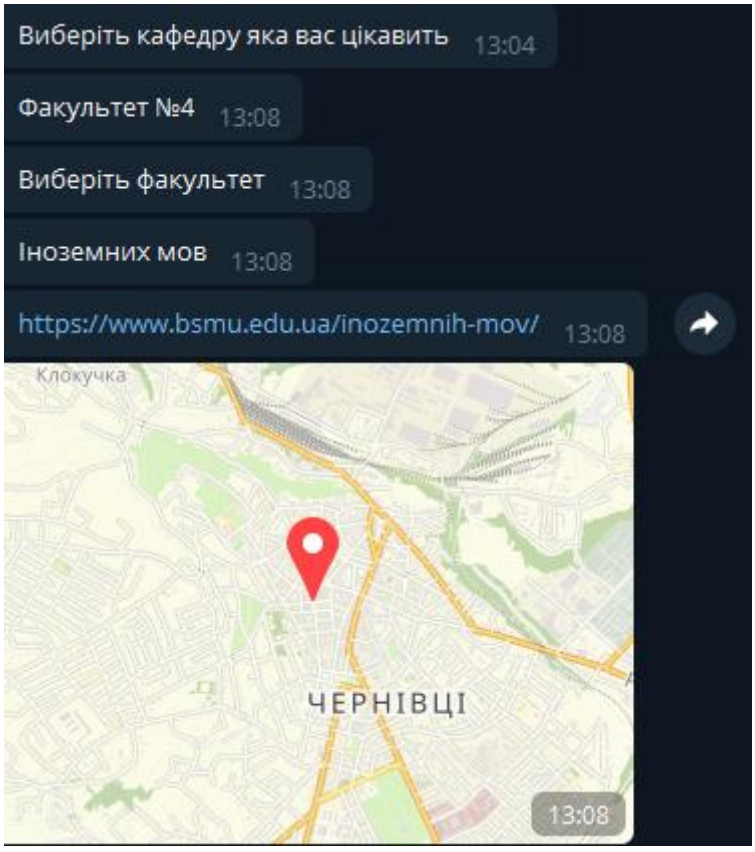


(Рис. 18)

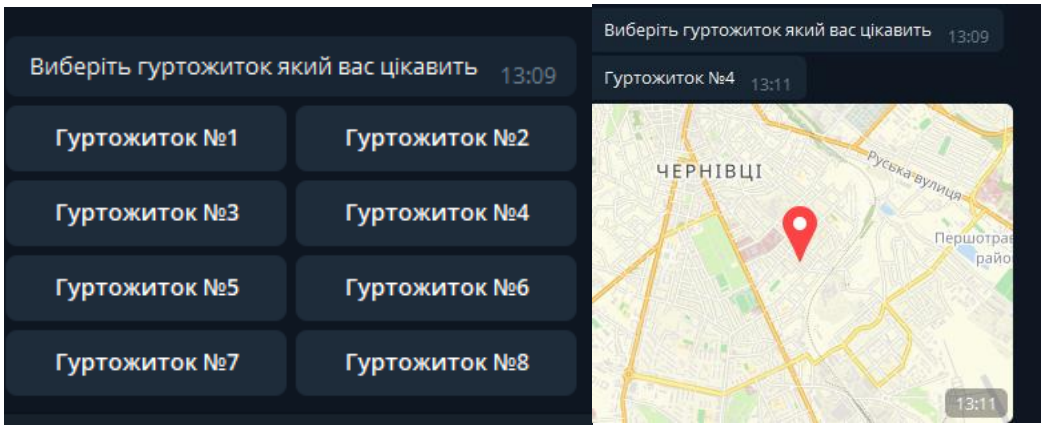




(Рис. 19 \ 20)

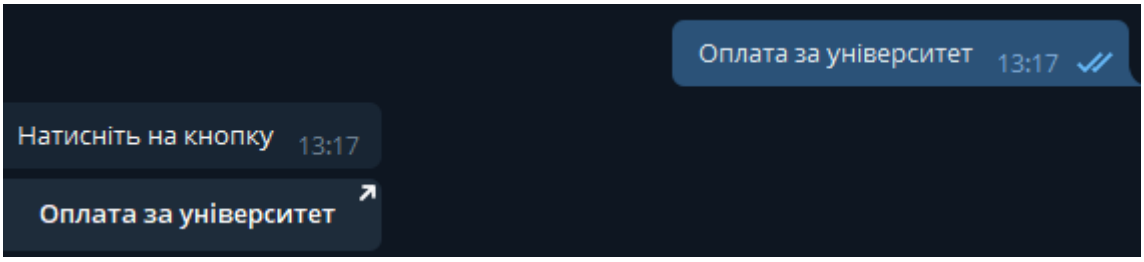


(Рис. 21)



(Рис. 22 \ 23)

(Рис. 24)



(Рис. 25)

Додаток 2.2(Код роботи бота)

```
import config
import telebot
import random
import imdb
import re
import pass_log

from InstagramAPI import InstagramAPI
from telebot import types
from sqlightDB import SQLighter
from datetime import datetime
from bs4 import BeautifulSoup


db = SQLighter('db.db')
bot = telebot.TeleBot(config.TOKEN)
ia = imdb.IMDb()
api = InstagramAPI(pass_log.login,pass_log.password)
api.login()

#Instagram
def get_lastposts(us_id):
  api.getUserFeed(us_id)
  if 'items' in api.LastJson:
    info = api.LastJson['items']
    posts=[]
    for media in info:
      if (media['caption']!=None):
       #print(media['caption']['media_id'])
        posts.append(media['caption']['media_id'])
    return posts[0]

def getInstagramUrlFromMediaId(media_id):
  alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-
_'
  shortened_id = ''

  while media_id > 0:
    remainder = media_id % 64
    # dual conversion sign gets the right ID for new posts
    media_id = (media_id - remainder) // 64
    # remainder should be casted as an integer to avoid a type error.
    shortened_id = alphabet[int(remainder)] + shortened_id
  return 'https://instagram.com/p/' + shortened_id

# Клавиатура "Головне меню"
btn_stud = telebot.types.KeyboardButton("Для студента")
btn_cov = telebot.types.KeyboardButton("Covid-19")
btn_news = telebot.types.KeyboardButton("Новини")
btn_fr_tm = telebot.types.KeyboardButton("Free time")

#Повернення на головну
return_main = telebot.types.KeyboardButton("Повернутись на головну")

#Клавіатура "Для студента"
btn_stud_kaf = telebot.types.KeyboardButton("Корпуси")
btn_stud_tmtb = telebot.types.KeyboardButton("Розклад")
btn_stud_jrl = telebot.types.KeyboardButton("Журнал оцінок")
```

```
btn_stud_std = telebot.types.KeyboardButton("Оплата за університет")

#Клавіатура "Вільний час"
free_button1 = telebot.types.KeyboardButton("Фільм на вечір \U0001F3A5")
free_button2 = telebot.types.KeyboardButton("Серіал на вечір \U0001F4FA")
free_button3 = telebot.types.KeyboardButton("Книга на вечір \U0001F4D6")
free_button4 = telebot.types.KeyboardButton("Гра в питання ")
free_button5 = telebot.types.KeyboardButton("Куди сходити вечером
\U0001F389")

#Кнопки вибору гуртожитків\кафдр
btn_hostels = telebot.types.KeyboardButton("Гуртожитки \U0001F3E2")
btn_faculty = telebot.types.KeyboardButton("Факультети \U0001F3DB")
loc_btn_main = telebot.types.KeyboardButton("Головний корпус")
loc_btn_study = telebot.types.KeyboardButton("Навчальний корпус")

#Кнопки факультетів
loc_btn_fac_stomat = telebot.types.InlineKeyboardButton("Стоматологічний",
callback_data = "loc_btn_fac_stomat")
loc_btn_fac_inst = telebot.types.InlineKeyboardButton("Інституту післядипломної
освіти", callback_data = "loc_btn_fac_inst")
loc_btn_fac_fac1 = telebot.types.InlineKeyboardButton("Факультет №1",
callback_data = "loc_btn_fac_fac1")
loc_btn_fac_fac2 = telebot.types.InlineKeyboardButton("Факультет №2",
callback_data = "loc_btn_fac_fac2")
loc_btn_fac_fac3 = telebot.types.InlineKeyboardButton("Факультет №3",
callback_data = "loc_btn_fac_fac3")
loc_btn_fac_fac4 = telebot.types.InlineKeyboardButton("Факультет №4",
callback_data = "loc_btn_fac_fac4")
loc_btn_fac_farm = telebot.types.InlineKeyboardButton("Фармацевтичний",
callback_data = "loc_btn_fac_farm")


#Кнопки локацій кафедр Стомат
loc_btn_fac_stomat_1 = telebot.types.InlineKeyboardButton("Клінічної
анатомії", callback_data = "loc_btn_fac_stomat_1")
loc_btn_fac_stomat_2 = telebot.types.InlineKeyboardButton("Гістології,
цитології та ембріології", callback_data = "loc_btn_fac_stomat_2")
loc_btn_fac_stomat_3 = telebot.types.InlineKeyboardButton("Дитячої хірургії
та отоларингології", callback_data = "loc_btn_fac_stomat_3")
loc_btn_fac_stomat_4 = telebot.types.InlineKeyboardButton("Ортопедичної
стоматології", callback_data = "loc_btn_fac_stomat_4")
loc_btn_fac_stomat_5 = telebot.types.InlineKeyboardButton("Пропедевтики
внутрішніх хвороб", callback_data = "loc_btn_fac_stomat_5")
loc_btn_fac_stomat_6 = telebot.types.InlineKeyboardButton("Стоматології
дитячого віку", callback_data = "loc_btn_fac_stomat_6")
loc_btn_fac_stomat_7 = telebot.types.InlineKeyboardButton("Терапевтичної
стоматології", callback_data = "loc_btn_fac_stomat_7")
loc_btn_fac_stomat_8 = telebot.types.InlineKeyboardButton("Хірургічної
стоматології та щелепно-лицевої хірургії", callback_data =
"loc_btn_fac_stomat_8")
loc_btn_fac_stomat_9 = telebot.types.InlineKeyboardButton("Курс ЛОР-хвороб",
callback_data = "loc_btn_fac_stomat_9")


#Кнопки локацій кафедра Інституту
loc_btn_fac_inst_1 = telebot.types.InlineKeyboardButton("Анестезіології та
реаніматології", callback_data = "loc_btn_fac_inst_1")
loc_btn_fac_inst_2 = telebot.types.InlineKeyboardButton("Внутрішньої
медицини, фізичної реабілітації, спортивної медицини", callback_data =
"loc_btn_fac_inst_2")
loc_btn_fac_inst_3 = telebot.types.InlineKeyboardButton("Дерматовенерології",
callback_data = "loc_btn_fac_inst_3")
```

```
loc_btn_fac_inst_4 = telebot.types.InlineKeyboardButton("Нервових хвороб,
психіатрії та медичної психології", callback_data = "loc_btn_fac_inst_4")
loc_btn_fac_inst_5 = telebot.types.InlineKeyboardButton("Онкології та
радіології", callback_data = "loc_btn_fac_inst_5")
loc_btn_fac_inst_6 = telebot.types.InlineKeyboardButton("Сімейної медицини",
callback_data = "loc_btn_fac_inst_6")

#Кнопки локацій кафедр Факультету №1
loc_btn_fac_fac1_1 = telebot.types.InlineKeyboardButton("Акушерства та
гінекології", callback_data = "loc_btn_fac_fac1_1")
loc_btn_fac_fac1_2 = telebot.types.InlineKeyboardButton("Внутрішньої
медицини, клінічної фармакології та професійних хвороб", callback_data =
"loc_btn_fac_fac1_2")
loc_btn_fac_fac1_3 = telebot.types.InlineKeyboardButton("Медицини катастроф
та військової медицини", callback_data = "loc_btn_fac_fac1_3")
loc_btn_fac_fac1_4 = telebot.types.InlineKeyboardButton("Педіатрії,
неонатології та перинатальної медицини", callback_data =
"loc_btn_fac_fac1_4")
loc_btn_fac_fac1_5 = telebot.types.InlineKeyboardButton("Психології та
філософії", callback_data = "loc_btn_fac_fac1_5")
loc_btn_fac_fac1_6 = telebot.types.InlineKeyboardButton("Фтизіатрії та
пульмонології", callback_data = "loc_btn_fac_fac1_6")
loc_btn_fac_fac1_7 = telebot.types.InlineKeyboardButton("Хірургії №2",
callback_data = "loc_btn_fac_fac1_7")

#Кнопки локацій кафедр Факультету №2
loc_btn_fac_fac2_1 = telebot.types.InlineKeyboardButton("Акушерства,
гінекології та пеританології", callback_data = "loc_btn_fac_fac2_1")
loc_btn_fac_fac2_2 = telebot.types.InlineKeyboardButton("Внутрішньої медицини
та інфекційних хвороб", callback_data = "loc_btn_fac_fac2_2")
loc_btn_fac_fac2_3 = telebot.types.InlineKeyboardButton("Інфекційних хвороб
та епідеміологоії", callback_data = "loc_btn_fac_fac2_3")
loc_btn_fac_fac2_4 = telebot.types.InlineKeyboardButton("Клінічної
імунології, алергології та ендокринології", callback_data =
"loc_btn_fac_fac2_4")
loc_btn_fac_fac2_5 = telebot.types.InlineKeyboardButton("Офтальмології ім.
Б.Л. Радзіховського", callback_data = "loc_btn_fac_fac2_5")
loc_btn_fac_fac2_6 = telebot.types.InlineKeyboardButton("Патологічної
анатомії", callback_data = "loc_btn_fac_fac2_6")
loc_btn_fac_fac2_7 = telebot.types.InlineKeyboardButton("Травматології,
ортопедії", callback_data = "loc_btn_fac_fac2_7")
loc_btn_fac_fac2_8 = telebot.types.InlineKeyboardButton("Хірургії №1",
callback_data = "loc_btn_fac_fac2_8")

#Кнопки локацій кафедр Факультету №3
loc_btn_fac_fac3_1 = telebot.types.InlineKeyboardButton("Іноземних мов",
callback_data = "loc_btn_fac_fac3_1")
loc_btn_fac_fac3_2 = telebot.types.InlineKeyboardButton("Педіатрії та дитячих
інфекційних хвороб", callback_data = "loc_btn_fac_fac3_2")
loc_btn_fac_fac3_3 = telebot.types.InlineKeyboardButton("Соціальної медицини
та ООЗ", callback_data = "loc_btn_fac_fac3_3")
loc_btn_fac_fac3_4 = telebot.types.InlineKeyboardButton("Судової медицини та
медичного правознавства", callback_data = "loc_btn_fac_fac3_4")
loc_btn_fac_fac3_5 = telebot.types.InlineKeyboardButton("Урології та
нейрохірургії", callback_data = "loc_btn_fac_fac3_5")
loc_btn_fac_fac3_6 = telebot.types.InlineKeyboardButton("Загальної хірургії",
callback_data = "loc_btn_fac_fac3_6")


#Кнопки локації кафедр факультету №4
loc_btn_fac_fac4_1 = telebot.types.InlineKeyboardButton("Гігієни та
екології", callback_data = "loc_btn_fac_fac3_1")
loc_btn_fac_fac4_2 = telebot.types.InlineKeyboardButton("Мікробіології та
```

```
вірусології", callback_data = "loc_btn_fac_fac3_2")
loc_btn_fac_fac4_3 = telebot.types.InlineKeyboardButton("Патологічної
фізіології", callback_data = "loc_btn_fac_fac3_3")
loc_btn_fac_fac4_4 = telebot.types.InlineKeyboardButton("Педіатрії та
медичної генетики", callback_data = "loc_btn_fac_fac3_4")
loc_btn_fac_fac4_5 = telebot.types.InlineKeyboardButton("Фармакології",
callback_data = "loc_btn_fac_fac3_5")

#Кнопки локацій кафедр фармацевтичного факультету
loc_btn_fac_farm_1 = telebot.types.InlineKeyboardButton("Біологічної фізики
та медичної інформатики", callback_data = "loc_btn_fac_farm_1")
loc_btn_fac_farm_2 = telebot.types.InlineKeyboardButton("Біоорганічної і
біологічної хімії та клінічної біохімії", callback_data =
"loc_btn_fac_farm_2")
loc_btn_fac_farm_3 = telebot.types.InlineKeyboardButton("Фармацевтичної
ботаніки та фармакогнозії", callback_data = "loc_btn_fac_farm_3")
loc_btn_fac_farm_4 = telebot.types.InlineKeyboardButton("Медичної біології та
генетики", callback_data = "loc_btn_fac_farm_4")
loc_btn_fac_farm_5 = telebot.types.InlineKeyboardButton("Медичної та
фармацевтичної хімії", callback_data = "loc_btn_fac_farm_5")
loc_btn_fac_farm_6 = telebot.types.InlineKeyboardButton("Суспільних наук та
українознавства", callback_data = "loc_btn_fac_farm_6")
loc_btn_fac_farm_7 = telebot.types.InlineKeyboardButton("Фармації",
callback_data = "loc_btn_fac_farm_7")
loc_btn_fac_farm_8 = telebot.types.InlineKeyboardButton("Фізіології
ім.Я.Д.Кіршенблата", callback_data = "loc_btn_fac_farm_8")

#Кнопки локацій гуртожитків
loc_btn_hostel_1 = telebot.types.InlineKeyboardButton("Гуртожиток №1",
callback_data = "loc_btn_hostel_1")
loc_btn_hostel_2 = telebot.types.InlineKeyboardButton("Гуртожиток №2",
callback_data = "loc_btn_hostel_2")
loc_btn_hostel_3 = telebot.types.InlineKeyboardButton("Гуртожиток №3",
callback_data = "loc_btn_hostel_3")
loc_btn_hostel_4 = telebot.types.InlineKeyboardButton("Гуртожиток №4",
callback_data = "loc_btn_hostel_4")
loc_btn_hostel_5 = telebot.types.InlineKeyboardButton("Гуртожиток №5",
callback_data = "loc_btn_hostel_5")
loc_btn_hostel_6 = telebot.types.InlineKeyboardButton("Гуртожиток №6",
callback_data = "loc_btn_hostel_6")
loc_btn_hostel_7 = telebot.types.InlineKeyboardButton("Гуртожиток №7",
callback_data = "loc_btn_hostel_7")
loc_btn_hostel_8 = telebot.types.InlineKeyboardButton("Гуртожиток №8",
callback_data = "loc_btn_hostel_8")

#Клавіатура жанрів
free_in_btn1 = telebot.types.InlineKeyboardButton("Фантастика", callback_data
= "free_in_btn1")
free_in_btn2 = telebot.types.InlineKeyboardButton("Комедія", callback_data =
"free_in_btn2")
free_in_btn3 = telebot.types.InlineKeyboardButton("Жахи", callback_data =
"free_in_btn3")
free_in_btn4 = telebot.types.InlineKeyboardButton("Боєвик", callback_data =
"free_in_btn4")
free_in_btn5 = telebot.types.InlineKeyboardButton("Фентезі", callback_data =
"free_in_btn5")
free_in_btn6 = telebot.types.InlineKeyboardButton("Драма", callback_data =
"free_in_btn6")
free_in_btn7 = telebot.types.InlineKeyboardButton("Романтика", callback_data
= "free_in_btn7")
free_in_btn8 = telebot.types.InlineKeyboardButton("Випадковий", callback_data
= "free_in_btn8")
```

```
#Вибір курсу
inline_btn_1 = telebot.types.InlineKeyboardButton("1",
callback_data="button1")
inline_btn_2 = telebot.types.InlineKeyboardButton("2",
callback_data="button2")
inline_btn_3 = telebot.types.InlineKeyboardButton("3",
callback_data="button3")

#Кнопки підтвердження
inline_btn_Y = telebot.types.InlineKeyboardButton("Так",
callback_data="button_Y")
inline_btn_N = telebot.types.InlineKeyboardButton("Ні",
callback_data="button_N")


markup_main = telebot.types.ReplyKeyboardMarkup(resize_keyboard =
True).row(btn_news, btn_cov).row(btn_fr_tm, btn_stud)
markup_frtm = telebot.types.ReplyKeyboardMarkup(resize_keyboard =
True,).row(free_button1,
free_button2).add(free_button5).row(free_button3,free_button4).add(return_mai
n)
markup_stud = telebot.types.ReplyKeyboardMarkup(resize_keyboard =
True).row(btn_stud_jrl, btn_stud_kaf).row(btn_stud_tmtb,
btn_stud_std).add(return_main)
markup_stud_course = telebot.types.InlineKeyboardMarkup().row(inline_btn_1,
inline_btn_2, inline_btn_3)
markup_YN = telebot.types.InlineKeyboardMarkup().row(inline_btn_Y,
inline_btn_N)
markup_loc_kafedra =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_stomat).add(loc_btn_fac_
inst).add(loc_btn_fac_fac1).add(loc_btn_fac_fac2).add(loc_btn_fac_fac3).add(l
oc_btn_fac_fac4).add(loc_btn_fac_farm)
markup_loc_hostels =
telebot.types.InlineKeyboardMarkup().row(loc_btn_hostel_1,loc_btn_hostel_2).a
dd(loc_btn_hostel_3,loc_btn_hostel_4).add(loc_btn_hostel_5,loc_btn_hostel_6).
add(loc_btn_hostel_7,loc_btn_hostel_8)
markup_loc_menu = telebot.types.ReplyKeyboardMarkup(resize_keyboard =
True).add(btn_faculty).row(loc_btn_main,
loc_btn_study).add(btn_hostels).add(return_main)
markup_location_stomat =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_stomat_1).add(loc_btn_fa
c_stomat_2).add(loc_btn_fac_stomat_3).add(loc_btn_fac_stomat_4).add(loc_btn_f
ac_stomat_5).add(loc_btn_fac_stomat_6).add(loc_btn_fac_stomat_7).add(loc_btn_
fac_stomat_8).add(loc_btn_fac_stomat_9)
markup_location_inst =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_inst_1).add(loc_btn_fac_
inst_2).add(loc_btn_fac_inst_3).add(loc_btn_fac_inst_4).add(loc_btn_fac_inst_
5).add(loc_btn_fac_inst_6)
markup_location_fac1 =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_fac1_1).add(loc_btn_fac_
fac1_2).add(loc_btn_fac_fac1_3).add(loc_btn_fac_fac1_4).add(loc_btn_fac_fac1_
5).add(loc_btn_fac_fac1_6).add(loc_btn_fac_fac1_7)
markup_location_fac2 =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_fac2_1).add(loc_btn_fac_
fac2_2).add(loc_btn_fac_fac2_3).add(loc_btn_fac_fac2_4).add(loc_btn_fac_fac2_
5).add(loc_btn_fac_fac2_6).add(loc_btn_fac_fac2_7).add(loc_btn_fac_fac2_8)
markup_location_fac3 =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_fac3_1).add(loc_btn_fac_
fac3_2).add(loc_btn_fac_fac3_3).add(loc_btn_fac_fac3_4).add(loc_btn_fac_fac3_
5).add(loc_btn_fac_fac3_6)
markup_location_fac4 =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_fac4_1).add(loc_btn_fac_
fac4_2).add(loc_btn_fac_fac4_3).add(loc_btn_fac_fac4_4).add(loc_btn_fac_fac4_
```

```
5)
markup_location_farm =
telebot.types.InlineKeyboardMarkup().add(loc_btn_fac_farm_1).add(loc_btn_fac_
farm_2).add(loc_btn_fac_farm_3).add(loc_btn_fac_farm_4).add(loc_btn_fac_farm_
5).add(loc_btn_fac_farm_6).add(loc_btn_fac_farm_7).add(loc_btn_fac_farm_8)


@bot.message_handler(commands = ['start', 'subscribe'])
def welcome (message):
  bot.send_message(message.chat.id, "Ласкаво просимо до чат-бота
Буковинського Державного Медичного Університету. ", reply_markup =
markup_main)
  if (not db.subscriber_exists(message.from_user.id)):
    # если юзера нет в базе, добавляем его
    db.add_subscriber(message.from_user.id, )
  else:
    # если он уже есть, то просто обновляем ему статус подписки
    db.update_subscription(message.from_user.id, True)
  bot.send_message(message.chat.id, "Ви підписані. Будь ласка додайте свою
пошту (/update)")


# Команда отписки
@bot.message_handler(commands=['unsubscribe'])
def unsubscribe(message: types.Message):
  if (not db.subscriber_exists(message.from_user.id)):
    # если юзера нет в базе, добавляем его с неактивной подпиской
(запоминаем)
    db.add_subscriber(message.from_user.id, False)
    bot.send_message(message.chat.id,"Ви вже відписані від цього каналу.")
  else:
    # если он уже есть, то просто обновляем ему статус подписки
    db.update_subscription(message.from_user.id, False)
    bot.send_message(message.chat.id,"Ви відписались від цього каналу.")

@bot.message_handler(commands=['update'])
def update_email_f(message: types.Message):
  if (db.subscriber_exists(message.from_user.id)):
    bot.send_message(message.chat.id,"Будь ласка введіть свою університетську
пошту:")
    bot.register_next_step_handler_by_chat_id(message.chat.id,email)
  else:
    bot.send_message(message.chat.id,"Ви не підписані на канал, будь ласка
підпишіться.")
def email (message):
  email = message.text
  bot.send_message(message.chat.id, "Це ваша адреса:")
  bot.send_message(message.chat.id, text=email, reply_markup = markup_YN)

@bot.callback_query_handler(func = lambda call: call.data in ['button_Y',
'button_N'])
def email_accept(call):
  try:
    if call.message:
      if call.data == 'button_Y':
        bot.send_message(call.message.chat.id, "Пошта підтверджена.")
        db.update_email(call.message.chat.id, email = call.message.text)
      elif call.data == 'button_N':
        bot.send_message(call.message.chat.id, "Будь ласка введіть свою
університетську пошту:")
        bot.register_next_step_handler_by_chat_id(call.message.chat.id,
email)
    bot.edit_message_reply_markup(call.message.chat.id, message_id =
```

```
call.message.message_id, reply_markup='')
  except Exception as e:
    print(repr(e))


@bot.message_handler()
def main_buttons(message: types.Message):
  #Головне меню
  if (message.text == "Для студента"):
    bot.send_message(message.chat.id,"Що вас цікавить?", reply_markup =
markup_stud)
  elif (message.text == "Covid-19"):
    bot.send_photo(message.from_user.id, photo =
'https://shev.kyivcity.gov.ua/files/2020/4/21/2020_final_Coronavirus_640.jpg'
)
    bot.send_message(message.chat.id,"Наразі місто знаходиться в помаранчевій
зоні, в університеті працює масковий режим, якого студенти зобов'язані
дотримуватись для захисту себе та оточуючих. Всі лекційні заняття будуть
проводитись дистанційно, а практичні заннятя - очно. Бережіть себе та
дотримуйтесь правил безпеки.")
  elif (message.text == "Новини"):
    media_ids = get_lastposts(1090379724)
    print(media_ids)
    href = getInstagramUrlFromMediaId(media_ids)
    bot.send_message(message.chat.id, text = "Новини з нашого Інтсаграм
каналу \n" + href)
    bot.send_message(message.chat.id, text = "Новини на нашому сайті \n" +
"https://www.bsmu.edu.ua/novini-ta-podiyi/")
  elif (message.text == "Free time"):
    bot.send_message(message.from_user.id, text = "Виберіть варіант
проведення вільного часу.", reply_markup = (markup_frtm))
  elif (message.text == "Повернутись на головну"):
    bot.send_message(message.from_user.id,text = "Виберіть те що вас
цікавить", reply_markup = (markup_main))

#Для студента

  elif (message.text == "Корпуси"):
    bot.send_message(message.chat.id, text = 'Виберіть локацію яка вас
цікавить', reply_markup = (markup_loc_menu))
  elif (message.text == "Оплата за університет"):
    keyboard = types.InlineKeyboardMarkup()
    url_button = types.InlineKeyboardButton(text="Оплата за університет",
url="https://next.privat24.ua/payments/form/%7B%22token%22:%22599eb577a4500fd
d58d61485b3e302f0nkam8j2w%22%7D?lang")
    keyboard.add(url_button)
    bot.send_message(message.chat.id, "Натисніть на кнопку", reply_markup =
keyboard)
  elif (message.text == "Розклад"):
    bot.send_message(message.chat.id, text =
"https://www.bsmu.edu.ua/studentu/rozkladi-zanyat/")
  elif (message.text == "Журнал оцінок"):
    bot.send_message(message.chat.id, text = "https://eznew.bsmu.edu.ua/" +
"\n" + "Через тимчасові проблеми, синхронізація бота та електронного журналу
не працює в найближчому часі цю несправність буде виправлено, дякуємо за
терпіння")

#Корпуси

  elif (message.text == "Гуртожитки \U0001F3E2"):
    bot.send_message(message.chat.id, text='Виберіть гуртожиток який вас
цікавить', reply_markup=(markup_loc_hostels))
  elif (message.text == "Факультети \U0001F3DB"):
```

```
    bot.send_message(message.chat.id, text='Виберіть кафедру яка вас
цікавить', reply_markup=(markup_loc_kafedra))
  elif (message.text == "Головний корпус"):
    lat = db.get_location_lat(id_loc=1)
    long = db.get_location_long(id_loc=1)
    name = db.get_location_name(id_loc=1)
    href = db.get_location_info(id_loc=1)
    bot.send_message(message.chat.id, text=name)
    bot.send_message(message.chat.id, text=href)
    bot.send_location(message.chat.id, latitude=lat, longitude=long)
  elif (message.text == "Навчальний корпус"):
    lat = db.get_location_lat(id_loc=5)
    long = db.get_location_long(id_loc=5)
    name = db.get_location_name(id_loc=5)
    href = db.get_location_info(id_loc=5)
    bot.send_message(message.chat.id, text=name)
    bot.send_message(message.chat.id, text=href)
    bot.send_location(message.chat.id, latitude=lat, longitude=long)

#Вільний час

  elif (message.text == "Фільм на вечір \U0001F3A5"):
    num = random.randrange(0, 249, 1)
    href = ia.get_top250_movies()
    href = href[num]
    id_href = ia.get_imdbID(href)
    bot.send_message(message.chat.id,text =
"https://www.imdb.com/title/tt"+id_href)
  elif (message.text == "Серіал на вечір \U0001F4FA"):
    num = random.randrange(0, 249, 1)
    href = ia.get_top250_tv()
    href = href[num]
    id_href = ia.get_imdbID(href)
    bot.send_message(message.chat.id, text="https://www.imdb.com/title/tt" +
id_href)
  elif (message.text == "Книга на вечір \U0001F4D6"):
    num = random.randrange(1, 122, 1)
    book_name = str(db.get_book_name(num))
    book_auther = str(db.get_book_auther(num))
    book_name = re.sub(r"[()\"',]",'',book_name)
    book_auther = re.sub(r"[()\"',]",'',book_auther)
    bot.send_message(message.chat.id, text = "Назва книги:
"+'"'+book_name+'"' +'\n'+ "Автор: " + book_auther)
  elif (message.text == "Гра в питання"):
    bot.send_game(chat_id=message.chat.id, game_short_name='bsmu_exam_game')
  elif (message.text == "Куди сходити вечером \U0001F389"):
    url_event = str(db.get_event(0))
    url_event_cor = re.sub(r"[()\"',]",'', url_event)
    url_event_image = db.get_event_img(0)
    bot.send_message(message.chat.id, text = url_event_image)
    bot.send_message(message.chat.id, text=url_event_cor)


  db.update_date(id_us=message.from_user.id, last_date=datetime.today())

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_hostel_1','loc_btn_hostel_2','loc_btn_hostel_3','loc_btn_hostel_4',
'loc_btn_hostel_5','loc_btn_hostel_6','loc_btn_hostel_7','loc_btn_hostel_8',]
)
def inline_location_hostels(call):
  try:
    if call.message:
      if call.data == 'loc_btn_hostel_1':
```

```
        lat = db.get_location_lat(id_loc = 4)
        long = db.get_location_long(id_loc = 4)
        name = db.get_location_name(id_loc = 4)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      elif call.data == 'loc_btn_hostel_2':
        lat = db.get_location_lat(id_loc = 10)
        long = db.get_location_long(id_loc = 10)
        name = db.get_location_name(id_loc=10)
        bot.send_message(call.message.chat.id, text= name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      elif call.data == 'loc_btn_hostel_3':
        lat = db.get_location_lat(id_loc = 9)
        long = db.get_location_long(id_loc = 9)
        name = db.get_location_name(id_loc = 9)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      elif call.data == 'loc_btn_hostel_4':
        lat = db.get_location_lat(id_loc = 8)
        long = db.get_location_long(id_loc = 8)
        name = db.get_location_name(id_loc = 8)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      elif call.data == 'loc_btn_hostel_5':
        lat = db.get_location_lat(id_loc = 3)
        long = db.get_location_long(id_loc = 3)
        name = db.get_location_name(id_loc = 3)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      elif call.data == 'loc_btn_hostel_6':
        lat = db.get_location_lat(id_loc = 2)
        long = db.get_location_long(id_loc = 2)
        name = db.get_location_name(id_loc = 2)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      elif call.data == 'loc_btn_hostel_7':
        lat = db.get_location_lat(id_loc = 7)
        long = db.get_location_long(id_loc = 7)
        name = db.get_location_name(id_loc = 7)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      elif call.data == 'loc_btn_hostel_8':
        lat = db.get_location_lat(id_loc = 6)
        long = db.get_location_long(id_loc = 6)
        name = db.get_location_name(id_loc = 6)
        bot.send_message(call.message.chat.id, text = name)
        bot.send_location(call.message.chat.id, latitude = lat, longitude =
long)
      bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_stomat','loc_btn_fac_fac1','loc_btn_fac_fac2','loc_btn_fac_fac3
','loc_btn_fac_fac4','loc_btn_fac_farm','loc_btn_fac_inst',])
```

```
def inline_location_kafedra(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_stomat':
        bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
        bot.send_message(call.from_user.id, text="Стоматологічний")
        bot.send_message(call.from_user.id, text="Виберіть факультет",
reply_markup=(markup_location_stomat))
      elif call.data == 'loc_btn_fac_fac1':
        bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
        bot.send_message(call.from_user.id, text="Факультет №1")
        bot.send_message(call.from_user.id, text="Виберіть факультет",
reply_markup=(markup_location_fac1))
      elif call.data == 'loc_btn_fac_fac2':
        bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
        bot.send_message(call.from_user.id, text="Факультет №2")
        bot.send_message(call.from_user.id, text="Виберіть факультет",
reply_markup=(markup_location_fac2))
      elif call.data == 'loc_btn_fac_fac3':
        bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
        bot.send_message(call.from_user.id, text="Факультет №3")
        bot.send_message(call.from_user.id, text="Виберіть факультет",
reply_markup=(markup_location_fac3))
      elif call.data == 'loc_btn_fac_fac4':
        bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
        bot.send_message(call.from_user.id, text="Факультет №4")
        bot.send_message(call.from_user.id, text="Виберіть факультет",
reply_markup=(markup_location_fac4))
      elif call.data == 'loc_btn_fac_farm':
        bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
        bot.send_message(call.from_user.id, text="Фармацевтичний")
        bot.send_message(call.from_user.id, text="Виберіть факультет",
reply_markup=(markup_location_farm))
      elif call.data == 'loc_btn_fac_inst':
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
        bot.send_message(call.from_user.id, text="Інститут післядипломної
освіти")
        bot.send_message(call.from_user.id, text="Виберіть факультет",
reply_markup=(markup_location_inst))
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_stomat_1','loc_btn_fac_stomat_2','loc_btn_fac_stomat_3','loc_bt
n_fac_stomat_4','loc_btn_fac_stomat_5','loc_btn_fac_stomat_6','loc_btn_fac_st
omat_7','loc_btn_fac_stomat_8','loc_btn_fac_stomat_9',])
def inline_location_stomat(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_stomat_1':
        i=14
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
```

```
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_2':
        i = 23
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_3':
        i = 25
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_4':
        i = 37
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_5':
        i = 44
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_6':
        i = 47
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_7':
        i = 51
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
```

```
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_8':
        i = 60
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_stomat_9':
        i = 61
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_inst_1','loc_btn_fac_inst_2','loc_btn_fac_inst_3','loc_btn_fac_
inst_4','loc_btn_fac_inst_5','loc_btn_fac_inst_6',])
def inline_location_inst(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_inst_1':
        i=15
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_inst_2':
        i = 20
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_inst_3':
        i = 24
        lat = db.get_location_lat(id_loc=i)
```

```
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_inst_4':
        i = 35
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_inst_5':
        i = 36
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_inst_6':
        i = 46
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_fac1_1','loc_btn_fac_fac1_2','loc_btn_fac_fac1_3','loc_btn_fac_
fac1_4','loc_btn_fac_fac1_5','loc_btn_fac_fac1_6','loc_btn_fac_fac1_7',])
def inline_location_fac1(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_fac1_1':
        i=11
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac1_2':
        i = 19
```

```
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac1_3':
        i = 33
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac1_4':
        i = 43
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac1_5':
        i = 45
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac1_6':
        i = 57
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac1_7':
        i = 59
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
```

```
call.message.message_id, reply_markup = '')
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_fac2_1','loc_btn_fac_fac2_2','loc_btn_fac_fac2_3','loc_btn_fac_
fac2_4','loc_btn_fac_fac2_5','loc_btn_fac_fac2_6','loc_btn_fac_fac2_7','loc_b
tn_fac_fac2_8',])
def inline_location_fac2(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_fac2_1':
        i=12
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac2_2':
        i = 18
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac2_3':
        i = 29
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac2_4':
        i = 30
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac2_5':
        i = 38
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
```

```
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac2_6':
        i = 40
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac2_7':
        i = 52
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac2_8':
        i = 58
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_fac3_1','loc_btn_fac_fac3_2','loc_btn_fac_fac3_3','loc_btn_fac_
fac3_4','loc_btn_fac_fac3_5','loc_btn_fac_fac3_6',])
def inline_location_fac3(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_fac3_1':
        i=28
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac3_2':
        i = 41
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
```

```
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac3_3':
        i = 48
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac3_4':
        i = 49
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac3_5':
        i = 53
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac3_6':
        i = 27
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_fac4_1','loc_btn_fac_fac4t_2','loc_btn_fac_fac4_3','loc_btn_fac
_fac4_4','loc_btn_fac_fac4_5',])
def inline_location_fac4(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_fac4_1':
        i=14
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
```

```
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac4_2':
        i = 23
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac4_3':
        i = 25
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac4_4':
        i = 37
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_fac4_5':
        i = 44
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
  except Exception as e:
    print(repr(e))

@bot.callback_query_handler(func = lambda call: call.data in
['loc_btn_fac_farm_1','loc_btn_fac_farm_2','loc_btn_fac_farm_3','loc_btn_fac_
farm_4','loc_btn_fac_farm_5','loc_btn_fac_farm_6','loc_btn_fac_farm_7','loc_b
tn_fac_farm_8',])
def inline_location_farm(call):
  try:
    if call.message:
      if call.data == 'loc_btn_fac_farm_1':
        i=16
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
```

```
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_farm_2':
        i = 17
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_farm_3':
        i = 21
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_farm_4':
        i = 31
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_farm_5':
        i = 32
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_farm_6':
        i = 50
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_farm_7':
```

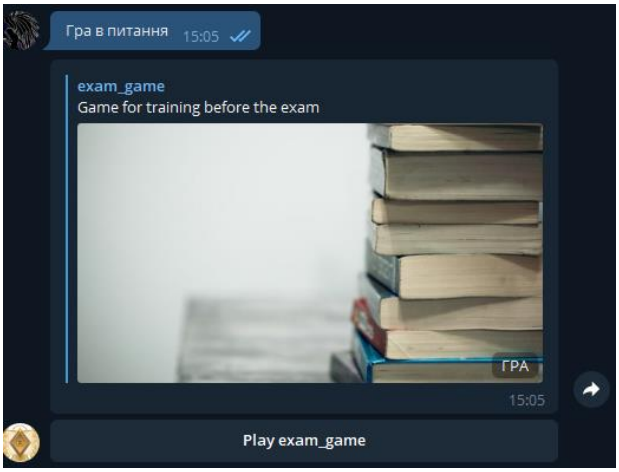
```
        i = 55
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
      elif call.data == 'loc_btn_fac_farm_8':
        i = 56
        lat = db.get_location_lat(id_loc=i)
        long = db.get_location_long(id_loc=i)
        name = db.get_location_name(id_loc=i)
        href = db.get_location_info(id_loc=i)
        bot.send_message(call.message.chat.id, text=name)
        bot.send_message(call.message.chat.id, text=href)
        bot.send_location(call.message.chat.id, latitude=lat, longitude=long)
        bot.edit_message_reply_markup(call.message.chat.id, message_id =
call.message.message_id, reply_markup = '')
  except Exception as e:
    print(repr(e))


@bot.callback_query_handler(func = lambda callback_query:
callback_query.game_short_name == 'bsmu_exam_game')
def game(call):
  bot.answer_callback_query(callback_query_id = call.id, url = 'http://exam-
game.test')
  return


@bot.callback_query_handler(func = lambda call: call.data in ['button1',
'button2', 'button3'])
def inline_curse(call):
  try:
    if call.message:
      if call.data == 'button1':
        bot.send_message(call.message.chat.id, "Нажаль для першого курсу
розкладу ще не має.")
      elif call.data == 'button2':
        bot.send_message(call.message.chat.id, "https://www.bsmu.edu.ua/wp-
content/uploads/2020/08/wlec_fak%E2%84%962_kurs2.pdf")
      elif call.data == 'button3':
        bot.send_message(call.message.chat.id, "https://www.bsmu.edu.ua/wp-
content/uploads/2020/08/wlec_fak%E2%84%961_kurs3.pdf")
      bot.edit_message_reply_markup(call.message.chat.id,
message_id=call.message.message_id, reply_markup='')
  except Exception as e:
    print(repr(e))


bot.polling(none_stop = True)
```

Додаток 3.1(код SQL запитів до бази)

```
import sqlite3
from datetime import datetime


class SQLighter:

  def __init__(self, database_file):
    """Підключаємось до бази даних"""
    self.connection = sqlite3.connect(database_file, check_same_thread =
False)
    self.cursor = self.connection.cursor()

  def get_subscriptions(self, status = True):
    """Отримуємо активних користувачів"""
    with self.connection:
      return self.cursor.execute("SELECT * FROM `subscriptions` WHERE
`status` = ?", (status,)).fetchall()

  def subscriber_exists(self, user_id):
    """Перевіряємо чи є юзер в базі"""
    with self.connection:
      result = self.cursor.execute("SELECT * FROM `subscriptions` WHERE
`user_id` = ?", (user_id,)).fetchall()
      return bool(len(result))

  def add_subscriber(self, user_id, status = True, sub_date =
datetime.today()):
    """Додаємо нового користувача"""
    with self.connection:
      return self.cursor.execute("INSERT INTO `subscriptions` (`user_id`,
'status','sub_date') VALUES(?,?,?)",(user_id, status, sub_date,))


  def update_subscription(self, user_id, status):
    """Оновлюємо статус підкписки користувача"""
    with self.connection:
      return self.cursor.execute("UPDATE `subscriptions` SET `status` = ?
WHERE `user_id` = ?", (status, user_id,))

  def update_email (self, id_us, email, last_date= datetime.today()):
    """Додаємо пошту користувача в базу"""
    with self.connection:
      return self.cursor.execute("INSERT INTO `user_data` (email, id_us,
last_date)VALUES(?,?,?)", (email, id_us, last_date))

  def update_date (self, id_us, last_date):
    """Перевріям коли користувач останній раз заходив"""
    with self.connection:
      return self.cursor.execute("UPDATE user_data SET last_date = ? WHERE
id_us = ?",( last_date, id_us,))

  def email_exists(self, user_id):
    """Перевіряємо чи є користувач в базі"""
    with self.connection:
      result = self.cursor.execute("SELECT * FROM `subscriptions` WHERE
`user_id` = ?", (user_id,)).fetchall()
      return bool(len(result))
```

```
  def free_time (self, id_g):
    with self.connection:
      return self.cursor.execute("SELECT genre FROM genre_db WHERE id_g = ?
",(id_g,)).fetchone()

  def get_location_lat (self, id_loc):
    """Витягуємо широту"""
    with self.connection:
      return self.cursor.execute("SELECT latitude FROM locations WHERE id_loc
= ? ", (id_loc,)).fetchone()

  def get_location_long (self, id_loc):
    """витягуємо довготу"""
    with self.connection:
      return self.cursor.execute("SELECT longitude FROM locations WHERE
id_loc = ?", (id_loc,)).fetchone()

  def get_location_info(self, id_loc):
    """отримуємо посиланняна на сайт кафедри"""
    with self.connection:
      return self.cursor.execute("SELECT href FROM locations WHERE id_loc =
?", (id_loc,)).fetchone()

  def get_location_name(self, id_loc):
    """Отримуємо назву локації"""
    with self.connection:
      return self.cursor.execute("SELECT loc_name FROM locations WHERE id_loc
= ?", (id_loc,)).fetchone()

  def get_book_name(self, id_b):
    """Отримуємо назву книги """
    with self.connection:
      name = self.cursor.execute("SELECT name_b FROM books WHERE id_b = ?",
(id_b,)).fetchone()
      return (name)

  def get_book_auther(self, id_b):
    """Отримуємо ПІБ автора книги"""
    with self.connection:
      auther = self.cursor.execute("SELECT auther_b FROM books WHERE id_b =
?", (id_b,)).fetchone()
      return (auther)

  def get_event(self, id):
    """Назву заходу """
    with self.connection:
      return self.cursor.execute("SELECT night_party FROM after_univ WHERE id
= ?", (id,)).fetchone()

  def get_event_img(self, id):
    """Посилання на захід"""
    with self.connection:
      return self.cursor.execute("SELECT url FROM after_univ WHERE id = ?",
(id,)).fetchone()

  def close(self):
    """Закрываем соединение с БД"""
    self.connection.close()
```

Додаток 5.3 (Гра в питання)



(Рис. 1)