НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

Фаховий молодший бакалавр
(освітньо-професійний ступінь)

на тему:_____ Розробка інформаційного Telegram-бота ІТ коледжу Львівської політехніки_____

_____

Виконав студент IV курсу, групи ***ОК- 41***
ОПП *«Обслуговування комп'ютерних систем та мереж»*
Спеціальності *123 Комп'ютерна інженерія*
Матвійчук Віталій Володимирович
(прізвище, ім'я по батькові)

Керівник _____ *Андрій Селемонавічус*
(підпис) (ім'я прізвище)

Нормоконтролер _____ *Любомира Кужій*
(підпис) (ім'я прізвище)

Рецензент _____
(підпис) (ім'я прізвище)

Голова ЕК _____ *Олег Гіщак*
(підпис) (ім'я прізвище)

Члени ЕК _____ *Любомира Кужій*
(підпис) (ім'я прізвище)

_____ *Андрій Селемонавічус*
(підпис) (ім'я прізвище)

Дипломний проєкт захищений в ЕК « __ » _____2025 р.

з оцінкою « _____ »

Львів 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Циклова комісія	***Комп'ютерних систем і мереж***
Освітньо-професійний ступінь	***Фаховий молодший бакалавр***
Освітньо-професійна програма	***Обслуговування комп'ютерних систем та мереж***
Спеціальність	***123 Комп'ютерна інженерія***

ЗАТВЕРДЖУЮ
Завідувач відділення
«Комп'ютерних систем і мереж»
_____ Володимир СТАХІВ
«____» _____ 2025 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Матвійчуку Віталію Володимировичу
(прізвище, ім'я та по батькові)

1. Тема проєкту Розробка інформаційного Telegram-бота ІТ коледжу Львівської політехніки

керівник проєкту *Селемонавічус Андрій Альвідасович*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом директора від «*20*» *березня 2025* року № *20 - ст*

2. Строк подання студентом проєкту «*10*» *червня* 2025 року

3. Вихідні дані до проєкту

3.1 В якості мови програмування використати мову C#;

3.2 В якості інформаційної системи використати Telegram-бот;

3.3 В якості інтегрованого середовища розробки використати. MicrosoftVisualStudioCommunity

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити

4.1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

4.2 . РОЗРОБКА ІНФОРМАЦІЙНОГО ЧАТ-БОТА

4.3 ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ІНСТРУКЦІЇ ЩОДО ЇЇ ВИКОРИСТАННЯ

4.4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

5. Перелік графічного матеріалу

5.1.	Демонстрація моделі роботи технологій LongPolling та Webhooks	
5.2.	Схема взаємодії користувача з чат-ботом на клієнтському рівні	
5.3.	Клієнтський інтерфейс взаємодії з чат-ботом у різних представленнях	
5.4.	Витрати на розробку та впровадження проектного рішення	

6 Консультанти розділів проєкту

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		Завдання видав	Завдання отримав
Техніко-економічне обґрунтовування	*Тетяна Підкуймуха*		
Охорона праці та безпека життєдіяльності	*Роман Томків*		

7. Дата видачі завдання « *01*»*квітня 2025* року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту	Термін виконання	Примітка
1	Аналіз предметної області	15.03.2025	
2	Вибір засобів розробки	30.03.2025	
3	Розробка інформаційного чат-бота	15.04.2025	
4	Розробка програми тестування інформаційної системи	25.04.2025	
5	Розробка демонстаційних креслень	01.05.2025	
6	Охорона праці	15.05.2025	
7	Техніко – економічне обгрунтування	25.05.2025	
8			

Студент _____ Віталій Матвійчук
(підпис) (ім'я, прізвище)

Керівник проєкту _____ Андпій Селемонавічус
(підпис) (ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка дипломного проекту: 83 стор., 35 рисунків, 4 таблиці, 12 джерел, 1 додаток.

Об'єкт проектування – чат-бот у якості інформаційної системи

Мета роботи – розробка інформаційного чат-боту в месенджері Telegram, призначеного для збереження, обробки та вибірки необхідної інформації у зручній формі для викладачів та студентів коледжу.

Галузь використання – інформаційні системи

ЧАТ-БОТ, МЕСЕНДЖЕРИ, TELEGRAM, ВЕБ-ЗАСТОСУНОК, СИСТЕМИ МИТТЄВОГО ОБМІНУ ПОВІДОМЛЕННЯМИ, ІНТЕРНЕТ, C#

ЗМІСТ

стор.

ВСТУП

Чат-бот - це віртуальний асистент, який спілкується з користувачами за допомогою повідомлень і може мати безліч специфічних функцій. Чат-боти можна використовувати як для розсилки інформації, так і для її збору та обробки.

Найбільшу популярність чат-боти отримали, коли почалося їх використання в месенджерах і соціальних мережах (наприклад, в Telegram, Viber, Facebook).

Telegram-бот – це аккаунт Telegram, різновид чат-бота, керований програмним забезпеченням, вони можуть інтегруватися з іншими службами або навіть передавати команди інших сервісів.

Типові Telegram-боти відповідають на спеціальні команди в персональних і групових чатах, також вони можуть здійснювати пошук в інтернеті або виконувати інші завдання.

Таким чином, сфера застосування Telegram-ботів безмежна. Компанія Uber, безліч банків, великі магазини і багато інших організацій використовують Telegram-ботів для спрощення та автоматизації внутрішніх робочих процесів.

Метою цієї дипломної роботи є створення Telegram-бота для коледжу телекомунікацій та комп’ютерних технологій, з функціями вибірки інформації про поточне знаходження студентів та викладачів під час навчальних годин на основі загальнодоступного розкладу занять. Окрім цього планується запровадити і інші функції, які можуть знадобитись студентам та викладачам коледжу.

Актуальність цієї дипломної роботи зумовлена швидким зростом популярності месенджерів і таких засобів автоматизації як чат-боти серед користувачів мережі інтернет.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

На сьогоднішній день, якщо грамотно використовувати технологію чат-ботів, можна реалізувати повноцінні магазини в месенджерах, технічну підтримку, і навіть внутрішні корпоративні аналітичні інструменти (робота з внутрішньою інформацією компанії).

1.1 Доцільність використання чат-ботів

Існує тип завдань і проблем, з якими чат-боти справляються краще за все. Чат-боти можуть взяти на себе ряд таких завдань: сегментація користувачів, оптимізація одноманітних завдань, цілодобова і безперебійна підтримка користувачів, продажі, виключення людського фактору, інтеграція інформаційних систем та ін.

1.1.1 Сегментація користувачів

Якщо існує кілька пропозицій для різних цільових аудиторій, відмінно працює бот, який проводить опитування і сегментує підписників в залежності від реакції на той чи інший контент. Ці дані використовуються, щоб в подальшому автоматично проводити внутрішню сегментацію підписників і розсилати максимально індивідуальні пропозиції.

Наприклад, https://t.me/amicoIHT_bot - чат-бот виробника БАДів і мережі медичних консультаційних центрів. Цей бот задає питання, які виявляють болю аудиторії і дають можливість підібрати потрібні товари з асортименту.

Бот запитує про вагу, зріст, вік, режим харчування, сну і інше. Варіанти відповідей він записує в «ярлики», за якими потім проводиться тематична розсилка. Наприклад, пропозиція з курсом лікування від гіпертонії йде тільки користувачам із зайвою вагою старше 40 років і малорухливим способом життя, а курс підвищення імунітету тільки користувачам, які хворіли ГРЗ більше 3 разів за

останні півроку. У будь-який момент менеджер може перехопити спілкування у бота і продовжити його у тому ж самому діалозі, без усіляких незручностей для користувача.

1.1.2 Оптимізація одноманітних завдань

До таких завдань належить, наприклад, перевірка відповідності документу від користувача прийнятим нормам. Чат-бот може сам перевірити документ, знайти помилки, написати користувачеві і попросити його виправити рахунок і повідомити менеджеру, коли рахунок потрібного формату буде готовий. Точно так само чат-бот може обробляти заяви на відпустку, збирати у співробітників звіти про відпрацьовані години і зводити їх в таблицю.По суті, будь-яка операція, яка проводиться по строго окресленому алгоритму і не вимагає від співробітників креативних рішень може бути доручена чат-боту.

1.1.3 Цілодобова і безперебійна підтримка користувачів

Забезпечення зворотнього зв'язку с користувачами 24 години на добу в будь-яких каналах. Це, мабуть, найочевидніша перевага чат-ботів. Чат-бот буде працювати доки працює мессенджер і сервер, на якому розміщений сам чат-бот. Очевидна вигода з цього – це надійність та економія коштів.

1.1.4 Продажі

На відміну від додатків або сайтів, в месенджерах спілкування ведеться за допомогою діалогу, і людям не потрібно вивчати новий інтерфейс. Більшість великих месенджерів вже випустили або планують випустити рішення, що дозволяють, не виходячи з чату, здійснювати безпечну оплату товарів і послуг. Тому месенджери стають новим каналом продажів і маркетингу, а інструментом для цього каналу якраз і є чат-боти.

1.1.5 Виключення людського фактору

Виключення людського фактору з комунікації. Ця функція чат-ботів вже показала себе в HR. Вони беруть на себе роль анонімного і безосібногго співрозмовника, щоб прибрати з процесу зайві емоції. Це стосується і таких завдань, як збір зворотного зв'язку, примус співробітників до своєчасної здачі звітів, щорічна оцінка персоналу і так далі. Компанія Amazon вже кілька років використовує роботів для таких задач.

1.1.6 Інтеграція інформаційних систем

Чат-боти легко інтегруються з будь-якими інформаційними системами. Це означає, що співробітники можуть спілкуватися з усіма цими системами через чат-бота. Їм не потрібно навчатися використанню різних інтерфейсів і витрачати час на введення або пошук даних в різних системах.

Все частіше люди використовують месенджери як найшвидший і простий спосіб спілкуватися в середовищі, де немає нічого зайвого. І тепер, таким чином, можна комунікувати не тільки з друзями, але і з брендами, отримувати доступ до найбільш необхідних послуг, минаючи пошукові системи і сайти

Чат-бот може виконувати роботу асистента - аналізувати дані, створювати звіти, заповнювати форми, задаючи власнику навідні запитання. Цими здібностями ботів користуються, наприклад, фінансисти, готуючись до нарад, або рекрутери, використовуючи чат-ботів для того, щоб кандидати могли в режимі діалогу заповнювати форми і не кидали цей процес.

1.2 Сфери застосування чат-ботів

На рис. 1.1 зображений ландшафт індустрії використання чат-ботів, створений американським сервісом KeyReply у 2017 році.

Рисунок 1.1 – Ландшафт індустрії використання чат-ботів

Компанії розподілені по осях залежно від функцій, які виконують їх боти (з правої сторони – підтримка, з лівої - маркетинг), і від того, де вони створені (знизу - в самій компанії, зверху - у студії на замовлення).

Коло в центрі - це популярні месенджери, які дозволяють створювати ботів через надійні API-інтерфейси.Наступне коло Brands - компанії, які запустили чат-ботів вказаного типу.

Providers - компанії, які допомагають брендам створювати ботів, а Tools - інструменти, якими користуються провайдери, бренди або розробники для запуску чат-ботів.

Сфери застосування чат-ботів необмежені, проте найбільш популярні з них: сфера продажів, служби підтримки, служби доставки, таксі, засоби масової інформації, банківська сфера

1.2.1 Сфера продажів

Чат-боти можуть створюватися для автоматизованого продажу квитків (у кіно, театр, на концерт, на транспорт), одягу, косметики та інших. Чудовим прикладом запровадження такого чат-бота в Україні є telegram-бот RailWayBot. Він пропонує купити квитки приблизно так само, як і на офіційному сайті Укрзалізниці, проте, якщо квитки на необхідний напрямок відсутні, цей бот дозволяє провести моніторинг. Кожні пять хвилин він буде провіряти наявність квитків, і якщо хтось здасть білет, бот надсилає користувачеві повідомлення про те, що квиток тепер доступний до продажу.

1.2.2 Служби підтримки

Використання чатботів для служби підтримки - повсюдно впроваджувана практика. Автоматизація процесів спілкування з клієнтами допомагає заощадити безліч часу і сил «живих» працівників компанії і виділити їм більше часу на вирішення складних завдань.

Чат-бот “Зоряна” мобільного оператора Київстар щомісяця заощаджує близько 20 тисяч доларів, обробляючи всього 10% запитів, які надходять в компанію. Щомісяця чат-ботом “Зоряна” користуються більше 100 000 користувачів в месенджерах Facebook , Telegram і Viber.

Схожі чат-боти використовує Державна міграційна служба України, Нова Пошта, оператор Lifecell, Укрпошта та безліч інших великих українських компаній.

Автоматичні співрозмовники також можуть проводити аналіз статистики розмов і знаходити зони в системах компанії, які найчастіше створюють проблеми для користувачів.

1.2.3 Служби доставки та таксі

Чатботи по доставці (квітів, подарунків і т.д.) користуються у клієнтів дуже великою популярністю. І пальму першості тут тримає служба доставки їжі.

Наприклад, у піцерії Domino є свій чатбот. Щоб замовити піцу, потрібно ввести в чаті слово pizza, і бот на основі історії повідомлень автоматично підбере бажаний користувачем тип цієї італійської страви. Якщо замовник хоче якусь конкретну піцу, йому потрібно буде вказати свої переваги в налаштуваннях. Звичайно ж, перед цим необхідно ввести свій номер телефону та адресу.

Чатботи можна використовувати і службам виклику таксі, - вони роблять процес замовлення і очікування машини зручнішим для клієнтів. Популярний зараз сервіс Uber використовує чат-бот в Facebook Messenger: за допомогою нього клієнт може викликати таксі і простежити за траєкторією його руху, - так він буде знати певно, що за ним їдуть і скільки часу залишилося чекати.Слідом за Uber чат ботів розпочали відкривати і інші компанії з цієї сфери, також з’явились боти-агрегатори таксі.

1.2.4 Засоби масової інформації

Чатботи використовуються багатьма газетами, журналами і телеканалами. Наприклад, бот від CNN показує статті з обраної користувачем теми, а при підписці щодня радує читача новинами і свіжими публікаціями.

Також у Telegram з’явився бот MediaMonitoringBot, який моніторить згадки в українських інтернет-виданнях. Він повідомляє користувача про згадки в медіа не пізніше, ніж за п'ять хвилин після появи матеріалу.

Щоб запустити моніторинг, потрібно додати ключове слово, за яким користувач хоче відстежувати згадки. Наприклад, назва компанії чи бренду, прізвище засновника і т.д. У разі виходу нового матеріалу з ключовим словом, бот надішле повідомлення про це. У повідомленні буде зазначено, яке

саме ключове слово було згадано, посилання на матеріал, кількість переглядів та потенційний прогноз охоплення публікації.

1.2.5 Банківська сфера

У банківській сфері чат-боти використовують найбільше. Банки дають можливість в своїх чат-ботах шукати банкомати і відділення, які знаходяться поруч, відстежувати курси валют та ін.

“Приватбанк” має 6 різних чат-ботів. Вони дозволяють перечислювати гроші безпосередньо через месенджер з прив’язкою до Приват24, приймають заявки, дозволяють оформлювати кредити та проводять консультацію користувачів.

По заявленням експертів, чат-боти «ПриватБанка» можуть обробляти близько 85% звернень, що дозволило вкоротити кількість операторів і дозволяє економити близько $2,2 млн на рік.

Також чат-ботами обзавелись такі українські банки як: “Альфа-банк”, “Райффайзен-банк”, “Таскомбанк”, “ОТП-Банк”, “Ощадбанк” та інші.

1.3 Класифікація чат-ботів

1.3.1 Класифікація чат-ботів за складністю

1. Прості. Функціонують на основі правил. Такий тип чат-ботів, як правило, реагує на точні і конкретні команди. Якщо ж послати невірний або некоректний запит, то система не зможе його зрозуміти.

2. Складні. Функціонують на основі машинного навчання. даний тип володіє штучним інтелектом. Штучний інтелект – це науковий напрям, в рамках якого ставляться і вирішуються завдання апаратного або програмного моделювання тих видів людської діяльності, які традиційно вважаються інтелектуальними .

Працюючи з ним, вже немає необхідності максимально точно і конкретно писати свій запит, так як системи другого типу розуміють не тільки вузькі команди, але і саму природню мову. В результаті спілкування з людиною, такий чат-бот стає розумнішими.

1.3.2 Класифікація чат-ботів за функціоналом

1. Комунікаційні чат-боти. До них відносяться всі боти, які виступають в ролі консультантів і спілкуються з клієнтами від імені компанії, або ж внутрішньокорпоративні боти, які структурують і спрощують процес комунікації між співробітниками. Також до них відносяться боти, які виконують маркетинг-функції.
2. Функціональні чат-боти. До них відносяться всілякі боти-помічники, в тому числі - розважальні. Їх функції безмежні: знайти, забронювати, замовити, купити, порівняти, підібрати, розважати, навчати та інше. Бот, створений у рамках цього дипломного проекту, відноситься саме до цього виду.

1.3.3 Класифікація чат-ботів за інтерфейсом

1. Кнопкові чат-боти. Прості в освоєнні, але малофункціональні. Адже якщо зробити панелі зі складними, розгалуженими меню, розібратися в них буде зовсім нелегко. У цих ботів ініціатором діалогу виступає бот, а користувач для отримання необхідної йому допомоги, зобов'язаний послідовно виконувати вимоги свого робота-співрозмовника.

2. Розмовні (текстові) боти. Вони реагують на репліки співрозмовника і можуть вести з ним діалог. Розмовні чат-боти можуть бути засновані на штучному інтелекті або на мовній моделі та правилах.

Розмовні чат-боти з штучним інтелектом здатні навчатися. Але для цього потрібні величезні обсяги спеціально підготовлених даних, зібрати які під силу далеко не кожному банку. Тому через недостатність навчання такі чат-боти

можуть вести себе непередбачувано, неправильно відповідати на питання, невірно оцінювати наміри клієнта і т.п.

3. Комбіновані чат-боти. Виходячи з усього вищесказаного, очевидно, що оптимальний варіант бота – це “Гібрид”, у якому, наприклад, частина функцій винесена на кнопки, але при цьому вони паралельно ведуть діалог зі співрозмовником і виконують текстові команди. До цього виду відноситься бот, створений у рамках цього дипломного проекту.

1.3.4 Класифікація чат-ботів за платформоювпровадження

1. Чат боти, впроваджені у месенджерах та соціальних мережах (Telegram, Facebook Messenger, Viber, VK)
2. Чат боти, впроваджені у мобільному застосунку (FakeTalk, Neiron, SimSimi)
3. Чат боти, впроваджені на веб-сайтах (при вході на безліч сайтів відкривається вікно чату, де користувача консультує бот, а не оператор підтримки.)
4. Чат боти, що відповідають на смс-запити (СМС-служби банків і мобільних операторів, впроваджені найдавніше, на сьогоднішній день поступово відходять на другий план)

1.4 Опис засобів розробки

1.4.1 Середовище розробки Microsoft VisualStudio 2019 Community

Microsoft VisualStudio 2019 Community – це спеціальна безкоштовна версія Microsoft VisualStudio для використання в навчальних цілях, для розробки проектів з відкритим кодом та у п’яти копіях для використання в некорпоративних організаціях з річним доходом меншим, ніж 1 млн. доларів США, та кількістю ПК, меншою ніж 250 одиниць. Також існують версії

Professional та Enterprise, проте вони досить дороговартісні і жодна з її переваг не зможе знайти застосування у цьому проекті.

Основні переваги цього IDE:

-Community версія є абсолютно безкоштовною і не має термінів дії

-Вбудована підтримка ASP.NET, Javascript, Python, C#/, C++, F#, Java, TypeScript, VisualBasic.

-Підказки IntelliSence дозволяють набагато швидше і зручніше писати код, також є підказки, що вказують на рекомендовані дії по вдосконаленню існуючого коду, наприклад пропонують перейменувати функцію, створити параметр, відформатувати фрагмент коду тощо.

-CodeLens допомагає легко отримувати важливі аналітичні відомості, наприклад про внесені в код зміни, про їх наслідки та про модульне тестування методів.

-Зручні вбудовані інструменти налагодження.

-Зручний інтерфейс для роботи з наборами тестів допомагає впорядковувати дані, аналізувати обсяг тестованого коду і миттєво бачити результати. Функції тестування коду прямо під час введення дозволяють швидко дізнаватися наслідки кожної внесеної зміни.

-Вбудовані функції роботи з системами контролю версій

1.4.2 Мова програмування C#

C# - це об'єктно-орієнтована мова програмування, розроблена в 1998-2001 роках. Синтаксис C# відноситься до C-подібних, найбільше він схожий на Java і C++.Основні особливості C#:статична типізація, підтримка поліморфізму і перевантаження операторів, асинхронні методи, події, узагальнена коваріантність і контрваріантність, атрибути, делегати, властивості,автовластивості,кортежі, ітератори, узагальнені типи і методи, анонімні типи, методи і функції.

Є кілька існуючих реалізацій C#:

-Реалізація C# у складі .NETFramework;

-Реалізація C# у складі проекту Rotor (SharedSourceCommonLanguageInfrastructure)компанії Microsoft;

-Реалізація C# у складі проектуMono;

-Реалізація C# у складі проекту DotGNU

-Реалізація C# у складі проекту DotNetAnyWhere.

Оскільки ця робота проводиться у середовищі Microsoft VisualStudio, то у ній використовується реалізація у вигляді компілятора, включена до складу .NET Framework версії 4.8.

2 РОЗРОБКА ІНФОРМАЦІЙНОГО ЧАТ-БОТА

2.1 Реєстрація чат-бота у Telegram

Для розробки Telegram-бота для початку його необхідно зареєструвати у сервісі Telegram. Для цього існує спеціальний Telegram-бот під назвою «BotFather». Приклад реєстрації Telegram-бота зображений на рисунку 2.1.

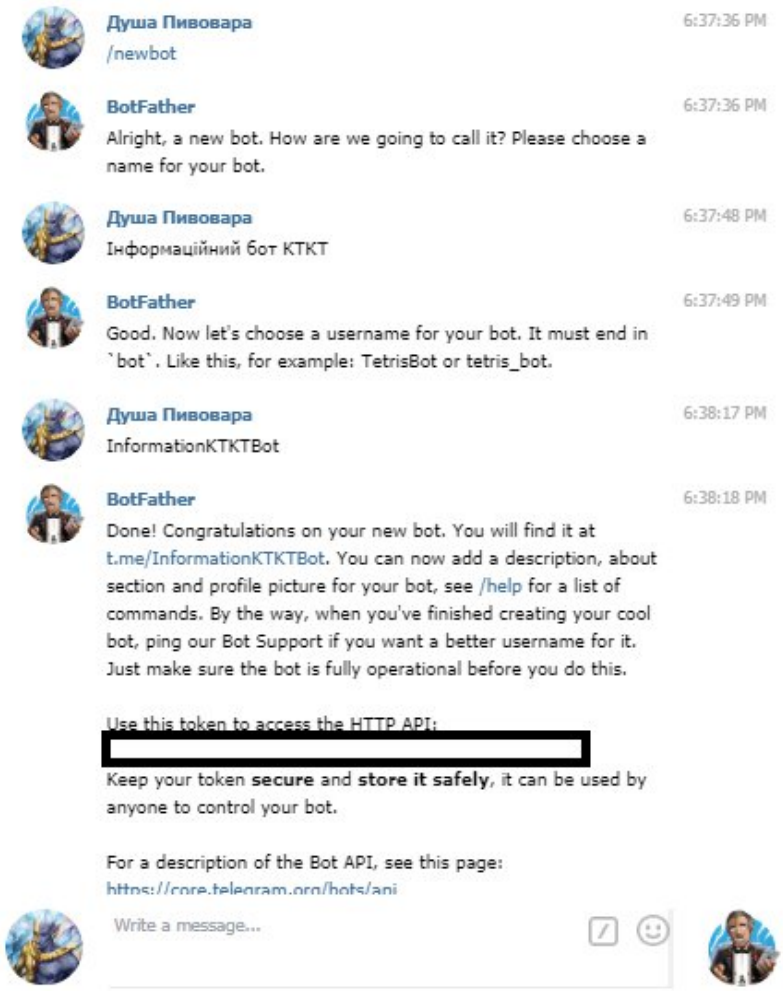


Рисунок 2.1 – Приклад реєстрації Telegram-бота

Щоб розпочати реєстрацію, необхідно ввести команду «/newbot», після чого ввести довільну після чого ввести довільну назву (name) для свого бота. Ця назва буде лише декоративною і виводитись для користувачів. Після введення назви необхідно ввести його ім’я (username). Ім’я чат-бота обов’язково мусить

закінчуватись на «Bot» або «_bot», це ім’я буде використовуватись у посиланні на бота. Після завдання назви та імені, «BotFather» видає токен. Токен – це унікальний набір символів для доступу до TelegramBot API.

Тепер можна встановити додаткові параметри, хоча вони не є обов’язкоми.Для налаштування додаткових параметрів існують спеціальні команди, описані у таблиці 2.1.

Таблиця 2.1. – Перечислення та опис команд BotFather

Команда	Опис команди
/newbot	Створити нового бота
/mybots	Переглянути список ботів
/setname	Змінити ім’я
/setdescription	Задати чи змінити опис бота. Опис бота буде виводитись при першому його відкритті, після фрази “Whatcanthisbotdo?” (Що може робити цей бот?)
/setabouttext	Задає чи змінює текст у полі “Про бота:”
/setuserpic	Задає картинку - “аватар” для бота.
/setcommands	Задає підсказку зі списком команд і їх призначенням для користувача
/deletebot	Видаляє бот
/setinline	Вмикає чи вимикає Inline-режим для бота. У Inline-режимі бота можна буде викликати із будь-якого чату. При його згадуванні до бота прийде спеціальний API-запит.
/setinlinefeedback	Дозволяє отримувати інформацію про кількість обраних користувачами команд
/setinlinegeo	Дозволяє або забороняє у Inline-бота можливість отримувати геолокацію користувачів
/setprivacy	Вмикає або вимикає режим, у якому бот спілкується зі всіма користувачами групового чату лише приватно
/setjoingroup	Дозволяє або забороняє боту бути приєднаним то групового чату
/token	Отримати токен бота
/revoke	Видалити токен бота

2.2. Проектування моделі взаємодії клієнта і серверу

Для початку необхідно спроектувати модель взаємодії чат-бота з телеграм-сервером. Існує два способи отримання повідомлень від Telegram: LongPolling та WebHooks. На рисунку 2.2. зображено схему взаємодії чат-боту, серверу Telegram і клієнта за допомогою LongPolling.

Суть LongPolling полягає в тому, що чат-бот, запущений на ПК, сам звертається до серверу Telegram за допомогою API методу GetUpdates і просить в нього щоб він надіслав чат-боту нові повідомлення від клієнтів, якщо вони з'являться протягом певного короткого часу (Timeout).

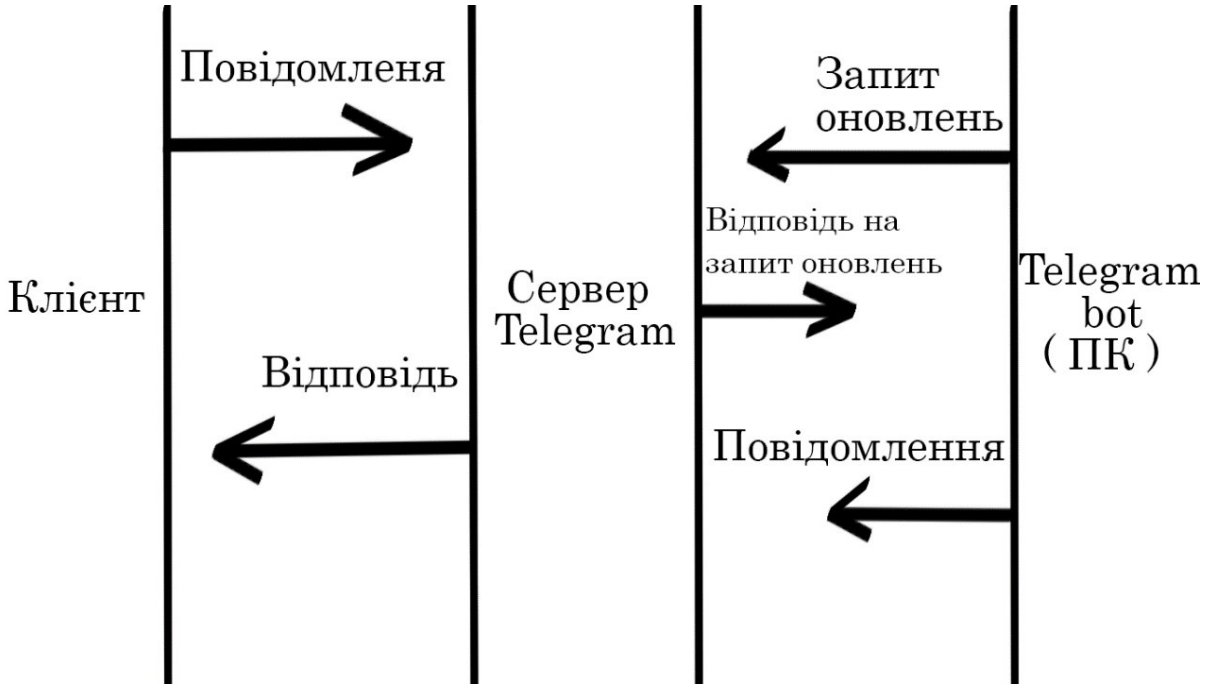


Рисунок 2.2 – Схема взаємодії чат-боту, серверу Telegram і клієнта з використаннямLongPolling

Якщо з'явилось повідомлення, чат-бот його обробляє і надсилає відповідь до серверу Telegram, який в свою чергу надсилає її до клієнта. Недоліком цього варіанту є те, що бот буде функціонувати лише тоді, коли на ПК запущений консольний застосунок цього бота.

На рисунку 2.3 зображено схему взаємодії чат-боту, серверу Telegram і клієнта за допомогою методу SetWebHook().

Суть моделі з використанням вебхуків полягає в тому, що чат-бот поміщається на домен. Сервісу Telegram повідомляється адреса цього домену, і він створює для себе строгу відповідність бота і хостингу.

Коли клієнт надсилає боту повідомлення, сервіс Telegram самостійно звертається до бота на хостингу, щоб той обробив повідомлення. Бот обробляє повідомлення і надсилає відповідь до сервісу Telegram, який в свою чергу направляє цю відповідь до клієнта.

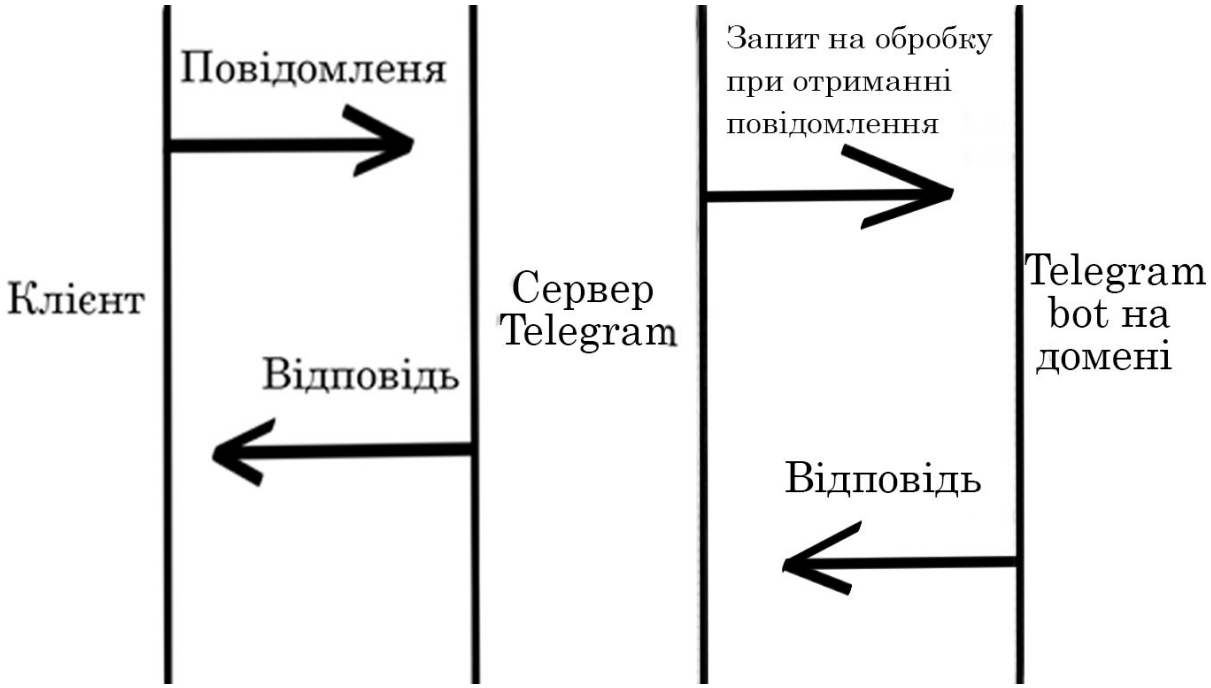


Рисунок 2.3 - Схема взаємодії чат-боту, серверу Telegram і клієнта з використанням WebHooks

Модель взаємодії чат-боту і серверу Telegram за допомогою вебхуків є надійнішою, проте у цій роботі використана модель взаємодії з використанням LongPolling, оскільки бюджет не дозволяє розмістити чат-бота на домені.

Проте, якби ця можливість з’явилась, це можна буде швидко реалізувати, змінивши кілька стрічок коду і повідомивши BotFather адрес домену після розміщення бота на домені.

2.3 Написання програми

Для початку написання програми необхідно було створити проект як .Net консольний застосунок, після чого встановити NuGet-пакет під назвою Telegram.bot. У цій роботі використано найновішу на даний момент версію цієї бібліотеки (15.5.1).

Telegram.bot – це бібліотека, яка є оболонкою над запитами до TelegramBot API. Використання цієї бібліотеки дозволяє у певній мірі спростити код програми, оскільки не потрібно з нуля створювати безліч классів і функцій із прямим зверненням до TelegramBotApi.

Після встановлення цього пакету, його було підключено до Program.cs
На рисунку 2.4 зображено список підключених бібліотек

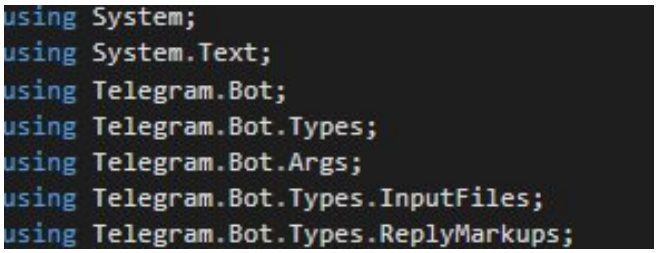


Рисунок 2.4 – Список підключених бібліотек

Варто наголосити, що директива using – не те саме, що директива include в інших С-подібних мовах, і підключених бібліотек насправді не сім, а дві. Ця директива просто розгортає простір імен, даючи можливість звертатися до функцій і методів без префікса імені простору.

Коли бібліотеки підключені, можна приступати до оголошення змінних контексту класу, які в ООП є аналогом глобальних змінних у імперативному програмуванні. Зазвичай такі змінні не є бажаними в ООП, та у цьому випадку вони допустимі. На рисунку 2.5 зображено фрагмент коду з оголошенням змінних контексту класу.

```
class Program
{
    private static ITelegramBotClient botClient;
    private static int t=0;
    private static int d;
    const string V = "Пара відсутня";
    //Викладачі
    private static string[][,] teachers = new string[][,]...;
    //Групи
    private static string[][,] groups = new string[][,]...;
    //Студенти
    private static string[][] students = new string[][]...;
```

Рисунок 2.5 – Оголошення змінних контексту класу

botClient – це екземпляр об’єкту, що реалізовує інтерфейс ITelegramBotClient. Інтерфейс ITelegramBotClient є інтерфейсом над запитами до TelegramBotApi, вньому описано безліч різних подій (наприклад, отримання повідомлення чи натискання Inline-кнопки), задач (наприклад, відправлення повідомлення з текстом чи медіа, дії над учасниками чату чи запити на отримання оновлень) та змінних (наприклад, час очікування для LongPolling чи унікальний ідентифікатор бота). Цей інтерфейс описаний у просторі імен Telegram.Bot, його опис продемонстровано на рисунку 2.6.

```
public interface ITelegramBotClient
{
    int MessageOffset { get; set; }
    bool IsReceiving { get; }
    TimeSpan Timeout { get; set; }
    int BotId { get; }

    event EventHandler<ReceiveGeneralErrorEventArgs> OnReceiveGeneralError;
    event EventHandler<CallbackQueryEventArgs> OnCallbackQuery;
    event EventHandler<ChosenInlineResultEventArgs> OnInlineResultChosen;
    event EventHandler<InlineQueryEventArgs> OnInlineQuery;
    event EventHandler<MessageEventArgs> OnMessageEdited;
    event EventHandler<MessageEventArgs> OnMessage;
    event EventHandler<ReceiveErrorEventArgs> OnReceiveError;
    event EventHandler<UpdateEventArgs> OnUpdate;
    event EventHandler<ApiResponseEventArgs> ApiResponseReceived;
    event EventHandler<ApiRequestEventArgs> MakingApiRequest;

    Task AddAnimatedStickerToSetAsync(int userId, string name, InputFileStream tgsSticker, string emojis, MaskPosition maskPosition
    Task AddStickerToSetAsync(int userId, string name, InputOnlineFile pngSticker, string emojis, MaskPosition maskPosition = null,
    Task AnswerCallbackQueryAsync(string callbackQueryId, string text = null, bool showAlert = false, string url = null, int cacheTi
    Task AnswerInlineQueryAsync(string inlineQueryId, IEnumerable<InlineQueryResultBase> results, int? cacheTime = null, bool isPers
    Task AnswerPreCheckoutQueryAsync(string preCheckoutQueryId, string errorMessage, CancellationToken cancellationToken = default);
    Task AnswerPreCheckoutQueryAsync(string preCheckoutQueryId, CancellationToken cancellationToken = default);
    Task AnswerShippingQueryAsync(string shippingQueryId, string errorMessage, CancellationToken cancellationToken = default);
    Task AnswerShippingQueryAsync(string shippingQueryId, IEnumerable<ShippingOption> shippingOptions, CancellationToken cancellatio
    Task CreateNewAnimatedStickerSetAsync(int userId, string name, string title, InputFileStream tgsSticker, string emojis, bool isM
```

Рисунок 2.6 – Інтерфейс ITelegramBotClient

Змінна контексту класу t – це прапорець, що вказує на активну функцію. Вона необхідна, наприклад, для того, щоб коли користувач буде вводити прізвище викладача після натискання кнопки “Знайти викладача”, бот шукав це прізвище саме серед викладачів, а не серед студентів, і відповідно виконував необхідну функцію. Ця змінна буде отримувати певне значення при натисканні функціональних кнопок, це значення буде перевірятись перед виконанням функції і скидатись назад до значення за замовчанням після її виконання.

Змінна контексту класу d буде отримувати своє значення в залежності від поточного дня тижня, це необхідно для реалізації більшості функцій. Цю змінну було створено тому, що ці функції для виводу інформації будуть використовувати масиви, один з індексів яких відповідає дню тижня. В цьому випадку пряме використання методу DayOfWeek неможливе, і було необхідно створити саме змінну типу int для ідентифікації дня тижня.

Після цих змінних оголошується три масиви. На рисунку 2.6 їх оголошення є згорнутим:

-Teachers - викладачі

-Groups - групи

-Students – студенти

Для початку, потрібно обґрунтувати необхідність введення цих масивів як глобальних змінних. Даний випадок – це дуже рідкісне виключення, коли такі дії є доцільними в ООП. Це виключення пов’язано з тим, що при створенні telegram-бота ми маємо лише два метода, вкладених у клас program: метод Main та метод BotOnMessage.

Метод BotOnMessage підписаний на подію botClient.OnMessage, іншими словами, цей метод виконується кожного разу, коли виникає подія botClient.OnMessage.

Через певні особливості мови програмування C# створювати змінні t і d всередині методу BotOnMessage не має сенсу, оскільки функції не могли би з ними працювати. В іншому випадку ці змінні можна було би вписати в параметри

методу BotOnMessage, та в даному випадку це неможливо, оскільки параметри методу BotOnMessage є суворо регламентованими.

З вище перерахованими масивами ситуація інша: їх можна спокійно оголосити у тілі методу BotOnMessage, проте, так як цей метод викликається кожного разу при отриманні повідомлення, то ці масиви також ініціалізувались би кожного разу при отриманні повідомлення. Отже, оголошення цих масивів у тілі методу BotOnMessage не є доцільним, оскільки це би створювало велике навантаження.

Другим варіантом є оголошення масивів всередині функцій FindTeacher(), FindGroup, FindStudent() та DetermineDay(). Це створило б менше навантаження, ніж у першому варіанті, оскільки при отриманні нового повідомлення оголошувались би лише необхідні для обробки цього повідомлення масиви.

Не зважаючи на все, як не дивно, найбільш доцільним варіантом у даній ситуації є ініціалізація масивів як глобальних змінних, тобто в контексті класу. У такому випадку масиви оголошуються всього один раз, а звертання до них можливе у будь-якій частині програми.

На початку проектування для зберігання даних про викладачів, студентів і групи планувалось використовувати базу даних, та пізніше було складено такий спосіб зберігання цих даних, при якому їх автоматизована вибірка буде найбільш раціональною, а структура максимально інтуїтивно зрозумілою, щоб будь-яка людина при бажанні змогла відредагувати ці дані. В цьому способі дані зберігаються в масивах масивів.

Масив масивів - це масив, елементи якого самі є масивами. Елементи масиву масивів можуть мати різні розміри та вимірність. Масиви масивів також називають нерегулярними масивами. Масиви масивів варто відрізняти від паралельних масивів. Паралельні масиви вважаються ознакою погано написаного коду і піддаються критиці зі всіх сторін, і не безпідставно, оскільки паралельні масиви насправді не пов’язані між собою, займають багато місця і знаходяться у різних комірках пам’яті.

На рисунку 2.7 зображено фрагмент масиву teachers

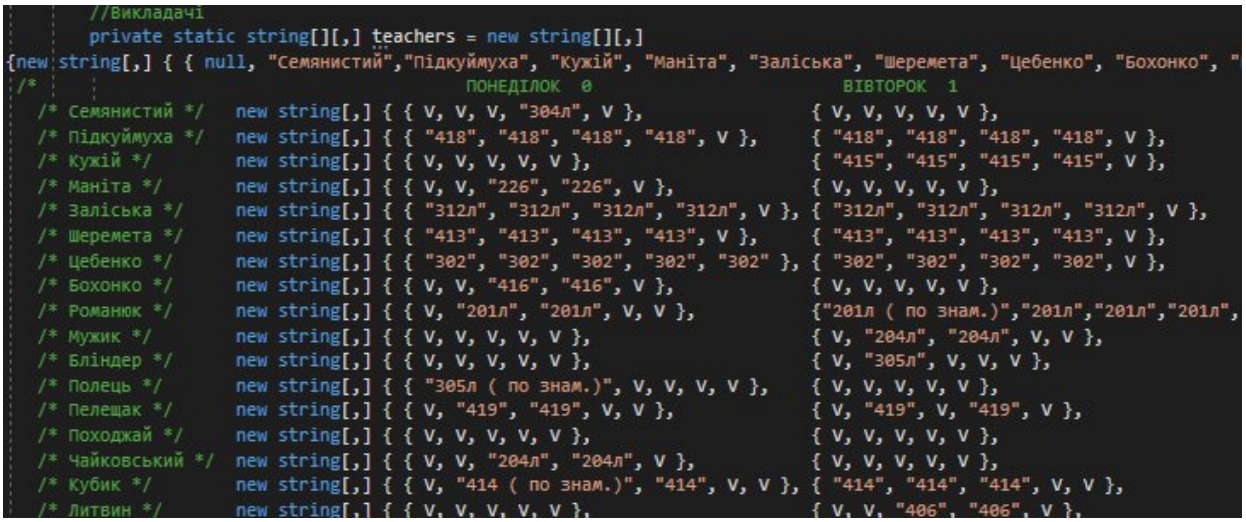


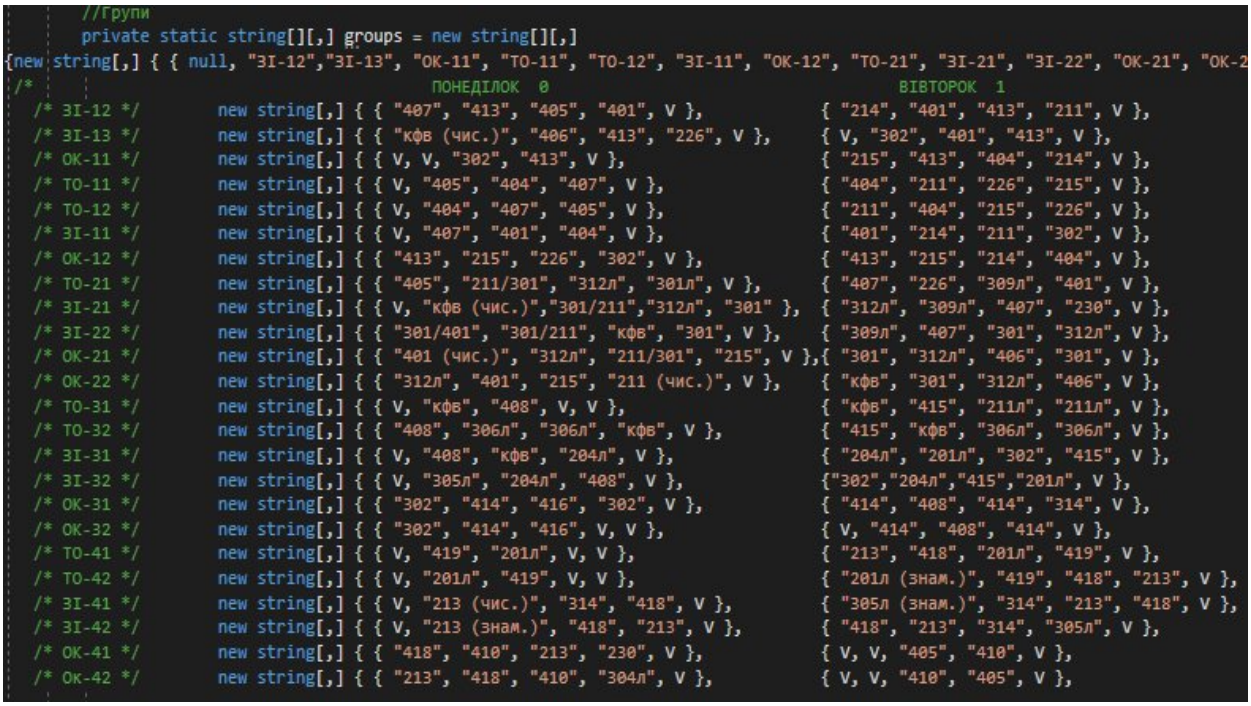
Рисунок 2.7 – Фрагмент масиву teachers

Кожен підмасив в цьому масиві масивів є двовимірним, на це вказують квадратні дужки «[][,]». Перший підмасив також насправді є двовимірним, але з одним рядком. Він містить прізвища викладачів, його перший елемент – null, це необхідно для правильної індексації у майбутньому, так як вибірка буде проводитись за допомогою циклу з лічильником.

Кожен підмасив teachers[i] вказує на викладача, прізвище якого вказано у елементі teachers[0][0,i]. Рядки вказують на день тижня, а стовпці на пару. Наприклад, елемент teachers[1][0,0] вказує на першу пару в понеділок викладача за прізвищем teachers[0][0,1], тобто за прізвищем Семянистий, оскільки індексація розпочинається з нуля. Також варто згадати за нововведену константу V, яка вказує на відсутність пари.

На рисунку 2.8 зображений фрагмент масиву groups. Його структура масиву майже повністю аналогічна масиву teachers.

Різницю структури масиву groups від масиву teachers складає лише той факт, що в першому підмасиві масиву groups описуються не викладачі, а групи.

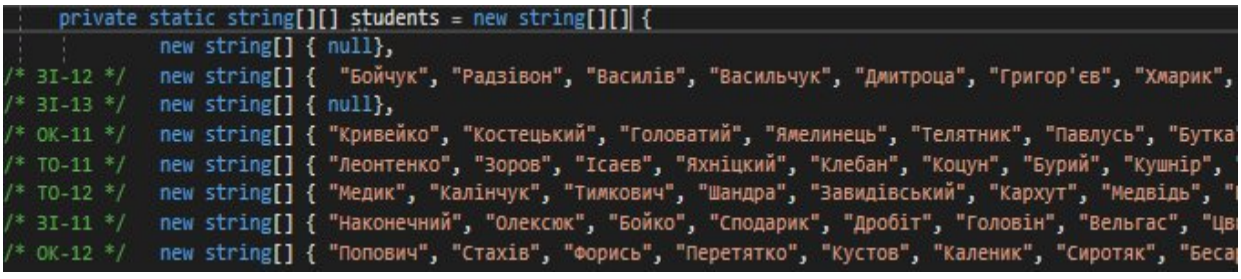


```
        //Групи
        private static string[][,] groups = new string[][,]
{new string[,] { { null, "ЗІ-12","ЗІ-13", "ОК-11", "ТО-11", "ТО-12", "ЗІ-11", "ОК-12", "ТО-21", "ЗІ-21", "ЗІ-22", "ОК-21", "ОК-2
/*                                      ПОНЕДІЛОК  0                                                    ВІВТОРОК  1
  /* ЗІ-12 */       new string[,] { { "407", "413", "405", "401", V },          { "214", "401", "413", "211", V },
  /* ЗІ-13 */       new string[,] { { "кфв (чис.)", "406", "413", "226", V },    { V, "302", "401", "413", V },
  /* ОК-11 */       new string[,] { { V, V, "302", "413", V },                   { "215", "413", "404", "214", V },
  /* ТО-11 */       new string[,] { { V, "405", "404", "407", V },               { "404", "211", "226", "215", V },
  /* ТО-12 */       new string[,] { { V, "404", "407", "405", V },               { "211", "404", "215", "226", V },
  /* ЗІ-11 */       new string[,] { { V, "407", "401", "404", V },               { "401", "214", "211", "302", V },
  /* ОК-12 */       new string[,] { { "413", "215", "226", "302", V },           { "413", "215", "214", "404", V },
  /* ТО-21 */       new string[,] { { "405", "211/301", "312л", "301л", V },     { "407", "226", "309л", "401", V },
  /* ЗІ-21 */       new string[,] { { V, "кфв (чис.)","301/211","312л", "301" }, { "312л", "309л", "407", "230", V },
  /* ЗІ-22 */       new string[,] { { "301/401", "301/211", "кфв", "301", V },   { "309л", "407", "301", "312л", V },
  /* ОК-21 */       new string[,] { { "401 (чис.)", "312л", "211/301", "215", V },{ "301", "312л", "406", "301", V },
  /* ОК-22 */       new string[,] { { "312л", "401", "215", "211 (чис.)", V },   { "кфв", "301", "312л", "406", V },
  /* ТО-31 */       new string[,] { { V, "кфв", "408", V, V },                   { "кфв", "415", "211л", "211л", V },
  /* ТО-32 */       new string[,] { { "408", "306л", "306л", "кфв", V },         { "415", "кфв", "306л", "306л", V },
  /* ЗІ-31 */       new string[,] { { V, "408", "кфв", "204л", V },              { "204л", "201л", "302", "415", V },
  /* ЗІ-32 */       new string[,] { { V, "305л", "204л", "408", V },             {"302","204л","415","201л", V },
  /* ОК-31 */       new string[,] { { "302", "414", "416", "302", V },           { "414", "408", "414", "314", V },
  /* ОК-32 */       new string[,] { { "302", "414", "416", V, V },               { V, "414", "408", "414", V },
  /* ТО-41 */       new string[,] { { V, "419", "201л", V, V },                  { "213", "418", "201л", "419", V },
  /* ТО-42 */       new string[,] { { V, "201л", "419", V, V },                  { "201л (знам.)", "419", "418", "213", V },
  /* ЗІ-41 */       new string[,] { { V, "213 (чис.)", "314", "418", V },        { "305л (знам.)", "314", "213", "418", V },
  /* ЗІ-42 */       new string[,] { { V, "213 (знам.)", "418", "213", V },       { "418", "213", "314", "305л", V },
  /* ОК-41 */       new string[,] { { "418", "410", "213", "230", V },           { V, V, "405", "410", V },
  /* Ок-42 */       new string[,] { { "213", "418", "410", "304л", V },          { V, V, "410", "405", V },
```

Рисунок 2.8 – Фрагмент масиву groups

Його підмасиви також є двовимірними, індексація також аналогічна: елемент groups[1][0,0] вказує на першу пару в понеділок групи groups[0][0,1], тобто ЗІ-12.

Від цих двох масивів досить сильно відрізняється за структурою масив students. Його фрагмент зображено на рисунку 2.9



```
    private static string[][] students = new string[][] {
            new string[] { null},
/* ЗІ-12 */   new string[] {  "Бойчук", "Радзівон", "Василів", "Васильчук", "Дмитроца", "Григор'єв", "Хмарик",
/* ЗІ-13 */   new string[] { null},
/* ОК-11 */   new string[] { "Кривейко", "Костецький", "Головатий", "Ямелинець", "Телятник", "Павлусь", "Бутка
/* ТО-11 */   new string[] { "Леонтенко", "Зоров", "Ісаєв", "Яхніцкий", "Клебан", "Коцун", "Бурий", "Кушнір",
/* ТО-12 */   new string[] { "Медик", "Калінчук", "Тимкович", "Шандра", "Завидівський", "Кархут", "Медвідь",
/* ЗІ-11 */   new string[] { "Наконечний", "Олексюк", "Бойко", "Сподарик", "Дробіт", "Головін", "Вельгас",
/* ОК-12 */   new string[] { "Попович", "Стахів", "Форись", "Перетятко", "Кустов", "Каленик", "Сиротяк",
```

Рисунок 2.9 – Фрагмент масиву students

Перший підмасив цього масиву пустий по тій причині, що немає сенсу його заповнювати, оскільки в такому випадку він у повній мірі повторював би перший підмасив масиву groups. Повторювати його сенсу немає, так як у місцях, де

необхідно звернутись до першого підмасиву масиву students ми можемо просто звертатись до першого підмасиву масиву groups.

Структура цього масиву досить проста: всі його підмасиви є одновимірними, в них описуються прізвища студентів групи, зазначеної в коментарях. Кожен підмасивstudents[i] описує прізвища студентів групи groups[0][0,i]. При перебиранні цього масиву циклом for необхідно мати на увазі, що довжина кожного підмасиву відрізняється, і її необхідно отримувати з властивості students[i].length.

На жаль, списки студентів деяких груп у цьому масиві відсутні, оскільки дані для його заповнення брались із загальнодоступних джерел

Після завдання глобальних змінних, наступним кроком було написання методу Main. Цей фрагмент коду зображений на рисунку 2.10.

```
Ссылок: 0
static void Main(string[] args)
{
    botClient = new TelegramBotClient("[redacted]") { Timeout = TimeSpan.FromSeconds(10) };
    var me = botClient.GetMeAsync().Result;
    Console.WriteLine($"Bot id {me.Id}. Bot Name {me.FirstName}");

    botClient.OnMessage += Bot_OnMessage;
    botClient.StartReceiving();
    Console.ReadKey();

}
```

Рисунок 2.10 – Метод Main

У методі Main об’єкту botClient присвоюється посилання на об’єкт TelegramBotClient, що знаходиться у однойменному класі TelegramBotClient, який наслідує інтерфейс ITelegramBotClient.

```
public class TelegramBotClient : ITelegramBotClient
{
    public TelegramBotClient(string token, HttpClient httpClient = null);
    public TelegramBotClient(string token, IWebProxy webProxy);
```

Рисунок 2.11 – Об’єкт TelegramBotClient

На рисунку 2.11 зображено фрагмент коду, у якому оголошується об’єкт TelegramBotClient. На цьому рисунку видно, що даний об’єкт може мати два параметра. Перший параметр – це токен бота, він є обов’язковим. Другий параметр – це проксі сервер. Задання цього параметру не є обов’язковим, проте він потрібен коли бот розміщується у країні, для якої доступ у телеграм заблокований. Прикладом таких країн є Росія, Китай, Пакістан та Оман. Необхідно про це пам’ятати, якщо розміщувати бота на серверах цих країн.

Після завдання токену було задано таймаут. Таймаут – це час, протягом якого після запиту GetUpdates() телеграм пришле оновлення при наявності нових повідомлення. Якщо нові повідомлення були ще до надсилання запиту GetUpdates(), то телеграм надасть відповідь моментально, після чого послідує наступний запит GetUpdates. Технічно, таймаут не є обов’язковим, проте якщо його не вказати то бот буде постійно з надзвичайно малим інтервалом надсилати запити до TelegramBotApi, це спричинить до навантаження на комп’ютер та навантаження на сервер телеграму, яке може автоматично розцінитись як DDoS-атака і IP-адреса, з якої надходять ці запити, може бути заблокована.

Задача GetMeAsync() має властивості, які дозволяють отримати самі різні дані про бота чи іншого користувача, такі як: унікальний ідентифікатор, ім’я, прізвище, інформація чи є цей користувач ботом, інформація чи може бот читати всі повідомлення в груповому чаті і чи може він взагалі приєднуватись до цих чатів, інформація про обрану користувачем мову, тощо. Вся ця інформація, звичайно ж, береться з серверів телеграму. Це чудова можливість для тестування з’єднання, що і було реалізовано.

Новооголошена змінна me – це короткий шаблон для отримання інформації від бота. Якщо підключення справне, то в консоль відбудеться вивід значень me.Id (унікальний ідентифікатор бота) та me.FirstName (ім’я бота). Результати тестування зображено у розділі 3.

Стрічка «botClient.OnMessage += Bot_OnMessage» – це “підписування” методу Bot_OnMessage на подію botClient.OnMessage. Іншими словами, при виникненні події botClient.OnMessage виконується тіло методу Bot_OnMessage.

Подія botClient.OnMessage виникає тоді, коли з’являється інформація про нове повідомлення для бота від користувача. Ця подія описана у просторі імен Telegram.Bot.Args (рис. 2.12).

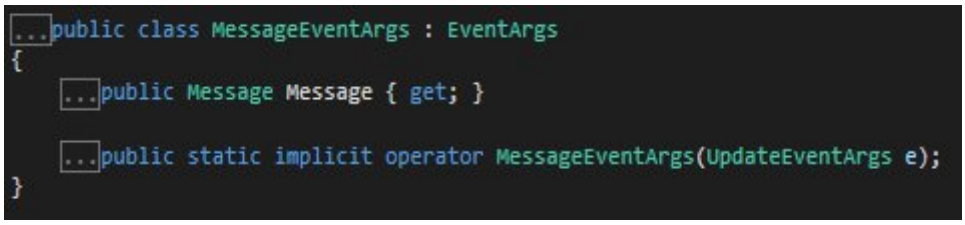


Рисунок 2.12 – Опис події OnMessage

При виклику методу botClient.StartReceiving бот-клієнт розпочинає отримувати повідомлення, використовуючи LongPolling, тобто API-метод GetUpdates().

Завершує метод Main команда Console.ReadKey(). Ця команда необхідна для того, щоб консоль не закривалась одразу після виводу інформації про ID та ім’я бота, а очікувала натиснення клавіші.

Основна частина програми – це метод BotOnMessage. Цей метод є підписаним на подію botClient.OnMessage, тобто він виконується кожного разу, коли бот отримує оновлення про нове повідомлення.

Метод BotOnMessage був оголошений автоматично, за допомогою підказки IntelliSense від інтегрованого середовища розробки Microsoft VisualStudioCommunity. Єдиною зміною після стандартного оголошення було додавання ключового слова async, що вказує на асинхронність методу, це дозволяє винести завдання обробки повідомлень з основного потоку програми, тобто виконувати їх одночасно з іншими частинами програми.

Фрагмент коду з оголошенням методу Bot_OnMessage зображено на рисунку 2.13

Рисунок 2.13 – Фрагмент коду з оголошенням методу Bot_OnMessage

Параметри методу Bot_OnMessage:

Objectsender – це користувач, який надіслав повідомлення.

MessageEventArgs e – це саме повідомлення.

Ці параметри також були створені автоматично за допомогою IntelliSense і в жодному разі не підлягають зміні.

Після оголошення методу Bot_OnMessage було оголошено змінну text. Змінна text – це текст повідомлення, надісланого користувачем боту. Варто розуміти, що текст і повідомлення – це не одне і те саме, оскільки повідомлення може також містити:

-Зображення

-Аудіофайли

-Голосове повідомлення

-Відеоповідомлення

-Відеофайли

-Стікери

-Геотег

-Контакт якоїсь особи

Ми відокремлюємо текст повідомлення від всіх вище перелічених речей, а якщо текст відсутній – бот не буде реагувати на повідомлення. Також, при відправленні повідомлення, воно буде приходити до нас у консоль з вказанням унікального ідентифікатора чату. Це потрібно для тестування з’єднання між ботом та сервером Telegram.

Наступним кроком було оголошення наступних функцій:

-Функція визначення дня тижня DetermineDay()

-Функція пошуку викладача FindTeacher()

-Функція пошуку групиFindGroup()

-Функція пошуку студентаFindStudent()

-Функція визначення парного/непарного тижняDetermineWeek()

Функція визначення дня тижня DetermineDay() – це проста функція, призначена для задання значення змінній для змінної d.

Функція DetermineDay() виконується не одноразово, а викликається при виконанні функції з пошуку викладача FindTeacher(), функції з пошуку студента FindStudent() та функції з пошуку групи FindGroup(), а також при зчитуванні функціональних кнопок, щоб заблокувати доступ до цих функцій у суботу та неділю. Це зумовлено тим, що день тижня постійно змінюється, тому значення для d не можна задати лише один раз. Ця функція зображена
на рисунку 2.14.

```
static int DetermineDay()
{
    DayOfWeek day = DateTime.Now.DayOfWeek;
    switch (day)
    {
        case DayOfWeek.Monday:
            d = 0;
            break;
        case DayOfWeek.Tuesday:
            d = 1;
            break;
        case DayOfWeek.Wednesday:
            d = 2;
            break;
        case DayOfWeek.Thursday:
            d = 3;
            break;
        case DayOfWeek.Friday:
            d = 4;
            break;
        case DayOfWeek.Saturday:
            d = 5;
            break;
        case DayOfWeek.Sunday:
            d = 6;
            break;

    }
    return d;
}
```

Рисунок 2.14 – Функція DetermineDay()

Відлік для змінної d розпочинається з нуля, оскільки ця змінна потрібна для індексації елементів масивів teachers та groups, перший рядок у підмасивах цих масивів за індексом 0 відповідає понеділку, за індексом 1 – вівторку і т.д.

Наступна функція – функція пошуку викладача FindTeacher(). Якщо текст введеного користувачем повідомлення збігається з прізвищем викладача teachers[0][0, i], то відбувається вивід елементу виду teachers[i][d,x]. Фрагменти цієї функції зображено на рисунках 2.15 та 2.16.

```
async void FindTeacher()
{
    DetermineDay();
    for (int i = 1; i < teachers[0].Length; i++)
    {

        if (text == teachers[0][0, i])
        {

            if (DateTime.Now.TimeOfDay < new TimeSpan(8, 30, 00))...
```

Рисунок 2.15 – Фрагмент функції FindTeacher()

Лічильник «i» вказує на викладача, змінна d, отримана після виконання функції DetermineDay() – на день. X – не змінна, цей індекс задається вручну, він вказує на поточну пару, а визначається за допомогою порівняння виду «кінець минулої пари < поточний час > кінець пари». Якщо перша пара ще не почалась або остання пара вже закінчилась, то користувачу повідомляється про це.

```
    else if ((DateTime.Now.TimeOfDay > new TimeSpan(14, 05, 00)) && (DateTime.Now.TimeOfDay < new TimeSpan(15, 40, 00)))
    {
                await botClient.SendTextMessageAsync(
                chatId: e.Message.Chat,
                text: $"{teachers[i][d, 3]}"
                ).ConfigureAwait(false);
    }

    else if ((DateTime.Now.TimeOfDay > new TimeSpan(15, 40, 00)) && (DateTime.Now.TimeOfDay < new TimeSpan(17, 25, 00)))
    {
                await botClient.SendTextMessageAsync(
                chatId: e.Message.Chat,
                text: $"{teachers[i][d, 4]}"
                ).ConfigureAwait(false);
    }

    else if (DateTime.Now.TimeOfDay > new TimeSpan(17, 25, 00))...
    t = 0; //Скидання прапорця
}
if ((t == 1) && (i == 18) && (text != "Знайти викладача"))
{
                await botClient.SendTextMessageAsync(
                chatId: e.Message.Chat,
                text: $"На жаль, я не знаю такого викладача"
                ).ConfigureAwait(false);
                t = 0;
}
```

исунок 2.16 – Фрагмент функції FindTeacher()

Якщо текст написаного користувачем повідомлення не співпадає із жодним із елементів teacher[0][0,i], користувачу також про це повідомляється. У цій умові існує виключення text != “Знайти викладача”, воно потрібно по тій причині, що при натисканні функціональної кнопки присвоюється значення для t, і тому текст кнопки підходить під умови для початку функції.

Наступна функція – функція пошуку групи FindGroup(). Вона досить схожа на функцію пошуку викладача FindTeacher(). Якщо текст введеного користувачем повідомлення збігається з назвою групи groups[0][0, i], то відбувається вивід елементу виду groups[i][d,x]. Фрагмент цієї функції зображено на рисунку 2.17.

Лічильник «i» вказує на групу, змінна d, отримана після виконання функції DetermineDay() – на день. Індекс X – не змінна, він вказує на поточну пару, а визначається за допомогою порівняння виду «кінець минулої пари < поточний час > кінець пари».

Якщо перша пара ще не почалась або остання пара вже закінчилась, то користувачу повідомляється про це.

```
async void FindGroup()
{
    DetermineDay();
    for (int i = 1; i < groups[0].Length; i++)
    {

        if (text == groups[0][0, i])
        {
            if (DateTime.Now.TimeOfDay < new TimeSpan(8, 30, 00))
            {
                    await botClient.SendTextMessageAsync(
                    chatId: e.Message.Chat,
                    text: $"По моїм даним, перша пара ще не наступила"
                    ).ConfigureAwait(false);
            }
            else if ((DateTime.Now.TimeOfDay > new TimeSpan(8, 30, 00)) && (DateTime.Now.TimeOfDay < new TimeSpan(10, 05, 00)))
            {
                        await botClient.SendTextMessageAsync(
                        chatId: e.Message.Chat,
                        text: $"{groups[i][d, 0]}"
                        ).ConfigureAwait(false);
            }
```

Рисунок 2.17 – Фрагмент функції FindTeacher()

Якщо текст написаного користувачем повідомлення не співпадає із жодним із елементів groups[0][0,i], користувачу також про це повідомляється. У цій умові існує виключення text != “Знайти групу”, воно потрібно по тій причині, що при натисканні функціональної кнопки присвоюється значення для t, і тому текст кнопки також підходить під умови для запуску функції.

Наступна функція – функція пошуку студента FindStudent(). Вона досить відрізняється від попередніх двох функцій, в першу чергу тим, що у ній використовується подвійний цикл for. Фрагмент цієї функції зображено на рисунку 2.18.

```
async void FindStudent()
{
    DetermineDay();
    for (int i = 1; i < (groups[0].Length - 1); i++)
    {
        for (int j = 0; j < students[i].Length; j++)
        {
            if (text == students[i][j])
            {   if (DateTime.Now.TimeOfDay < new TimeSpan(8, 30, 00))...

                else if ((DateTime.Now.TimeOfDay > new TimeSpan(8, 30, 00)) && (DateTime.Now.TimeOfDay < new TimeSpan(10, 05, 00)))
                {
                    if (groups[i][d, 0] == V)
                    {
                            await botClient.SendTextMessageAsync(
                            chatId: e.Message.Chat,
                            text: $"В даний момент у студента групи {groups[0][0, i]} за прізвищем {text} пара відсутня"
                            ).ConfigureAwait(false);
                    }

                    else await botClient.SendTextMessageAsync(
                            chatId: e.Message.Chat,
                            text: $"Студент групи {groups[0][0, i]} за прізвищем {text} повинен знаходитись у аудиторії №{groups[i][d, 0]}"
                            ).ConfigureAwait(false);
                }
```

Рисунок 2.18 – Фрагмент функції FindStudent()

Подвійний цикл for в цій функції використовується по тій причині, що перебір даних проходить не з одного масиву, а з двох, а також тому, що прізвища студентів знаходяться не у одному підмасиві, а у всіх підмасивах масиву students.

Якщо введений користувачем текст співпадає з прізвищем студента students[i][j], відбувається перевірка поточної пари за допомогою порівняння «кінець минулої пари < поточний час > кінець пари». Коли визначено цю пару, користувачеві одразу не надсилається номер аудиторії, у якій повинен перебувати студент. Це зумовлено тим, що прізвища студентів часто повторюються, і не буде добре, якщо користувачеві просто надішлеться кілька аудиторій на вибір, у яких мусять знаходитись студенти з однаковим прізвищем.

Звичайно, можна було б організувати ідентифікацію студента не лише за прізвищем, а і за ім’ям. Проте, користувач не завжди може знати ім’я та фамілію, а об’єм даних для заповнення в такому випадку збільшився би у два рази, так що було вирішено що користувачеві просто буде надсилатись інформація про групу кожного знайденого студента, що спростить його ідентифікацію серед інших студентів із таким же прізвищем.

Коли визначено, прізвище якого студента містить текст повідомлення, і котра в даний момент пара, проходить перевірка, чи присутня ця пара в розкладі його групи, чи ні. Якщо пара відсутня, то до користувача надходить повідомлення, що в даний момент пара у студента певної групи за певним прізвищем відсутня. Якщо пара є, то надходить повідомлення, що студент за певним прівищем певної групи мусить знаходитись у певній аудиторії. При цьому, алгоритм пошуку прізвища студента серед елементів масиву students не закінчується, і таких повідомлень може бути кілька.

Якщо прізвище студента, введене користувачем, не співпадає з жодним із елементів масиву students, користувач отримує повідомлення про це. У цій умові існує виключення text != “Знайти студента”, воно потрібно по тій причині, що при натисканні функціональної кнопки присвоюється значення для t, і тому текст кнопки також підходить під умови для розпочинання функції. Наступна функція – функція видачі розкладу GiveSchedule(). З першого погляду вона здається дуже

простою, проте насправді її реалізація досить цікава і нестандартна. Код функції GiveSchedule() зображено на рисунку 2.19.

```
async void GiveSchedule()
{
    for (int i = 1; i < groups[0].Length; i++)
    {

        if (text.ToUpper() == groups[0][0, i])
        {
            {
                await botClient.SendPhotoAsync(
                chatId: e.Message.Chat,
                photo: $"https://raw.githubusercontent.com/vyavdyk/dip/master/{i}.jpg");
            }
                t = 0; //Скидання прапорця
        }
        if ((t == 4) && (i == (groups[0].Length-1)) && (text != "Розклад занять"))
        {

                await botClient.SendTextMessageAsync(
                chatId: e.Message.Chat,
                text: $"Назва групи введена некоректно. "
                ).ConfigureAwait(false);
                t = 0;
        }
    }
}
```

Рисунок 2.19 – Функція видачі розкладу GiveSchedule().

По своїй структурі, ця функція – це звичайний цикл з лічильником. Якщо текст введеного користувачем повідомлення, піднесений до верхнього регістру, співпадає з назвою групи groups[0][0,i], виводиться посилання. Насправді це посилання на зображення. Оскільки воно відправляється за допомогою методу SendPhotoAsync, то воно відправляється не як текст, а як фото.

Перед лапками, у яких поміщене посилання на фото, знаходиться ключовий символ «$». Він вказує на те, що текст у лапках, вкладений у фігурні дужки, буде вважатись змінною. Тобто, якщо i=5, то посилання вигляду $“https://raw.githubusercontent.com/vyavdyk/dip/master/{i}.jpg” буде розумітись як “https://raw.githubusercontent.com/vyavdyk/dip/master/5.jpg”.

Було перебрано і перевірено 9 різних хостингів картинок, і на жодному з них ніяким чином в посиланні на картинку не зберігалась оригінальна назва. Тоді було звернено увагу на репозиторій GitHub.

GitHub - це сервіс онлайн-хостингу репозиторіїв, що володіє всіма функціями розподіленого контролю версій і функціональністю управління вихідним кодом, зазвичай цей сервіс використовують саме за цим призначенням, проте є безліч інших варіантів його використання.

Із загальнодоступних джерел було отримано зображення розкладу занять станом на другий семестр 2020 року. Їх було обрізано на окремі 24 зображення – по одному для кожної групи, і завантажено на репозиторій GitHub. Вигляд репозиторію із зображеннями розкладу зображено на рисунку 2.20.

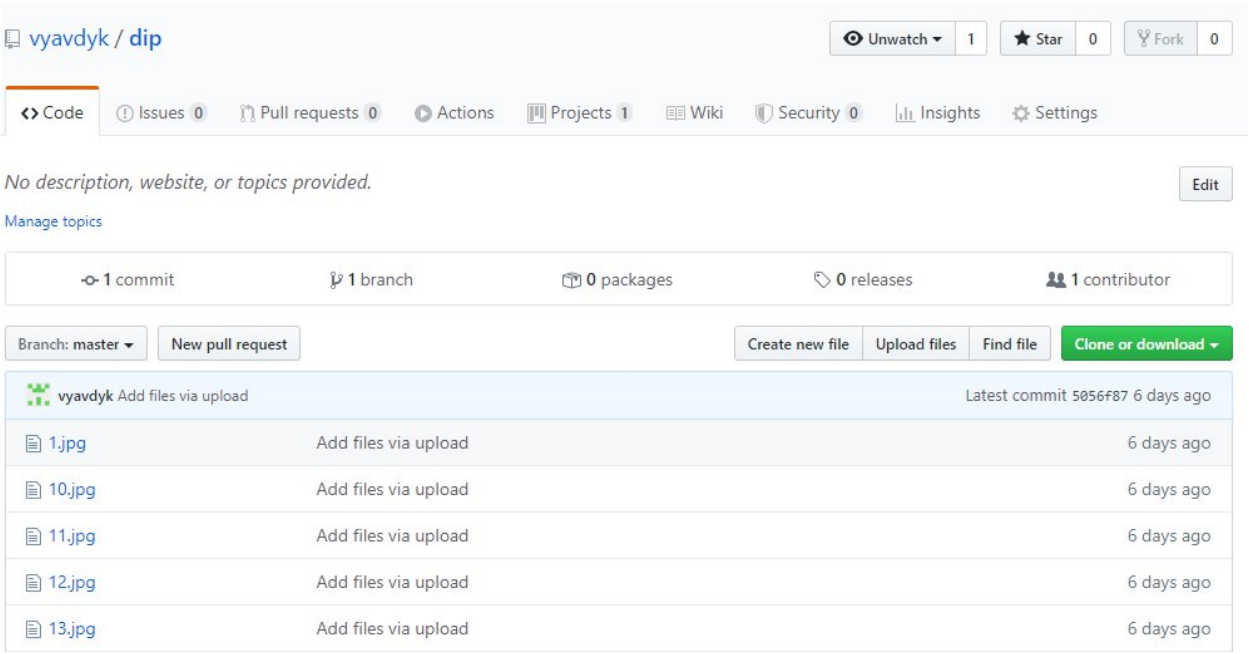


Рисунок 2.20 – Репозиторій GitHub

Цим зображенням були видані імена по такому принципу, щоб вони збігались з індексом «i» в елементах groups[0][0,i], які вказують на назви груп. Таким чином, зображення з розкладом групи ЗІ-12 має назву 1.jpg, зображення з розкладом групи ЗІ-13 має назву 2.jpg і т.д.

Якщо текст повідомлення, введеного користувачем, піднятий до верхнього регістру, не співпадає з жодним із елементівgroups[0][0,i], користувачу буде

приходити повідомлення про те, що скоріш за все назва групи була введена некоректно. У цій умові існує виключення text != “Розклад занять”, воно потрібно по тій причині, що при натисканні функціональної кнопки присвоюється значення для t, і тому текст кнопки також підходить під умови для запуску функції.Остання функція – це функція визначення парного/непарного тижня DetermineWeek().

На стадії проектування планувалось, що в цій функції вручну буде задано 25 тижнів – з них 12 парних і 13 непарних, інформація про їх чередування була взята з загальнодоступних джерел.

Проте, під час реалізації цієї функції, було помічено, що чередування навчальних тижнів є повністю передбаченим і без виключень вони чередуються кожні 7 днів, після цього було розроблено алгоритм функції, який зображений на рисунку 2.21.

Спочатку було задано стартову дату, яка є початком першого парного тижня, і дату, яка є початком першого непарного тижня. Також було задано фінішну дату, після якої навчання закінчується. Для зручності, ці дати було задано у зручному форматі «ДД.ММ» у змінних типу string, і після цього конвертовано у тип DateTime за допомогою функції ConvertToDateTime().
Всередині нескінченного циклу do… while(true) постійно задаються змінні par2 (кінець парного тижня) і unpar2 (кінець непарного тижня), після чого відбувається перевірка умов: Якщо par<= поточний тиждень > par2, відбувається вивід повідомлення про те, що поточний тиждень – парний, після чого цикл завершується. Якщо unpar<= поточний тиждень > unpar2, відбувається вивід повідомлення про те, що поточний тиждень – непарний, після чого цикл завершується.

Функція DetermineWeek() зображена на рисунку 2.21.

```
async void DetermineWeek()
{
    DateTime now = DateTime.Today;
    string startdate = "13.01";
    DateTime par = Convert.ToDateTime(startdate);
    string unpaired = "20.01";
    DateTime unpar = Convert.ToDateTime(unpaired);
    string finishdate = "28.06";
    DateTime fin = Convert.ToDateTime(finishdate);
    do
    {
        DateTime par2 = par.AddDays(7);
        if (now >= par && now < par2)
        {
            await botClient.SendTextMessageAsync(
            chatId: e.Message.Chat,
            text: $"Цей тиждень - парний ( у знаменнику )"
            ).ConfigureAwait(false);
            break;
        }
        DateTime unpar2 = unpar.AddDays(7);
        if (now >= unpar && now < unpar2)
        {
            await botClient.SendTextMessageAsync(
            chatId: e.Message.Chat,
            text: $"Цей тиждень - непарний ( у чисельнику )"
            ).ConfigureAwait(false);
            break;
        }
        par = par.AddDays(14);
        unpar = unpar.AddDays(14);
        if (par > fin)
        {
            await botClient.SendTextMessageAsync(
            chatId: e.Message.Chat,
            text: $"Схоже, що цей тиждень - не навчальний"
            ).ConfigureAwait(false);
            break;
        }

    } while (true);
}
```

Рисунок 2.21 – Функція DetermineWeek()

Якщо жодна з наведених вище умов не є правдивою, до змінних par і unpar додається по 14 днів, і цикл розпочинається заново. Якщо змінна par набула значення, більшого ніж змінна fin (дата кінця навчального тижня), користувачеві надходить повідомлення про те, що навчальний тиждень вже закінчено, після чого цикл завершується.

Тепер, коли функції оголошено, можна підходити до заключної частини програми – до самої реакції на повідомлення. Першим повідомленням в діалозі з будь-яким ботом завжди є команда /start. До першого введення команди /start

любий чат-бот є призупиненим і не може надсилати повідомлення до користувача, команда /start – це згода на діалог з ботом. При цьому користувач завжди може знову призупинити бота за допомогою спеціальних кнопок, які є вбудованими у телеграм-клієнті, і знову запустити бота за допомогою команди /start.

У нашому випадку при введенні користувачем команди /start відбувається ініціалізація функціональної клавіатури та виведення привітального повідомлення.

У офіційній документації до TelegramBot API говориться, що у Telegramicнує два типи кнопок: Reply-кнопки та Inline-кнопки. Reply-кнопки утворюють клавіатуру, що знаходиться підполем вводу повідомлення. Натискання Reply-кнопки повністю аналогічне ручному написанню її тексту. Перевірка їх натиснення також аналогічна перевірці введеного тексту. Вигляд Reply-кнопок зображено на рисунку 2.22

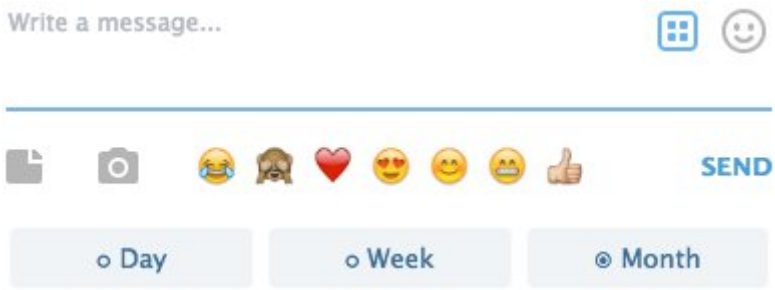


Рисунок 2.22 – Reply-кнопки

Inline-кнопки є частиною повідомлення від бота, це повідомлення може бути динамічним і змінюватись в залежності від натискання кнопок. Натискання Inline-кнопок не має нічого спільного з надсиланням повідомлення, грубо кажучи, воно викликає подію. Вигляд Inline-кнопок зображено на рисунку 2.23.

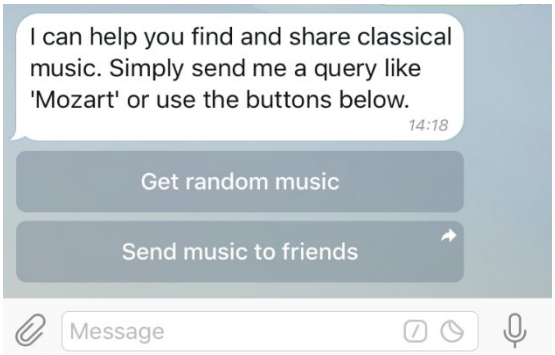


Рисунок 2.23 – Inline-кнопки

У нашому випадку більш доцільним буде використання Reply-кнопок, оскільки Inline-кнопки краще всього використовувати при складній деревоподібній структурі функцій, а в цьому проекті планується, що буде 5 чітких функцій, після вибору яких текст набагато зручніше буде вводити вручну. Код ініціалізації клавіатури та виведення привітального повідомлення зображено на рисунку 2.24.

```
if (text == "/start")
{
    var replyKeyboardMarkup = new ReplyKeyboardMarkup(
                        new KeyboardButton[][]
                        {
                new KeyboardButton[] { "Знайти викладача", "Знайти групу" },
                new KeyboardButton[] { "Знайти студента", "Розклад занять" },
                new KeyboardButton[] { "Визначення навчального тижня"},
                        },
                        resizeKeyboard: true
                    );
    await botClient.SendTextMessageAsync(
        chatId: e.Message.Chat,
        text: "Доброго дня. Оберіть необхідну функцію",
        replyMarkup: replyKeyboardMarkup
    );
```

Рисунок 2.24 – Ініціалізація клавіатури та привітального повідомлення

Параметр resizeKeyboard : true вказує на те, що розміри клавіатури будуть підлаштовуватись під кількість кнопок і їх розмір їх тексту.

Оскільки «/start» – це команда початку роботи та ініціалізації клавіатури, то після її обробки в програмі розпочинається обробка натискання кнопок. Фрагмент коду з обробки натискання кнопок зображено на рисунку 2.25.

```
if (text == "Розклад занять")
{       t = 4;
        await botClient.SendTextMessageAsync(
        chatId: e.Message.Chat,
        text: $"Введіть назву групи через дефіс, наприклад: ОК-42, ТО-31"
        ).ConfigureAwait(false);
}

else if (text == "Визначення навчального тижня")...

DetermineDay();
 if ((d == 5 || d == 6) && (text == "Знайти викладача" || text == "Знайти групу" || text == "Знайти студента"))
{
         await botClient.SendTextMessageAsync(
         chatId: e.Message.Chat,
         text: $"За моїми даними, сьогодні вихідний день. Дана функція не працює."
         ).ConfigureAwait(false);
}

   else if (text == "Знайти викладача")...

   else if (text == "Знайти групу")...

   else if (text == "Знайти студента")...

   else if (t == 0 && text !="Визначення навчального тижня" && (text == e?.Message?.Text))
    {
            await botClient.SendTextMessageAsync(
            chatId: e.Message.Chat,
            text: $"На жаль, я вас не розумію. Будь ласка, оберіть функцію."
            ).ConfigureAwait(false);
    }
```

Рисунок 2.25 – Фрагмент коду з обробки натискання кнопок

Спочатку обробляються кнопки тих функцій, які не будуть заблоковані у вихідні дні. Це функції видачі розкладу занять та визначення навчального тижня. При виборі кнопки видачі розкладу занять користувачеві пропонується ввести назву групи, а при виборі кнопки визначення навчального тижня розпочинається функція DetermineWeek()

Після цього відбувається перевірка дня тижня. Якщо день тижня – субота чи неділя, і при цьому була нажата кнопка пошуку викладача, пошуку групи чи пошуку студента, то користувач отримує повідомлення про те, що обрана ним функція заблокована.

Далі йде обробка кнопок пошуку викладача, пошуку групи і пошуку студента. Користувачеві пропонується ввести прізвище викладача, студента чи назву групи, залежно від обраної кнопки, після чого задається значення для змінної t. Якщо виявляється, що не була натиснута жодна з вище перечислених кнопок, а змінна t рівна нулю, користувачеві повідомляється, щоб він обрав необхідну функцію.

Якщо прийшло нове повідомлення, а змінна t відрізняється від нуля, це означає що користувач вже обрав функцію перед цим повідомленням, отже повідомлення мусить містити прізвище викладача, студента, або назву групи. Розпочинається обробка цих повідомлень: в залежності від значення змінної t запускається функція пошуку викладача FindTeacher(), функція пошуку студента FindStudent(), функція пошуку групи FindGroup() або функція видачі розкладу GiveSchedule(). Код обробки цих повідомлень зображений на рисунку 2.26.

```
//Пошук викладача
else if (t == 1 && ( text == e?.Message?.Text))
{
    FindTeacher();
}

//Пошук групи
else if (t == 2 && (text == e?.Message?.Text))
{
    FindGroup();
}

//Пошук студента
else if (t == 3 && (text == e?.Message?.Text))
{
    FindStudent();
}

//Видача розкладу
else if (t == 4 && (text == e?.Message?.Text))
{
    GiveSchedule();
}
```

Рисунок 2.26 – Фрагмент коду обробки повідомлень

3 ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ІНСТРУКЦІЇ ЩОДО ЇЇ ВИКОРИСТАННЯ

Для початку роботи з інформаційним телеграм-ботом КТКТ необхідно мати аккаунт у сервісі Telegram і знайти цього бота у пошуку або перейти за посиланням «t.me/InformationKTKTBot». Інструкції щодо пошуку чат-бота в сервісі Telegram зображені на рисунку 3.1.

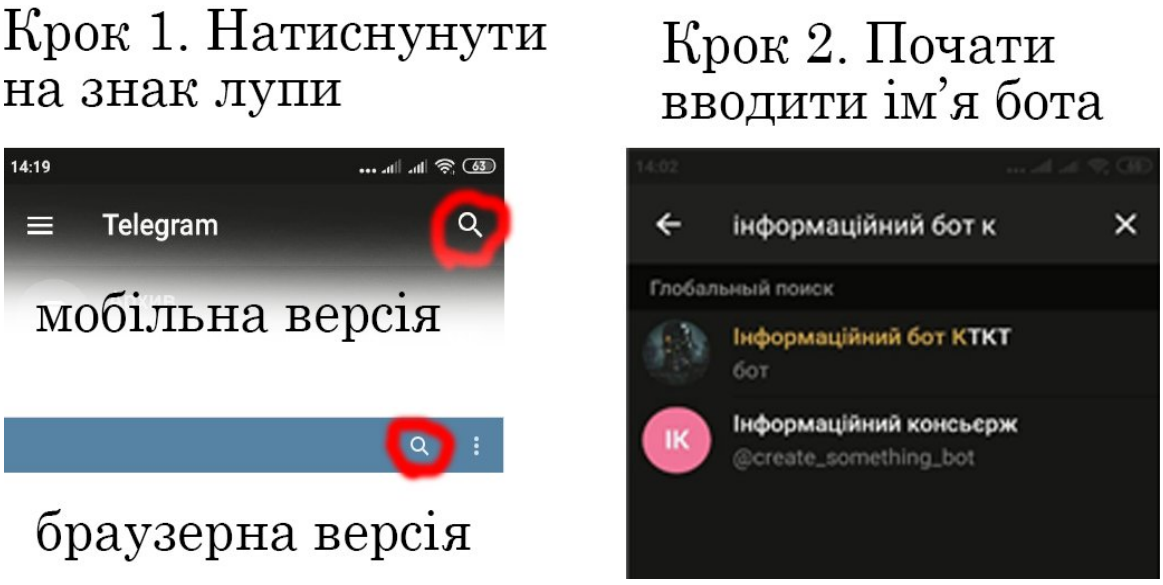


Рисунок 3.1 – Інструкції щодо пошуку чат-бота в сервісі Telegram

Після знаходження чат-бота, його можна буде побачити у “сплячому” режимі. У цьому режимі бот не може надсилати до вас повідомлення, тому його необхідно “пробудити”. Для цього потрібно натиснути вбудовану кнопку «СТАРТ».

Описаний вище процес є обов’язковим для будь-якого чат-бота у Telegram. Виключення можуть складати лише Inline-боти. У випадку з Inline-ботами, ім’я бота, починаючи зі знаку «@» (наприклад, @InformationKTKTBot), згадується у будь-якому загальнодоступному чаті з кількома користувачами, після чого бот сам приєднується до цього чату. Демонстрація дій щодо запуску бота зображена на рисунку 3.2.

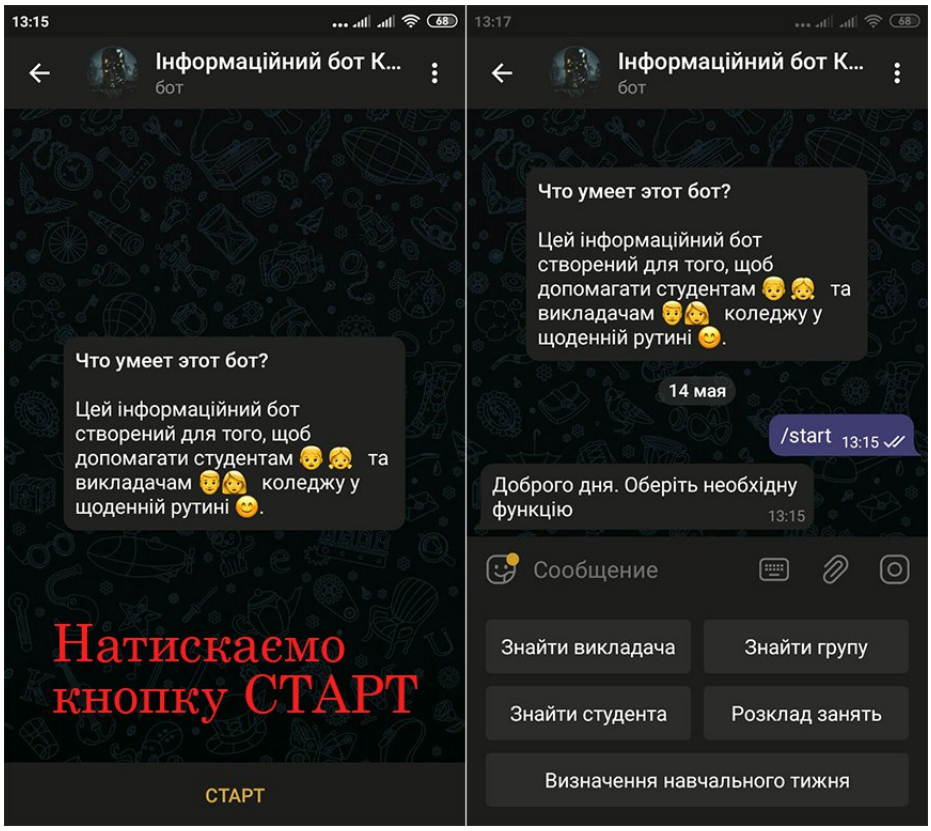


Рисунок 3.2 – Демонстрація дій щодо запуску бота

Натиснення кнопки «СТАРТ» є аналогічним команді «/start», і обробляється як команда «/start», отже в нашому випадку після її натиснення розпочинається ініціалізація клавіатури.

Також користувач завжди може знову призупинити бота за допомогою спеціальних кнопок, які є вбудованими у телеграм-клієнті, і знову запустити бота за допомогою команди «/start».

Після ініціалізації клавіатури можна обирати потрібну функцію. Функції знаходження викладача, студента та групи доступні лише у будні дні. Функція визначення тижня і функція видачі розкладу доступні у будь-який день.

Після обрання функцій пошуку викладача, студента чи групи, а також після вибору функції видачі розкладу користувачеві пропонується ввести певні дані: прізвище викладача, прізвище студента чи назву групи, в залежності від обраної функції. Демонстрація загальних інструкцій щодо роботи з функціями зображена на рисунку 3.3.

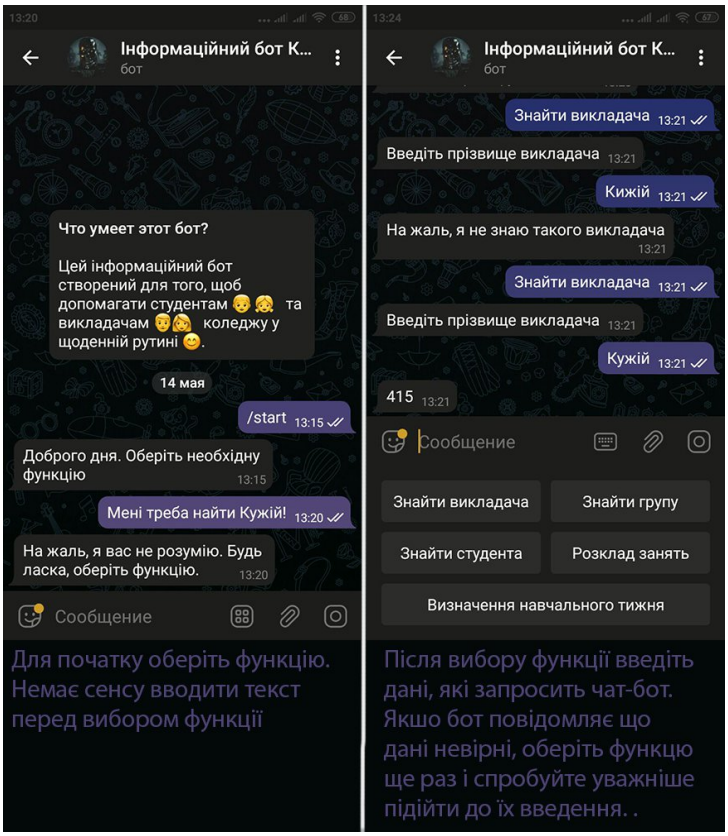


Рисунок 3.3 – Демонстрація загальних інструкцій щодо роботи з функціями

Якщо дані були введені неправильно, бот про це повідомляє. Якщо після цього користувач бажає спробувати ввести дані ще раз, йому необхідно перед цим ще раз обрати необхідну функцію. У іншому випадку бот повідомить про це користувачу. Це зумовлено тим, що при неправильному введенні даних прапорець t також скидується, і функція вважається виконаною.

Якщо функцію не обрано, а користувач ввів повідомлення, бот повідомляє йому, що необхідно обрати функцію.

У функції знаходження викладача необхідно ввести лише прізвище викладача в називному відмінку з великої літери, без імені та ініціалів.

У рамках функцій з пошуку групи та видачі розкладу користувачеві пропонується ввести назву групи. Демонстрація роботи функцій з видачі розкладу та пошуку групи зображена на рисунку 3.4.

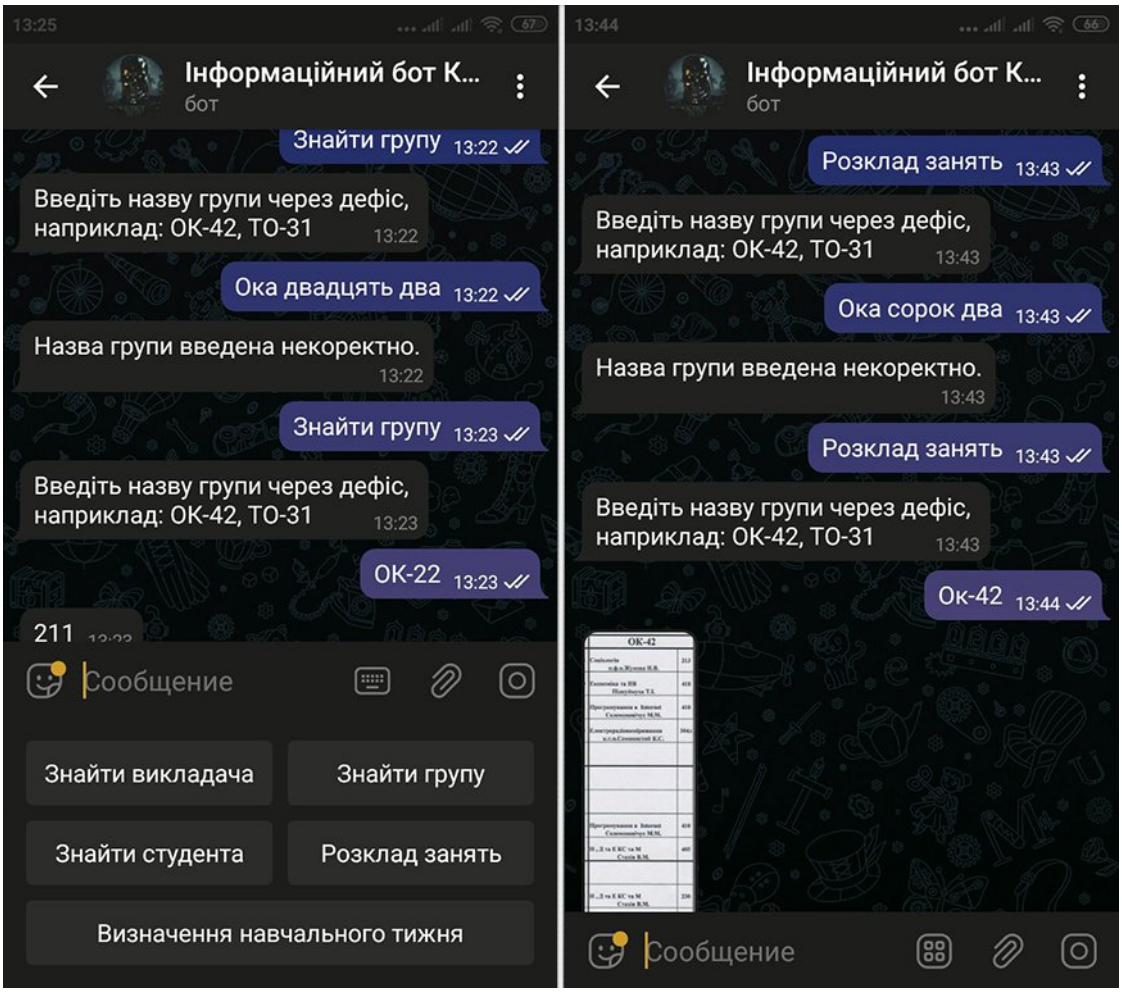


Рисунок 3.4 – Демонстрація роботи функцій з видачі розкладу та пошуку групи

Для повідомлення з назвою групи від користувача є лише одне правило, і помилитись важко: назва групи має бути введена через дефіс. Регістр у цьому повідомленні значення не має, оскільки при перевірці воно все одно підноситься до верхнього регістру.

Функції знаходження викладача, студента та групи блокуються у вихідні дні. На рисунку 3.5 продемонстровано блокування цих функцій та робота функцій видачі розкладу і визначення навчального тижня у вихідний день.

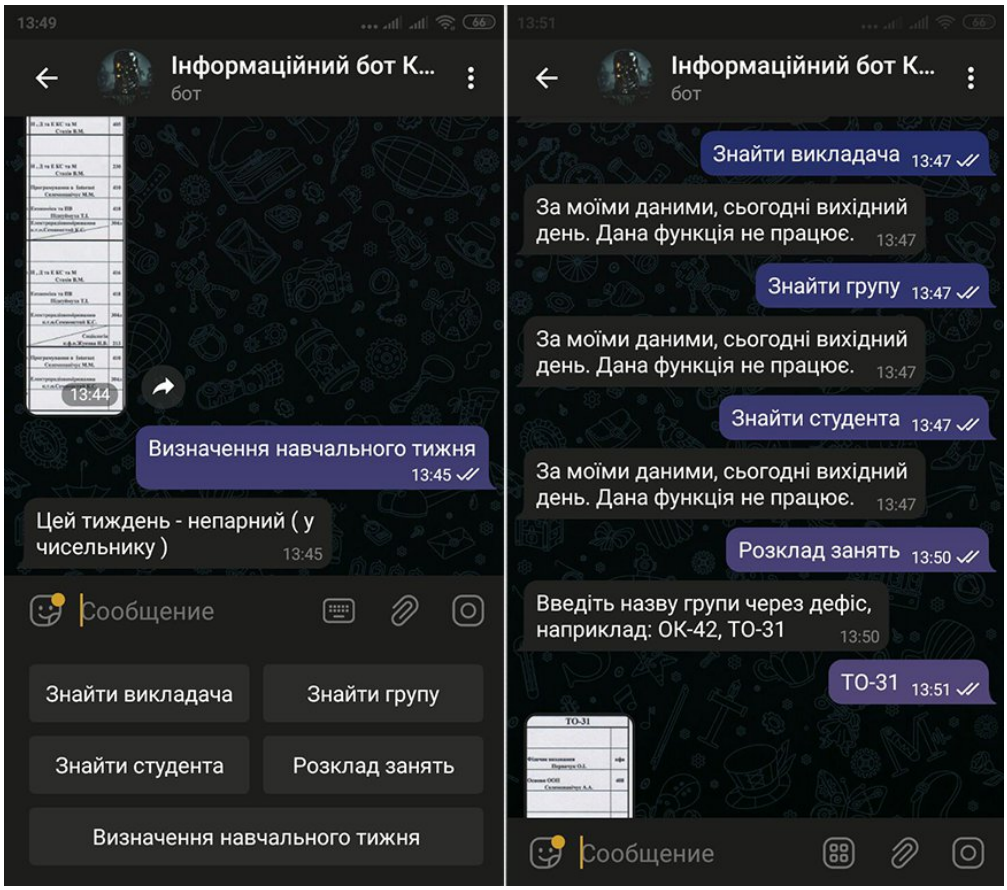


Рисунок 3.5 – Демонстрація роботи функцій чат-бота у вихідні дні

Функція пошуку студента відноситься до функцій, що працюють лише у будні дні. У цій функції необхідно ввести лише прізвище викладача в називному відмінку з великої літери, без імені та ініціалів. У випадку, якщо в коледжі існує кілька студентів з таким прізвищем, чат бот почергово виведе місцезнаходження всіх студентів з цим прізвищем, із вказанням групи, у якій навчається кожен з них.

Демонстрація роботи функції пошуку студента зображена на рисунку 3.6.

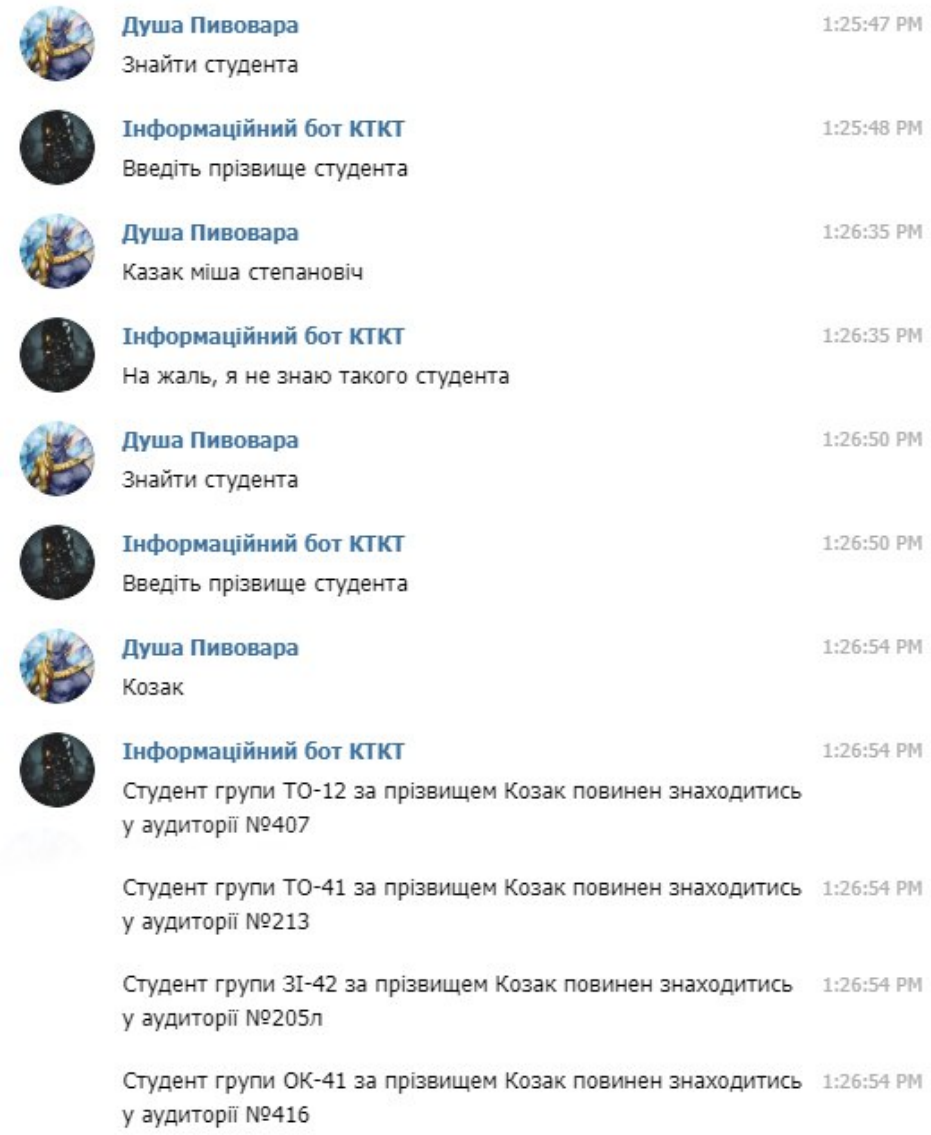


Рисунок 3.6 – Демонстрація роботи функції пошуку студента

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.1 Розрахунок витрат на розробку програмного продукту

Витрати на розробку та впровадження програмного продукту (К) включають:

$$К=К1+К2,$$

де К1 – витрати на розробку програмного продукту, грн.

К2 – витрати на налагодження та дослідну експлуатацію програмного продукту на ПК, грн.

Витрати на розробку програмного продукту включають:

- витрати на оплату праці розробників (Воп);
- нарахування на зарплату (Нз);
- витрати на куповані вироби (Вк);
- накладні витрати (Нв);
- інші витрати (Він).

Для розробки програмного продукту потрібні 4 спеціалісти-розробники, а саме:

- керівник проекту (Кп);
- консультант з економічної частини (Ке);
- консультант з охорони праці (Коп);
- студент-дипломник (Сд).

Згідно з штатним розписом ВСП «Фаховий коледж інформаційних технологій «НУ ЛП», одна година навантажень становить для:

- керівник проекту (Кп) – 118,13 грн;
- консультант з економічної частини (Ке) - 118,13 грн;
- консультант з охорони праці (Коп) - 103,48 грн;
- студент-дипломник (Сд) - 8,73 грн.

Денна оплата студента дипломника визначається:

Стипендія / 173,

де 173 - місячний фонд робочого часу, год.

1510,00 / 173 = 8,73 грн/год

Розрахунок витрат на оплату праці всіх спеціалістів проекту визначається за формулою:

$$Воп=\sum Рі * ti * Зпі,$$

де Рі – чисельність розробників проекту і-спеціальності, роб.;

ti – час, витрачений за розробку проекту розробником і-спеціальності, год.;

Зпі – погодинна заробітна плата розробника і-спеціальності, грн..

Таким чином, витрати на оплату праці розробників складають:

Зкп = 1 * 14 * 118,13 = 1 653,82 грн

Зке = 1 * 1 * 118,13 = 118,13 грн

Зкоп = 1 * 1 * 103,48 = 103,48 грн

Зсд = 1 * 180 * 8,73 = 1 571,4 грн

Сумарні витрати на оплату праці:

Воп = (1* 4 * 118,13) + (1*1 * 118,13) + (1*1 * 103,48) + (1 * 180 * 8,73) =

= 1 653,82 + 118,13 + 103,48 + 1 571,4 = 3 446,83 грн

Розрахунок витрат на оплату праці розробників наведено у таблиці 4.1.

Таблиця 4.1 – Розрахунок витрат на оплату праці

Спеціальність розробника	Кількість розробників, роб.	Час роботи, год.	Погодинна заробітна плата розробника, грн.	Витрати на оплату праці, грн.
Керівник проекту	1	14	118,13	1 653,82
Консультант з економічної частини	1	1	118,13	118,13
Консультант з охорони праці	1	1	103,48	103,48
Студент-дипломник	1	180	8,73	1 571,4
Всього	4	-	-	3 446,83

Нарахування на зарплату визначаються за формулою:

$$Нз = (Воп- Зсд) * 22,0 / 100,$$

де Воп – витрати на оплату праці, тис.грн.

22 – норматив нарахувань на зарплату, %

$$Нз= (3\ 446,83-1\ 571,4) * 22,0 / 100 = 412,59 \text{ грн}$$

Витрати на куповані вироби визначаються за їх фактичними цінами з врахуванням найменування, номенклатури та необхідної кількості в проекті. Транспортно-заготівельні витрати становлять 10% від суми витрат на куповані вироби.

У таблиці 4.2 наведено розрахунок витрат на куповані вироби

Таблиця 4.2 – Розрахунок витрат на куповані вироби.

Найменування купованих виробів	Одиниця виміру	Кількість, шт.	Ціна за одиницю, грн.	Сума витрат, грн.
Папір А4 New Future Laser 80 г/м2	Пачка	1	250	250
Роздрук пояснювальної записки	Аркуш	80	3	240
Папка для дипломного проєкту	-	1	230	230
Разом	-	-	-	520
Транспортно-заготівельні витрати (10%)	-	-	-	52
Всього	-	-	-	572

Витрати на куповані вироби становлять:

$$Вк = 250 + 240 + 230 + 52 = 572 \text{ грн}$$

Накладні витрати становлять 30% від витрат на оплату праці:

$$Нв = 3\ 446{,}83 * 30 / 100 = 1\ 034{,}05 \text{ грн}$$

Інші витрати обчислюються по їх питомій вазі у структурі собівартості (10%) :

$$Він = (Воп + Нз + Нв + Вк) * 10 / 90$$

$$Він = (3\ 446{,}83 + 412{,}59 + 1\ 034{,}05 + 572) * 10 / 90 = 607{,}27 \text{ грн}$$

Витрати на розробку програмного продукту визначаються за формулою:

$$К1 = Воп + Нз + Нв + Вк + Він,$$

$$К1 = 3\ 446,83 + 412,59 + 1\ 034,05 + 572 + 607,27 = 6\ 072,74 \text{ грн}$$

4.2 Розрахунок витрат на налагодження та дослідну експлуатацію програмного продукту на ПК

Програма була розроблена на протязі 30 днів із розрахунком 6 годин на день (t 180 год.).

Потужність комп'ютерної техніки (Р) включає ПК який споживає 1,2 кВт/год. Вартість однієї машино-години роботи визначається за формулою:

$$Sm.r. = P * Тф,$$

де Р – потужність комп'ютерної техніки, кВт;

Тф - вартість 1 кВт-год електроенергії, грн. (Тф = 7,50).

$$Sm.r. = 1,2 * 7,50 = 9 \text{ грн/год}$$

Витрати на налагодження та дослідну експлуатацію програмного продукту на ПК визначаються за формулою:

$$К2 = Smr * t,$$

де Sm.r. – вартість однієї машино-години роботи, грн \ год.

t – машинний час, витрачений на налагодження та дослідну експлуатацію програмного продукту, год.

$$К2 = 9 * 180 = 1\ 620 \text{ грн}$$

Таким чином, витрати на розробку та впровадження програмного продукту становлять:

$$К = 6\ 072,74 + 1\ 620 = 7\ 752,74 \text{ грн}$$

У таблиці 4.3 наведено кошторис витрат на розробку та впровадження програмного продукту.

Таблиця 4.3 Кошторис витрат на розробку та впровадження програмного

Найменування елементів витрат	Сума витрат, грн.
Витрати на оплату праці	3 446,83
Нарахування на зарплату	412,59
Витрати на куповані вироби	572
Накладні витрати	1 034,05
Інші витрати	607,27
Витрати на налагодження та дослідну експлуатацію	1 620
Всього:	7 752,74

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

Охорона праці – це надзвичайно важлива система, яка складається з різноманітних правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів. Вона має на меті збереження життя, здоров'я та працездатності людини у процесі трудової діяльності. У цьому контексті, важливими об'єктами охорони праці є людина в процесі праці, середовище виробництва та організації праці на ньому.

На підприємствах вкрай важливо створювати безпечні умови праці, адже це не тільки зберігає здоров'я та життя працівників, але й сприяє ефективнішому виробництву. Відповідно, слід дотримуватися вимог, викладених у нормативних документах. Це значно знизить наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з відеодисплейними терміналами.

5.1 Електробезпека

Для забезпечення безпечної та ефективної експлуатації ЕОМ та ВДТ необхідно дотримуватись певних принципів та правил. Перш за все, регулярні щомісячні діагностичні огляди та чистка пристроїв допоможуть уникнути накопичення пилу, які можуть спричинити утворення електростатичного струму та пробою. Під час оглядів також необхідно оцінювати стан радіокомпонентів та замінювати їх на компоненти аналогічних номіналів за необхідності, щоб забезпечити найвищу продуктивність пристроїв. Крім того, монтаж електропроводів, кабелів та ЕОМ, а також їх ремонт та обслуговування має проводитись за допомогою діелектричних засобів та при повному відімкненні від загальної мережі для усунення можливості утворення струмів короткого замикання або струмів перенавантаження.

Електропроводи та кабелі, а також ЕОМ з ВДТ і ПП мають відповідати виконанням та ступеню захисту класу зони за НПАОП 40.1-1.32-01. Для

забезпечення безпеки працівників, які працюють з ЕОМ та персональними комп'ютерами, необхідно дотримуватись вимог електробезпеки, встановлених нормативними документами, зокрема: «Правила улаштування електроустановок» (ПУЕ) та «Правилах охорони праці під час експлуатації електронно-обчислюваних машин» (НПАОП 0.00-1.28-10.). Також важливо враховувати, що лінія електромережі для живлення ЕОМ з ВДТ має бути виконана як окрема групова трипровідна мережа з фазового, нульового та захисного провідника. Усі провідники мають відповідати вимогам НПАОП 40.1-1.32-01. Нульовий захисний провідник має бути прокладений від стійки групового розподільного щита. ЕОМ з ВДТ має під'єднуватись до електромережі лише за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення.

У приміщенні, де одночасно експлуатується або обслуговується більше ніж п'ять персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Неприпустимим є підключення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ до звичайної двопровідної електромережі, в тому числі — з використанням перехідних пристроїв. Загалом, дотримання цих принципів та правил забезпечить безпечну та ефективну роботу ЕОМ та ВДТ.

5.2 Техніка безпеки при роботі за комп’ютером

Під час роботи на комп'ютері можуть діяти шкідливі фактори, які можуть вплинути на здоров'я користувача. Інформаційна безпека та електробезпека забезпечуються за допомогою ряду профілактичних заходів. Один з найважливіших аспектів є саме електробезпека. Для усунення ризику ураження струмом необхідно розміщувати обладнання та кабелі відповідно до вимог безпеки. Це може бути досягнуто за допомогою захисного заземлення, використання безпечних розеток та електропроводки, розрахованих на потужність системи, а також ізоляції всіх проводів.

Для забезпечення ефективності та продуктивності роботи комп'ютера важливо регулярно чистити внутрішні частини від пилу, користуватися окремими вогнетривкими столами для комп'ютерів та іншого устаткування. Щоб запобігти іскрінню, необхідно рідше вставляти та виймати вилки з розеток.

Освітлення на робочому місці повинно відповідати характеру зорової роботи, який визначається трьома параметрами: об'єктом розрізнення, фоном та контрастом об'єкта і фону. Потрібно забезпечити рівномірне розподілення яскравості на моніторі та навколишньому просторі, відсутність різких тіней, відблисків та стабільну освітленість під час роботи. Вибирати оптимальну спрямованість світлового потоку та необхідний склад світла. Для забезпечення здоров'я користувача можна використовувати спеціальні світильники та підсвічування, які забезпечують оптимальні умови для зорової роботи. КПО>1,5%.

Правила безпеки при роботі за комп'ютером:

- Увімкніть кондиціонер у приміщенні, щоб уникнути перегрівання пристроїв та забезпечити комфортну температуру для користувача.
- Переконайтесь у стабільному розташуванні обладнання на столі. Не забудьте, що нестабільна підставка для монітора може призвести до його падіння та пошкодження. Відкрийте монітор так, щоб було зручно спостерігати екран – прямо (не збоку) і трохи зверху вниз, з нахилом екрана, його нижній край ближче до користувача.
- Перевірте загальний стан обладнання, справність електропроводки, кабелів, вилок, розеток та заземлення захисного екрана. Якщо щось несправне, виправте це перед роботою.
- Налаштуйте освітлення робочого місця. Краще мати якісне та достатнє освітлення, щоб зменшити навантаження на очі та запобігти їх перевтомленню.

-Регулюйте та фіксуйте висоту крісла та зручний нахил спинки для користувача. Важливо, щоб користувач почував себе комфортно та міг працювати тривалий час без відчуття дискомфорту.

-Під’єднайте необхідне обладнання до системного блоку. Кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері, щоб уникнути пошкодження пристроїв та пошкодження даних.

- Увімкніть комп'ютерне обладнання послідовно: монітор, системний блок, принтер (якщо потрібно друкувати). Це дозволить уникнути перезавантаження системи та забезпечити її стабільну роботу.

-Налаштуйте яскравість монітора та контрастність. Не робіть зображення надто яскравим, щоб не перевтомлювати очі. Також важливо відпочивати та робити паузи під час тривалої роботи за комп'ютером, щоб не перенапружувати очі та запобігти втомі.

5.3 Пожежна безпека

Пожежна безпека об'єкта – це важливий стан об'єкта, про який регламентом визначено імовірність виникнення та розвитку пожежі та впливу на людей її небезпечних факторів, а також забезпечується захист матеріальних цінностей. Цей стан регулюється нормативними актами, зокрема ДБН В.1.1-7-2002 "Пожежна безпека об'єктів будівництва", ДСТУ 2272:2006 "Пожежна безпека. Терміни та визначення основних понять", НАПБ А.01.001-2004 "Правила пожежної безпеки в Україні".

Важливою складовою пожежної безпеки є належне ознайомлення всіх працівників з правилами пожежної безпеки, проходження протипожежних інструктажів та перевірки знань з питань пожежної безпеки. Зокрема, автоматична пожежна сигналізація повинна завжди бути у ввімкненому, черговому стані.

Приміщення для роботи повинно відповідати нормам, зазначеним у НАПБ А.01.001.-2004 про будівлі та приміщення для ЕОМ. Зокрема, підлога та стіни в такому приміщенні повинні входити в групу горючості Г1. Заборонено зберігати легкозаймисті та горючі речовини в будь-яких кількостях в приміщенні з ЕОМ.

Важливо не залишати ЕОМ без нагляду під час роботи та вимикати їх від глобальної електромережі по закінченню роботи з ними. На робочому місці має бути встановлений легкий доступ до двох газових вогнегасників як засобів первинної пожежної безпеки. Меблі та обладнання повинні розміщуватися таким чином, щоб забезпечувався вільний евакуаційний прохід до дверей виходу з приміщення (завширшки не менше 1 м). Евакуаційні шляхи та виходи необхідно постійно утримувати вільними, нічим не захаращувати. Документи, папір та інші горючі матеріали слід зберігати на відстані не менше 1 м від електрощитів і електрокабелів; 0,5 м від електросвітильників; 0,6 м від сповіщувачів пожежної сигналізації та 0,15 м від приладів центрального водяного опалення. Засоби протипожежного захисту слід утримувати у справному стані. Відстань від найбільш віддаленого місця до вогнегасника не повинна бути більшою за 20 м.

Приміщення, у яких розміщені ПЕОМ, слід оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-2 або один ВВПА-400 на три ПЕОМ, але не менше ніж один вогнегасник зазначених типів на приміщення.

Крім того, рекомендується забезпечити додатковий простір для працівників, який забезпечить їм додаткові можливості для руху та роботи з ПЕОМ, а також забезпечити доступ до відповідних джерел електроживлення. З метою забезпечення повної пожежної безпеки рекомендується проводити щорічні перевірки на відповідність пожежним нормам та стандартам. Для гасіння пожеж класу В (горючі рідини) та класу Е (електроустановки, що знаходяться під напругою) вуглекислотні вогнегасники (ВВК) ефективно застосовуються. Їх механізм дії полягає в тому, що вони працюють шляхом витіснення кисню з зони горіння та охолодження матеріалів, що горять. Вогнегасники ВВК не залишають слідів після використання, що робить їх ідеальними для гасіння пожеж в електронному обладнанні та на об'єктах з цінними матеріалами.Вуглекислотний вогнегасник ВВК-2 наведено на рисунку 5.1

Рисунок 5.1 – Вогнегасник вуглекислотний ВВК-2.

5.4 Виробниче приміщення та робоче місце

Згідно з вимогами ДСанПіН 3.3.2.007-98, приміщення, в яких планується робота з ВДТ, повинні відповідати проєктній документації будинку, погодженій з уповноваженими державними органами. Якщо ви плануєте працювати з відомчими документами та матеріалами на комп'ютері, то вам необхідно взяти до уваги правила розташування робочого місця, які забезпечать вам максимальний комфорт та безпеку під час роботи.

Розміщення робочих місць з ВДТ у підвальних приміщеннях та на цокольних поверхах заборонено. Якщо ваше робоче місце знаходиться в приміщенні, то зверніть увагу на доступність основних умов, необхідних для продуктивної роботи. На одного працівника площа робочого місця має становити не менше ніж 6,0 м^2, а об'єм – не менше ніж 20,0 м^3.

У залежності від того, яку роботу ви виконуєте, має бути враховано чинні санітарні норми освітлення, температури, відносної вологості повітря, сили та ступеня вібрації, звукового шуму, вогнестійкості, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря.

Щоб забезпечити високий рівень комфорту та безпеки під час роботи з комп'ютером, на кожну кімнату, де працюють співробітники, повинні бути наявні елементи природного та штучного освітлення відповідно до ДБН В.2.5-28-2006. Для досягнення максимального рівня безпечності та охорони праці при роботі з комп’ютером виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації й вогнегасниками. На вікнах повинні бути встановлені сонцезахисні плівки.

При розташуванні елементів робочого місця користувача ПК необхідно враховувати робочу позу користувача, простір для розміщення користувача, можливість огляду елементів робочого місця, можливість ведення записів, розміщення документації та матеріалів, якими користуватиметься працівник. Робочі місця з ПК мають бути розташовані від стіни з вікнами на відстані не менше ніж 1,5 м, від інших стін – на відстані не менше ніж 1 м. Недопустиме таке розташування ПК, при якому працівник повернутий обличчям або спиною до вікон кімнати або до задньої частини ПК, у яку вмонтовані вентилятори. Загальні рекомендації до робочої пози та робочого місця наведені на рисунку 5.2

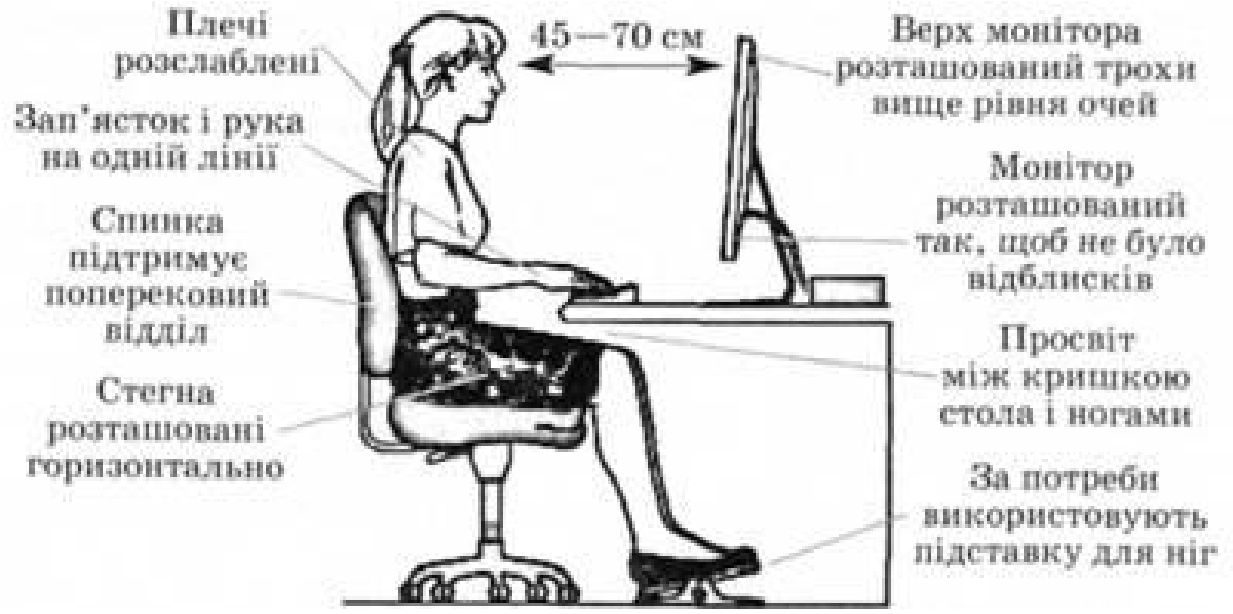


Рисунок 5.2 – Рекомендації до робочої пози та робочого місця

5.5 Висновки до розділу з охорони праці та безпеки життєдіяльності

Дипломний проєкт був виконаний з дотриманням усіх зазначених нижче стандартів та норм, що гарантували мою безпеку під час роботи:

- Електробезпека. Під час використання комп'ютера, були виконані всі вимоги електробезпеки, вказані в технічній документації виробника;

-Техніка безпеки при роботі за комп'ютером. Всі вимоги щодо безпеки користувача під час роботи за комп'ютером, такі як правильне розташування апаратури, ергономіка робочого місця, перерви в роботі та забезпечення зорового комфорту, були дотримані належним чином;

-Пожежна безпека. Умови зберігання та експлуатації, що забезпечують пожежну безпеку, згідно з технічною документацією виробників комп'ютерів, були виконані протягом усієї роботи;

-Виробниче приміщення та робоче місце. Робота над дипломним проєктом відбувалась в приміщенні з робочим місцем, яке відповідало всім стандартам та ергономічним нормам.

ВИСНОВКИ

У даному дипломному проекті був спроектований, створений та описаний чат-бот для месенджеру Telegram, призначенням якого є видача необхідної інформації для студентів та викладачів коледжу.

Під час роботи були виконані наступні задачі:

- Аналіз предметної області
- Огляд існуючих рішень, виявлення їх переваг, недоліків та особливостей
- Вибір сервісу для розміщення чат-бота
- Вибір інструментів розробки
- Вибір способу отримання оновлень
- Розробка алгоритму роботи чат-бота
- Створення та налаштування аккаунту для чат-бота у сервісі Telegram
- Структурування інформації, отриманої із загальнодоступних джерел для подальшого використання у функціях бота
- Написання коду для чат-бота у середовищі VisualStudio
- Тестування роботи чат-бота, складання інструкції щодо його використання

Цей чат-бот було розроблено на мові програмування C#, у інтегрованому середовищі розробки MicrosoftVisualStudioCommunity із застосуванням бібліотеки Telegram.bot, оскільки у якості платформи для розміщення чат-бота серед наявних месенджерів було обрано саме Telegram. У якості системи контролю версій було використано Git, у якості сховища для зображень було використано репозиторій GitHub.

ПЕРЕЛІК ПОСИЛАНЬ

1 https://visualstudio.microsoft.com/
Microsoft Visual Studio.
2. https://telegrambots.github.io/book/
Telegram Bots Book
3. https://core.telegram.org/bots
Bots: An introduction for developers
4. https://www.sohamkamani.com/blog/2016/09/21/making-a-telegram-bot/
Making a Telegram Bot
5. https://www.pubnub.com/blog/http-long-polling/
What is HTTP Long Polling?
6. Albahari J., Albahari B.C# 7.0 Pocket Reference. O'Reilly Media, 2017.
7. https://tlgrm.ru/docs/bots/samples#c-sharp
Документация Telegram: Примеры ботов
8. https://vc.ru/services/66319-chat-boty-chto-biznes-dumaet-o-virtualnyh-pomoshchnikah
Чат-боты: что бизнес думает о виртуальных помощниках
9. http://www.epochta.ru/blog/articles/chat-bots/.
Чат-боты в маркетинге и бизнесе: функции, роли, возможности
10. https://privatbank.ua/ru/udalenniy-banking/chat-boty
Чат-боты: от развлечений до бизнес-решений
11. https://mediamonitoringbot.com/ua
MediaMonitoringBot - Telegram-бот моніторингу згадок в онлайн-ЗМІ України
12. https://www.poglyad.tv/ukrayinski-telegram-boty-yaki-zminyat-zhyttya-na-krashhe/
Українські Telegram-боти, які змінять життя на краще

ДОДАТОК А

Лістинг програми

```
usingSystem;
usingSystem.Text;
usingTelegram.Bot;
usingTelegram.Bot.Types;
usingTelegram.Bot.Args;
usingTelegram.Bot.Types.InputFiles;
usingTelegram.Bot.Types.ReplyMarkups;
usingSystem.Runtime.CompilerServices;
namespaceDemonGaizerBot
{
classProgram
   {
privatestaticITelegramBotClientbotClient;
privatestaticint t = 0;
privatestaticint d;
conststring V = "Пара відсутня";
       //Викладачі
privatestaticstring[][,] teachers = newstring[][,]
{newstring[,] { { null, "Семянистий","Підкуймуха", "Кужій", "Маніта", "Заліська", "Шеремета", "Цебенко",
"Бохонко", "Романюк", "Мужик", "Бліндер", "Полець", "Пелещак", "Походжай", "Чайковський", "Кубик",
"Литвин", null } },
 /*                                ПОНЕДІЛОК                    ВІВТОРОК                    СЕРЕДА
ЧЕТВЕР                        ПЯТНИЦЯ
   /* Семянистий */   newstring[,] { { V, V, V, "304л", V },              { V, V, V, V, V },                    { V, V, V,
"304л", "304л ( по чис. )" },        { V, "304л", V, "304л", V },              { "304л", "304л", "304л ( по чис. )", V, V } },
   /* Підкуймуха */   newstring[,] { { "418", "418", "418", "418", V },     { "418", "418", "418", "418", V },          { "418",
"418", "418", "418", V },             { "418", "418", "418", "418", V },          { V, V, V, V, V } },
   /* Кужій */        newstring[,] { { V, V, V, V, V },                  { "415", "415", "415", "415", V },          { "415", "415",
"415", "415 ( по чис )", V },   { V, V, "415", "415", V },                { V, V, V, V, V } },
   /* Маніта */       newstring[,] { { V, V, "226", "226", V },           { V, V, V, V, V },                    { V, V, "226",
"226", V },                  { V, V, V, V, V },                    { "211", "211", "211", V, V } },
   /* Заліська */     newstring[,] { { "312л", "312л", "312л", "312л", V }, { "312л", "312л", "312л", "312л", V },
{ "312л", "312л", "312л", "312л", V },         { V, "312л", "312л", "312л", V },          { V, V, V, V, V } },
   /* Шеремета */     newstring[,] { { "413", "413", "413", "413", V },     { "413", "413", "413", "413", V },          { V, V,
V, V, V },                    { V, V, V, V, V },                    { V, V, V, V, V } },
   /* Цебенко */      newstring[,] { { "302", "302", "302", "302", "302" }, { "302", "302", "302", "302", V },          { V, V,
V, V, V },                    { "302", "302", "302", "302", V },          { "302", "302", "302", V, V } },
   /* Бохонко */      newstring[,] { { V, V, "416", "416", V },           { V, V, V, V, V },                    { "416", V, V, V,
V },                    { V, V, V, V, V },                    { V, V, "416", V, V } },
   /* Романюк */      newstring[,] { { V, "201л", "201л", V, V },         {"201л ( по знам.)","201л","201л","201л", V },
{ V, "201л", "201л", "201л", V },             { "201л ( по чис. )", "201л", V, V, V },    { "201л", "201л", V, V, V } },
   /* Мужик */        newstring[,] { { V, V, V, V, V },                  { V, "204л", "204л", V, V },                { V, V, V, V,
V },                    { V, V, V, V, V },                    { V, V, V, V, V } },
   /* Бліндер */      newstring[,] { { V, V, V, V, V },                  { V, "305л", V, V, V },                   { "205л ( по чис.)",
V, "205л", "205л", V },    { V, V, "205л", "205л", V },              { V, "305л", V, V, V } },
   /* Полець */       newstring[,] { { "305л ( по знам.)", V, V, V, V },    { V, V, V, V, V },                    { V, V, V,
"305л ( по знам.)", V },            { V, V, V, V, V },                    { "305л ( по знам.)", V, V, V, V } },
   /* Пелещак */      newstring[,] { { V, "419", "419", V, V },           { V, "419", V, "419", V },                { V, "419",
"419", V, V },                  { V, V, V, V, V },                    { V, V, V, V, V } },
   /* Походжай */     newstring[,] { { V, V, V, V, V },                  { V, V, V, V, V },                    { V, "209л", "209л",
V, V },                { V, V, V, V, V },                    { "209л", "209л", V, V, V } },
   /* Чайковський */  newstring[,] { { V, V, "204л", "204л", V },         { V, V, V, V, V },                    { V, V, V, V,
V },                    { V, V, V, "204л", "204л" },              { V, V, V, V, V } },
   /* Кубик */        newstring[,] { { V, "414 ( по знам.)", "414", V, V }, { "414", "414", "414", V, V },          { V, V, V,
V, V },                    { V, V, "414", "414", V },                { "414", "414", V, V, V } },
```

```
  /* Литвин */      newstring[,] { { V, V, V, V, V },                { V, V, "406", "406", V },              { "406", "406",
"406", V, V },              { "406", "406", V, V, V },              { V, "406", "406", V, V }},
};
      //Групи
privatestaticstring[][,] groups = newstring[][,]
{newstring[,] { { null, "ЗІ-12","ЗІ-13", "ОК-11", "ТО-11", "ТО-12", "ЗІ-11", "ОК-12", "ТО-21", "ЗІ-21", "ЗІ-22", "ОК-
21", "ОК-22", "ТО-31", "ТО-32", "ЗІ-31", "ЗІ-32", "ОК-31", "ОК-32", "ТО-41", "ТО-42", "ЗІ-41", "ЗІ-42", "ОК-41",
"ОК-42", null } },
 /*                              ПОНЕДІЛОК  0                    ВІВТОРОК  1                    СЕРЕДА  2
ЧЕТВЕР 3                              ПЯТНИЦЯ  4
  /* ЗІ-12 */      newstring[,] { { "407", "413", "405", "401", V },        { "214", "401", "413", "211", V },        { V,
"302/кфв", "230", "213", V },                { "407", "405", "230", "302", V },              { V, "кфв", "211", V, V } },
  /* ЗІ-13 */      newstring[,] { { "кфв (чис.)", "406", "413", "226", V },    { V, "302", "401", "413", V },
{ "405", "211", "407", "230", V },                { "кфв", "230", "405", "401", V },              { "213", "214", "401
(чис.)", V, V } },
  /* ОК-11 */      newstring[,] { { V, V, "302", "413", V },              { "215", "413", "404", "214", V },
{ "302/кфв", V, "226", "226", V },                { "406", "413", "406", "215", V },              { "кфв", "215", "406", V,
V } },
  /* ТО-11 */      newstring[,] { { V, "405", "404", "407", V },              { "404", "211", "226", "215", V },        { "213",
"413", "302/кфв", V, V },                { "404", "215", "407", "кфв", V },              { "405", "211", "214", V, V } },
  /* ТО-12 */      newstring[,] { { V, "404", "407", "405", V },              { "211", "404", "215", "226", V },        { "413",
"213", "кфв/302", "226", V },                { "404", "215", "407", "кфв", V },              { "214", "405", V, V, V } },
  /* ЗІ-11 */      newstring[,] { { V, "407", "401", "404", V },              { "401", "214", "211", "302", V },        { "302",
"кфв", "213", "225", V },                { "225", "407", "404", "405", V },              { "211", "кфв", "405", V, V } },
  /* ОК-12 */      newstring[,] { { "413", "215", "226", "302", V },        { "413", "215", "214", "404", V },        { V,
"кфв/302", "211", "406", V },                { "413", "406", "кфв", "406", V },              { "416", "203", V, V, V } },
  /* ТО-21 */      newstring[,] { { "405", "211/301", "312л", "301л", V },    { "407", "226", "309л", "401", V },
{ "211", "312л", "301", "кфв", V },                { "401/кфв", "225", "301", "312л", V },        { "407", "215", "301л",
V, V } },
  /* ЗІ-21 */      newstring[,] { { V, "кфв (чис.)","301/211","312л", "301" },  { "312л", "309л", "407", "230", V },
{ "312л", "301", "301л", "302", V },                { "405", "211", "226", "230", V },              { "301л/401", "407",
"кфв", V, V } },
  /* ЗІ-22 */      newstring[,] { { "301/401", "301/211", "кфв", "301", V },    { "309л", "407", "301", "312л", V },
{ "215", "405", "312л", "кфв (знам.)", V },              { "211", "401", "312л", "2226", V },              { "215", "301", "407",
V, V } },
  /* ОК-21 */      newstring[,] { { "401 (чис.)", "312л", "211/301", "215", V },{ "301", "312л", "406", "301", V },
{ "404", "215", "406", "312л", V },                { "кфв/301", "301", "401", "211", V },            { "302", "230", "кфв",
V, V } },
  /* ОК-22 */      newstring[,] { { "312л", "401", "215", "211 (чис.)", V },    { "кфв", "301", "312л", "406", V },
{ "406", "404", "215", "301", V },                { "301/401", "312л", "211", "кфв/301", V },        { "301", "302", "230",
V, V } },
  /* ТО-31 */      newstring[,] { { V, "кфв", "408", V, V },              { "кфв", "415", "211л", "211л", V },
{ "306л", "415", "211л", "211л", V },                { "302", "408", "204л/211л", "211л", V },        { "408", "306л",
"302", V, V } },
  /* ТО-32 */      newstring[,] { { "408", "306л", "306л", "кфв", V },        { "415", "кфв", "306л", "306л", V },
{ V, "306л", "415", "215", V },                { "215", "306л", "306л", "408", V },            { V, "408", "306л", V,
V } },
  /* ЗІ-31 */      newstring[,] { { V, "408", "кфв", "204л", V },            { "204л", "201л", "302", "415", V },
{ "419", "201л", "305л", "415 (чис.)", V },              { V, "302", "408", "305л", "204л" },            { "419", "кфв", "408
( чис. )", V, V } },
  /* ЗІ-32 */      newstring[,] { { V, "305л", "204л", "408", V },            {"302","204л","415","201л", V },
{ "кфв", "4008", "201л", "415 ( знам.)", V },            { "419", "305л", "302", "204л", V },            { "кфв", "419", "408
( знам. )", V, V } },
  /* ОК-31 */      newstring[,] { { "302", "414", "416", "302", V },        { "414", "408", "414", "314", V },
{ V,"кфв", "408", "306л", V },                { V, "314", "кфв", "414", V },              { "414", "416", "404", V,
V } },
  /* ОК-32 */      newstring[,] { { "302", "414", "416", V, V },            { V, "414", "408", "414", V },            { "416",
"314", "кфв", "408", V },                { V, "кфв", "414", "314", "302" },            { "306л", "414", "416", V, V } },
  /* ТО-41 */      newstring[,] { { V, "419", "201л", V, V },              { "213", "418", "201л", "419", V },
{ "415(знам.)", "419","209л","201л (чис.)","213 (чис.)" }, { "201л (чис.)", "418", "213", "415", V },        { "201л",
"209л", V, V, V } },
```

```
  /* ТО-42 */       newstring[,] { { V, "201л", "419", V, V },                { "201л (знам.)", "419", "418", "213", V },
{ "415 (чис.)", "209л","419","201л (знам.)","213(знам.)" },{ "418", "201л", "415", "213", V },              { "209л",
"201л", V, V, V } },
  /* ЗІ-41 */       newstring[,] { { V, "213 (чис.)", "314", "418", V },        { "305л (знам.)", "314", "213", "418", V },
{ "418", "314", "205л", "305л", V },                  { V, "213", "314", "205л", V },                  { "305л", "314", V, V,
V } },
  /* ЗІ-42 */       newstring[,] { { V, "213 (знам.)", "418", "213", V },       { "418", "213", "314", "305л", V },
{ "205л (чис.)", "418", "314", "205л", V },              { V, "314", "205л", "314", V },                  { "314", "305л", V, V,
V } },
  /* ОК-41 */       newstring[,] { { "418", "410", "213", "230", V },          { V, V, "405", "410", V },                { V,
"410", "418", "304л", V },                    { V, "304л", "416", "418", "213 (чис.)" },        { "304л", "410", "304л", V,
V } },
  /* Ок-42 */       newstring[,] { { "213", "418", "410", "304л", V },         { V, V, "410", "405", V },                { V,
"230", "410", "418", "304 (чис.)"},               { V, "416", "418", "304л", "213 (знам.)"},        { "410", "304л", V, V,
V }},

};
      //Студенти
privatestaticstring[][] students = newstring[][] {
newstring[] { null},
  /* ЗІ-12 */   newstring[] {  "Бойчук", "Радзівон", "Василів", "Васильчук", "Дмитроца", "Григор'єв", "Хмарик",
"Мороз", "Самуляк", "Бенюх", "Свищ", "Фещук", "Гут", "Демко", "Марусяк", "Панас", "Матвіїв", },
  /* ЗІ-13 */   newstring[] { null},
  /* ОК-11 */   newstring[] { "Кривейко", "Костецький", "Головатий", "Ямелинець", "Телятник", "Павлусь", "Бутка",
"Богдан", "Сень", "Наконечний", "Годованець", "Ковальчук", "Яремко", "Нагорняк", "Кузьмів", "Тріска", "Назар",
"Яворський", "Лесков", "Кундис", },
  /* ТО-11 */   newstring[] { "Леонтенко", "Зоров", "Ісаєв", "Яхніцкий", "Клебан", "Коцун", "Бурий", "Кушнір",
"Ганусяк", "Наседкін", "Гаврилишин", "Дністрянський", "Маковецький", "Кірчицький", "Яциник", "Лукасевич",
"Вуйта", "Лаврик",  },
  /* ТО-12 */   newstring[] { "Медик", "Калінчук", "Тимкович", "Шандра", "Завидівський", "Кархут", "Медвідь",
"Байда", "Гринюх", "Лоджук", "Різник", "Шевчик", "Луцак", "Степаненко", "Ломачинський", "Барилюк",
"Козак", },
  /* ЗІ-11 */   newstring[] { "Наконечний", "Олексюк", "Бойко", "Сподарик", "Дробіт", "Головін", "Вельгас", "Цвик",
"Стрижак", "Зубрицький", "Дяченко", "Балицький", "Тим'як", "Козакевич", "Кравець", "Карпишин", "Черняк",
"Кисіль", "Мельник", },
  /* ОК-12 */   newstring[] { "Попович", "Стахів", "Форись", "Перетятко", "Кустов", "Каленик", "Сиротяк",
"Бесарабчик", "Труш", "Яремко", "Сичов", "Війтів", "Стронців", "Дудич", "Ротарь", "Русаков", "Савіцька",
"Оверко", "Костів", "Верхівський", },
  /* ТО-21 */   newstring[] { null},
  /* ЗІ-21 */   newstring[] { null},
  /* ЗІ-22 */   newstring[] { null},
  /* ОК-21 */   newstring[] { null},
  /* ОК-22 */   newstring[] { null},
  /* ТО-31 */   newstring[] { null},
  /* ТО-32 */   newstring[] { null},
  /* ЗІ-31 */   newstring[] { null},
  /* ЗІ-32 */   newstring[] { null},
  /* ОК-31 */   newstring[] { null},
  /* ОК-32 */   newstring[] { null},
  /* ТО-41 */   newstring[] { "Козак", "Дідик", "Жук", "Пундяк", "Сумик", "Кондратюк", "Куц", "Куциняк",
"Спеціляк", "Левків", "Стосик", "Недільський", "Плювак", "Либа", "Яворська", "Каняк", "Бадай", "Лазарєв",
"Іванців",},
  /* ТО-42 */   newstring[] { null},
  /* ЗІ-41 */   newstring[] { "Мартиновський", "Мельник", "Коваль", "Ісаєв", "Равлик", "Гложик", "Пронь", "Шевц",
"Гур'єв", "Мельничук", "Стахів", "Петриняк", "Хомік", "Куніцький", "Швед", "Демковський", },
  /* ЗІ-42 */   newstring[] { "Царів", "Боришко", "Горбатенко", "Підчеха", "Кондра", "Возна", "Добровольський",
"Вивка", "Лопотяк", "Козак", "Марко", "Стадник", "Гевяк", "Шильвінський", "Будзановський", "Хомік",  },
  /* ОК-41 */   newstring[] {  "Жигаль", "Гамаль", "Дутковський", "Валовий", "Мирка", "Буга", "Курач",
"Гнатишин", "Труш", "Романчак", "Ждан", "Муряш", "Фик", "Вантух", "Боднар", "Куціль", "Кунинець", "Куць",
"Козак", "Тютюнник", "Зінь", "Квасниця",},
```

```
  /* ОК-42 */   newstring[] {  "Боднарчук", "Будник", "Васько", "Волощук", "Гловацький", "Головчак", "Горак",
"Дмитрів", "Дрофяк", "Кокошко", "Комаринський", "Крят", "Лучковський", "Марцін", "Мацьків", "Медвідь",
"Олинський", "Онишко", "Садова", "Сторожук", "Харчишин", "Шпік", "Явдик" },
};
staticvoidMain(string[] args)
      {
botClient = new TelegramBotClient("1208961314:AAFpDJgabe-3odSc6Jxc3oNe6YY9psN8dfc") { Timeout =
TimeSpan.FromSeconds(10) };
varme = botClient.GetMeAsync().Result;
Console.WriteLine($"Botid {me.Id}. BotName {me.FirstName}");
botClient.OnMessage += Bot_OnMessage;
botClient.StartReceiving();
Console.ReadKey();
      }
privatestaticasyncvoidBot_OnMessage(objectsender, MessageEventArgs e)
      {
vartext = e?.Message?.Text;
if (text == null)
return;
Console.WriteLine($"receivedmessage '{text}' inchat '{e.Message.Chat.Id}'");
staticintDetermineDay()
         {
DayOfWeekday = DateTime.Now.DayOfWeek;
switch (day)
            {
caseDayOfWeek.Monday:
                  d = 0;
break;
caseDayOfWeek.Tuesday:
                  d = 1;
break;
caseDayOfWeek.Wednesday:
                  d = 2;
break;
caseDayOfWeek.Thursday:
                  d = 3;
break;
caseDayOfWeek.Friday:
                  d = 4;
break;
caseDayOfWeek.Saturday:
                  d = 5;
break;
caseDayOfWeek.Sunday:
                  d = 6;
break;

            }
return d;
         }
asyncvoidFindTeacher()
         {
DetermineDay();
for (int i = 1; i <teachers[0].Length; i++)
            {
if (text == teachers[0][0, i])
               {
if (DateTime.Now.TimeOfDay<newTimeSpan(8, 30, 00))
                  {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"По моїм даним, перша пара ще не наступила"
```

```
                    ).ConfigureAwait(false);
                }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(8, 30, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(10, 05,
00)))
                {
                    {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{teachers[i][d, 0]}"
                        ).ConfigureAwait(false);
                    }
                }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(10, 05, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(11, 50,
00)))
                {
                    {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{teachers[i][d, 1]}"
                        ).ConfigureAwait(false);
                    }
                }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(11, 50, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(13, 55,
00)))
                {
                    {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{teachers[i][d, 2]}"
                        ).ConfigureAwait(false);
                    }
                }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(14, 05, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(15, 40,
00)))
                {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{teachers[i][d, 3]}"
                    ).ConfigureAwait(false);
                }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(15, 40, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(17, 25,
00)))
                {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{teachers[i][d, 4]}"
                    ).ConfigureAwait(false);
                }
elseif (DateTime.Now.TimeOfDay>newTimeSpan(17, 25, 00))
                {
                    {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"По моїм даним, пари вже закінчились"
                        ).ConfigureAwait(false);
                    }
                }
                t = 0; //Скидання прапорця
            }
if ((t == 1) && (i == 18) && (text != "Знайти викладача"))
                {
```

```
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"На жаль, я не знаю такого викладача"
                    ).ConfigureAwait(false);
                    t = 0;
                }
            }
        }
asyncvoidFindGroup()
        {
DetermineDay();
for (int i = 1; i <groups[0].Length; i++)
            {
if (text.ToUpper() == groups[0][0, i])
                {
if (DateTime.Now.TimeOfDay<newTimeSpan(8, 30, 00))
                    {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"По моїм даним, перша пара ще не наступила"
                        ).ConfigureAwait(false);
                    }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(8, 30, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(10, 05,
00)))
                    {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{groups[i][d, 0]}"
                        ).ConfigureAwait(false);
                    }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(10, 05, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(11, 50,
00)))
                    {
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{groups[i][d, 1]}"
                                ).ConfigureAwait(false);
                        }
                    }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(11, 50, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(13, 55,
00)))
                    {
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{groups[i][d, 2]}"
                                ).ConfigureAwait(false);
                        }
                    }

elseif ((DateTime.Now.TimeOfDay>newTimeSpan(14, 05, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(15, 40,
00)))
                    {
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{groups[i][d, 3]}"
                                ).ConfigureAwait(false);
                        }
                    }
```

```
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(15, 40, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(17, 25,
00)))
                    {
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"{groups[i][d, 4]}"
                                ).ConfigureAwait(false);
                        }
                    }
elseif (DateTime.Now.TimeOfDay>newTimeSpan(17, 25, 00))
                    {
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat
text: $"По моїм даним, пари вже закінчились"
                                ).ConfigureAwait(false);
                        }
                    }
                    t = 0; //Скидання прапорця
                }
if ((t == 2) && (i == 25) && (text != "Знайти групу"))
                {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Назва групи введена некоректно. "
                        ).ConfigureAwait(false);
                    t = 0;
                }
            }
        }
asyncvoidFindStudent()
        {
DetermineDay();
for (int i = 1; i < (groups[0].Length - 1); i++)
            {
for (int j = 0; j <students[i].Length; j++)
                {
if (text == students[i][j])
                    {
if (DateTime.Now.TimeOfDay<newTimeSpan(8, 30, 00))
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"По моїм даним, перша пара ще не наступила"
                                ).ConfigureAwait(false);
                        }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(8, 30, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(10, 05,
00)))
                        {
if (groups[i][d, 0] == V)
                            {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"В даний момент у студента групи {groups[0][0, i]} за прізвищем {text} пара відсутня"
                                ).ConfigureAwait(false);
                            }
elseawaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Студент групи {groups[0][0, i]} за прізвищем {text} повинен знаходитись у аудиторії №{groups[i][d, 0]}"
                                ).ConfigureAwait(false);
                        }
```

```
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(10, 05, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(11, 50,
00)))
                    {
if (groups[i][d, 1] == V)
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"В даний момент у студента групи {groups[0][0, i]} за прізвищем {text} пара відсутня"
                            ).ConfigureAwait(false);
                        }
elseawaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Студент групи {groups[0][0, i]} за прізвищем {text} повинен знаходитись у аудиторії №{groups[i][d, 1]}"
                            ).ConfigureAwait(false);
                    }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(11, 50, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(13, 55,
00)))
                    {
if (groups[i][d, 2] == V)
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"В даний момент у студента групи {groups[0][0, i]} за прізвищем {text} пара відсутня"
                            ).ConfigureAwait(false);
                        }
elseawaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Студент групи {groups[0][0, i]} за прізвищем {text} повинен знаходитись у аудиторії №{groups[i][d, 2]}"
                            ).ConfigureAwait(false);
                    }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(14, 05, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(15, 40,
00)))
                    {
if (groups[i][d, 3] == V)
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"В даний момент у студента групи {groups[0][0, i]} за прізвищем {text} пара відсутня"
                            ).ConfigureAwait(false);
                        }
elseawaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Студент групи {groups[0][0, i]} за прізвищем {text} повинен знаходитись у аудиторії №{groups[i][d, 3]}"
                            ).ConfigureAwait(false);
                    }
elseif ((DateTime.Now.TimeOfDay>newTimeSpan(15, 40, 00)) && (DateTime.Now.TimeOfDay<newTimeSpan(17, 25,
00)))
                    {
if (groups[i][d, 4] == V)
                        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"В даний момент у студента групи {groups[0][0, i]} за прізвищем {text} пара відсутня"
                            ).ConfigureAwait(false);
                        }

elseawaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Студент групи {groups[0][0, i]} за прізвищем {text} повинен знаходитись у аудиторії №{groups[i][d, 4]}"
                            ).ConfigureAwait(false);
                    }
elseif (DateTime.Now.TimeOfDay>newTimeSpan(17, 25, 00))
```

```
                    {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"За моїми даними, пари вже закінчились"
                            ).ConfigureAwait(false);
                    }
                    t = 0; //Скидання прапорця
                }
if ((t == 3) && (i == 24) && (j == (students[i].Length - 1)) && (text != "Знайти студента") && (text != students[i][j]))
                {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"На жаль, я не знаю такого студента"
                        ).ConfigureAwait(false);
                    t = 0;
                }
            }
        }
    }
asyncvoidGiveSchedule()
    {
for (int i = 1; i <groups[0].Length; i++)
        {
if (text.ToUpper() == groups[0][0, i])
            {
                {
awaitbotClient.SendPhotoAsync(
chatId: e.Message.Chat,
photo: $"https://raw.githubusercontent.com/vyavdyk/dip/master/{i}.jpg");
                }
                t = 0; //Скидання прапорця
            }
if ((t == 4) && (i == (groups[0].Length - 1)) && (text != "Розклад занять"))
            {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Назва групи введена некоректно. "
                ).ConfigureAwait(false);
                t = 0;
            }
        }
    }
asyncvoidDetermineWeek()
    {
DateTimenow = DateTime.Today;
stringstartdate = "13.01";
DateTimepar = Convert.ToDateTime(startdate);
stringunpaired = "20.01";
DateTimeunpar = Convert.ToDateTime(unpaired);
stringfinishdate = "28.06";
DateTimefin = Convert.ToDateTime(finishdate);
do
        {
DateTime par2 = par.AddDays(7);
if (now>= par&&now< par2)
            {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Цей тиждень - парний ( у знаменнику )"
                ).ConfigureAwait(false);
break;
            }
```

```
DateTime unpar2 = unpar.AddDays(7);
if (now>= unpar&&now< unpar2)
                {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Цей тиждень - непарний ( у чисельнику )"
                  ).ConfigureAwait(false);
break;
                }
par = par.AddDays(14);
unpar = unpar.AddDays(14);
if (par>fin)
                {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Схоже, що цей тиждень - не навчальний"
                  ).ConfigureAwait(false);
break;
                }

            } while (true);
        }
if (text == "/start")
        {
varreplyKeyboardMarkup = newReplyKeyboardMarkup(
newKeyboardButton[][]
newKeyboardButton[] { "Знайти викладача", "Знайти групу" },
newKeyboardButton[] { "Знайти студента", "Розклад занять" },
newKeyboardButton[] { "Визначення навчального тижня"},
                          },
resizeKeyboard: true
                        );
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: "Доброго дня. Оберіть необхідну функцію",
replyMarkup: replyKeyboardMarkup
            );
        }
        // Обробка кнопок
elseif (text == "Розклад занять")
        {
            t = 4;
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Введіть назву групи через дефіс, наприклад: ОК-42, ТО-31"
            ).ConfigureAwait(false);
        }
elseif (text == "Визначення навчального тижня")
        {
DetermineWeek();
        }
DetermineDay();
if ((d == 5 || d == 6) && (text == "Знайти викладача" || text == "Знайти групу" || text == "Знайти студента"))
        {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"За моїми даними, сьогодні вихідний день. Дана функція не працює."
            ).ConfigureAwait(false);
        }
elseif (text == "Знайти викладача")
        {
            t = 1;
```

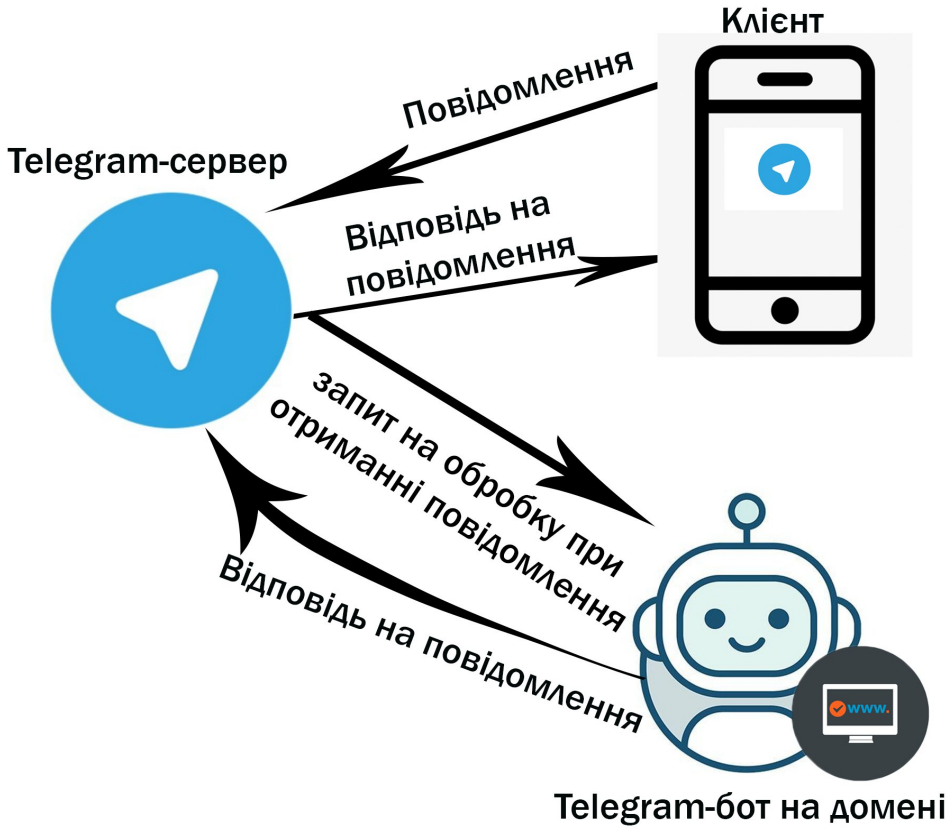
```
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Введіть прізвище викладача"
                ).ConfigureAwait(false);
            }
elseif (text == "Знайти групу")
            {
                t = 2;
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Введіть назву групи через дефіс, наприклад: ОК-42, ТО-31"
                ).ConfigureAwait(false);
            }
elseif (text == "Знайти студента")
            {
                t = 3;
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"Введіть прізвище студента"
                ).ConfigureAwait(false);

            }
elseif (t == 0 &&text != "Визначення навчального тижня" &&text != "/start" && (text == e?.Message?.Text))
            {
awaitbotClient.SendTextMessageAsync(
chatId: e.Message.Chat,
text: $"На жаль, я вас не розумію. Будь ласка, оберіть функцію."
                ).ConfigureAwait(false);
            }
            //Пошук викладача
elseif (t == 1 && (text == e?.Message?.Text))
            {
FindTeacher();
            }
            //Пошук групи
elseif (t == 2 && (text == e?.Message?.Text))
            {
FindGroup();
            }
            //Пошук студента
elseif (t == 3 && (text == e?.Message?.Text))
            {
FindStudent();
            }
            //Видача розкладу
elseif (t == 4 && (text == e?.Message?.Text))
            {
GiveSchedule();
            }
        }
    }
}
```

КОПІЇ ОБОВ’ЯЗКОВИХ КРЕСЛЕНЬ

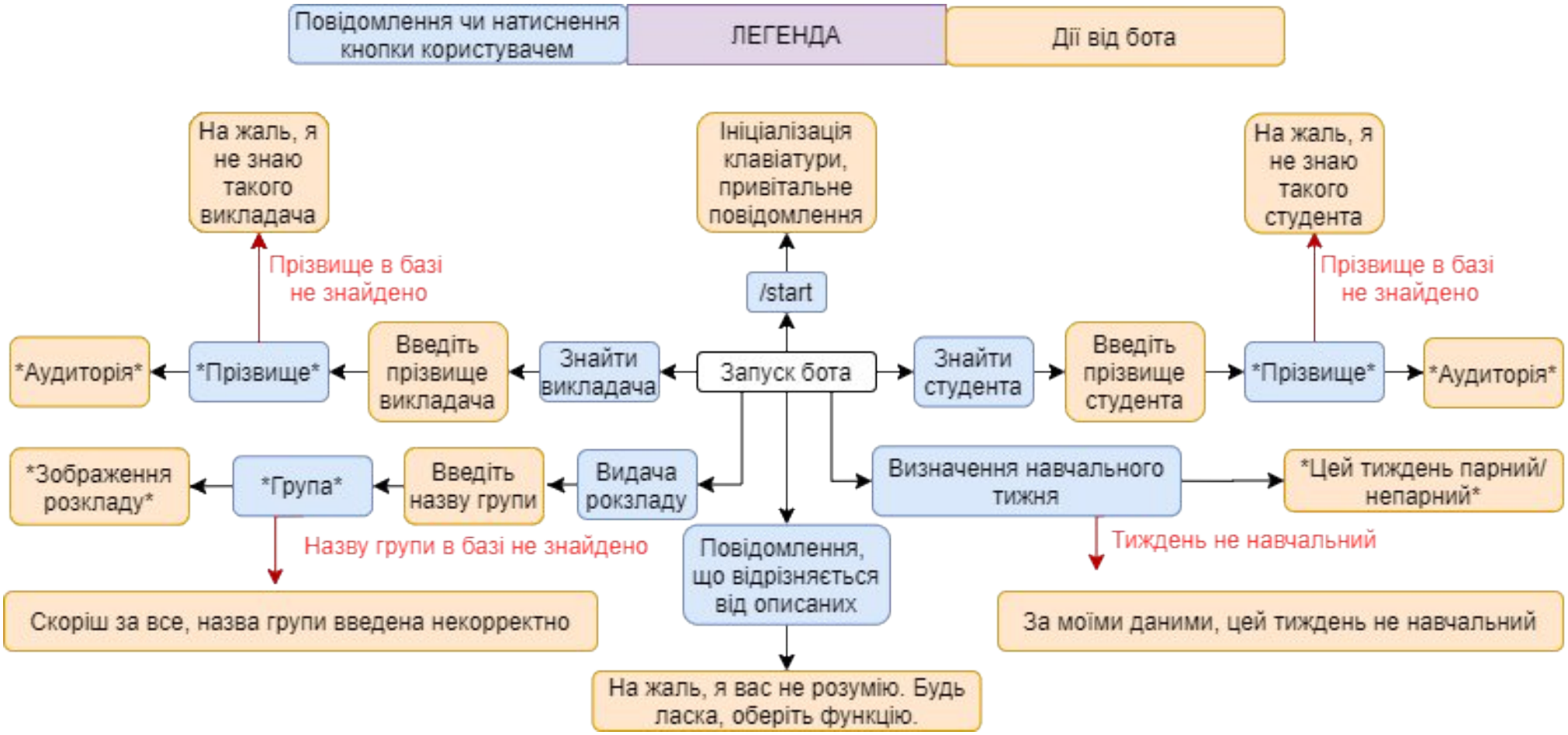
LongPolling

WebHooks

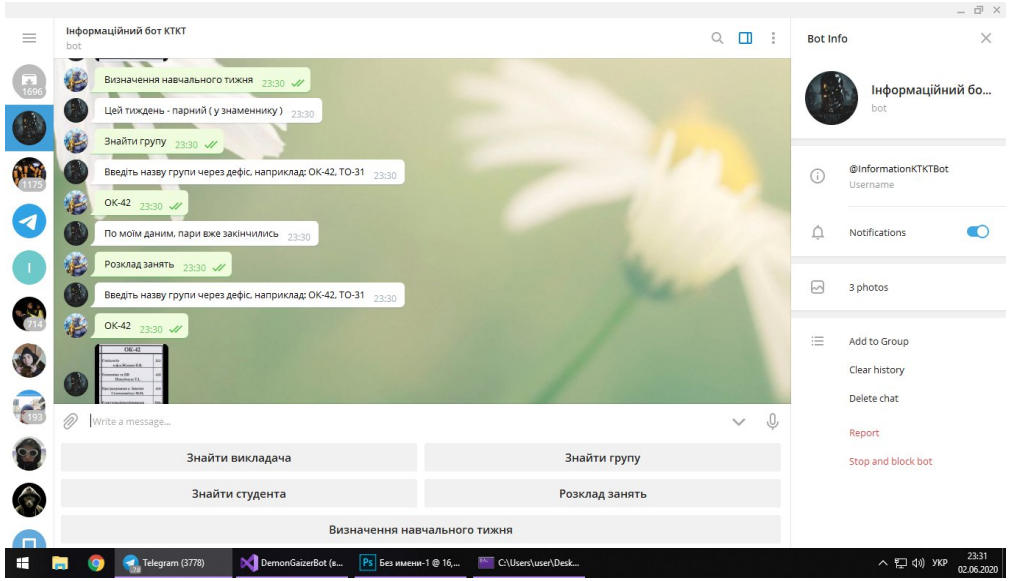


					Розробка інформаційного Telegram-бота ІТ коледжу Львівської політехніки
Зм.	Аркуш	№ докум.	Підпис	Дата	Демонстрація моделі роботи технологій LongPolling та Webhooks
Розроб.		Матвійчук В.В.			
Керівник		Селемонавічус А.А.			
Реценз.					
Н. контр.		КужійЛ.І.			
Затверд.					

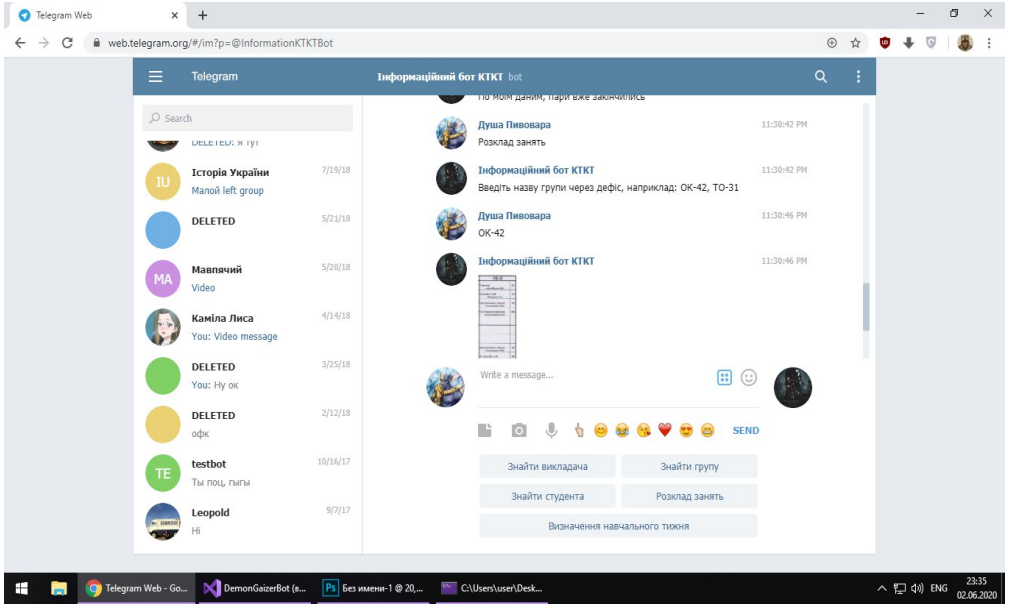
Літера	Маса	Маштаб
Аркуш	Аркушів	



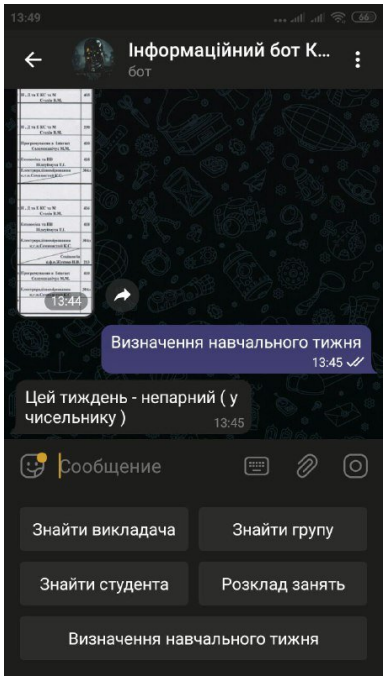
					Розробка інформаційного Telegram-бота ІТ коледжу Львівської політехніки
Зм.	Аркуш	№ докум.	Підпис	Дата	Схема взаємодії користувача з чат-ботом на клієнтському рівні
Розроб.		Матвійчук В.В.			Літера: — Маса: — Масштаб: —
Керівник		Селемонавічус А.А.			Аркуш — Аркушів —
Реценз.					
Н. контр.					
Затверд.					



Діалогове вікно між ботом та користувачем у Telegram Desktop



Діалогове вікно між ботом та користувачем у Web.Telegram.Org



Діалогове вікно між ботом та користувачем у мобільній версії Telegram

					Розробка інформаційного Telegram-бота ІТ коледжу Львівської політехніки
					Клієнтський інтерфейс взаємодії з чат-ботом у різних представленнях
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*	
Розроб.		Матвійчук В.В.			
Керівник		Селемонавічус А.А.			
Реценз.					
Н. контр.					
Затверд.					

Літера	*Маса*	*Масштаб*
Аркуш	*Аркушів*	

Кошторис витрат на розробку та впровадження програмного рішення

Найменування елементів витрат	Сума витрат, грн.
Витрати на оплату праці	3 446,83
Нарахування на зарплату	412,59
Витрати на куповані вироби	572
Накладні витрати	1 034,05
Інші витрати	607,27
Витрати на налагодження та дослідну експлуатацію	1 620
Всього:	7 752,74

					Розробка інформаційного Telegram-бота ІТ коледжу Львівської політехніки			
Зм.	Аркуш	№ докум.	Підпис	Дата	Витрати на розробку та впровадження проектного рішення	Літера	Маса	Масштаб
Розроб.		Матвійчук В.В.						
Керівник		Селемонавічус А.А.				Аркуш	Аркушів	
Реценз.								
Н. контр.		Кужій Л.І.						
Затверд.								