

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

з дисципліни «Основи WEB - технологій»

Тема: «Створення telegram-боту з меню та запитом до ChatGPT»

Перевірів:

Доц. Голубєв Л. П.

Виконав:

студент групи ІМ-22

Балахон Михайло

Варіант 11

Завдання1

Створити telegram-бот з меню та задеплоїти його на будь-якому сервісі (PythonAnywhere, Glitch, AWS або ін.)

Структура меню:

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

Хід роботи:

1. Підготовка проекту

1.1. Створення структури проекту

```
mkdir lab3
```

```
cd lab3
```

```
npm init -y
```

1.2. Встановлення необхідних пакетів

```
npm install node-telegram-bot-api openai dotenv
```

```
npm install --save-dev nodemon
```

Встановлені пакети:

- node-telegram-bot-api - для роботи з Telegram Bot API
- openai - для інтеграції з ChatGPT API
- dotenv - для роботи з змінними оточення
- nodemon - для автоматичного перезапуску під час розробки

2. Налаштування конфігурації

2.1. Файл .env

TELEGRAM_TOKEN=your_telegram_bot_token_here

OPENAI_API_KEY=your_openai_api_key_here

2.2. Отримання токенів

1. Telegram Bot Token:

- Створення бота через @BotFather
- Отримання токена за допомогою команд /newbot

2. OpenAI API Key:

- Реєстрація на platform.openai.com
- Створення API ключа в розділі API Keys

3. Реалізація основної логіки бота

3.1. Ініціалізація бота та конфігурація

```
const TelegramBot = require('node-telegram-bot-api');
```





```
const OpenAI = require('openai');
```

```
require('dotenv').config();
```

```
const bot = new TelegramBot(process.env.TELEGRAM_TOKEN, { polling: true });
```

```
const openai = new OpenAI({  
  apiKey: process.env.OPENAI_API_KEY,  
});
```

3.2. Створення меню клавіатури

```
const mainMenuKeyboard = {  
  reply_markup: {  
    keyboard: [  
      [' Студент', ' ІТ-технології'],  
      [' Контакти', ' Prompt ChatGPT']  
    ],  
    resize_keyboard: true,  
    one_time_keyboard: false  
  }  
};
```





3.3. Система станів користувачів

Реалізовано систему відстеження стану користувачів для управління режимом ChatGPT:

```
const userStates = {};
```

4. Реалізація обробників меню

4.1. Інформація про студента

```
async function handleStudentInfo(chatId) {  
  const studentInfo = `  
 *Інформація про студента*  
  
 **Прізвище:** Балахон М.О.  
  
 **Група:** ІМ-22  
  
 **Факультет:** Інформатики та обчислювальної техніки
```

 ****Спеціальність:** Інженерія програмного забезпечення**

 ****Курс:** 4**

`;
`;

```
bot.sendMessage(chatId, studentInfo, {  
  parse_mode: 'Markdown',  
  ...backKeyboard  
});  
}
```

4.2. IT-технології

Відображення списку використовуваних технологій:


- Front-end: HTML5, CSS3, JavaScript, React.js, Vue.js
- Back-end: Node.js, Express.js, Python, FastAPI
- Інструменти: Git, GitHub, Docker, AWS

4.3. Контактна інформація

```
async function handleContacts(chatId) {
```

```
  const contactInfo = `
```

 ***Контактна інформація***

 ****Телефон:** +380 50-555-55-55**

 ****Email:** student@example.com**

 ****Telegram:** @mibal_ua**

 ****LinkedIn:** linkedin.com/in/mibal-ua**

 ****GitHub:** github.com/mibal-ua**

`;
`;

```
}
```

5. Інтеграція з ChatGPT

5.1. Обробка ChatGPT запитів

```
async function handleChatGPTMessage(chatId, message) {
  try {
    bot.sendChatAction(chatId, 'typing');

    const completion = await openai.chat.completions.create({
      model: "gpt-3.5-turbo",
      messages: [
        {
          role: "system",
          content: "Ти корисний асистент, який відповідає українською мовою."
        },
        {
          role: "user",
          content: message
        }
      ],
      max_tokens: 1000,
      temperature: 0.7
    });

    const response = completion.choices[0].message.content;
    bot.sendMessage(chatId, `🤖 *ChatGPT відповідає:* \n\n${response}`, {
      parse_mode: 'Markdown',
      ...backKeyboard
    });
  } catch (error) {
```

```
// Обробка помилок API
}
}
```

5.2. Система станів для ChatGPT

- Активація режиму ChatGPT при виборі відповідного пункту меню
- Відстеження стану користувача для правильної обробки повідомлень
- Можливість повернення до головного меню

6. Обробка помилок та безпека

6.1. Обробка помилок OpenAI API

```
if (error.code === 'insufficient_quota') {
  errorMessage = '❌ Вичерпано ліміт запитів до ChatGPT API.';
} else if (error.code === 'invalid_api_key') {
  errorMessage = '❌ Невірний API ключ OpenAI.';
}
```


6.2. Graceful shutdown

Реалізовано коректне завершення роботи бота:

```
process.on('SIGINT', () => {
  console.log('Shutting down bot...');
  bot.stopPolling();
  process.exit(0);
});
```

7. Функціональні можливості бота

7.1. Основні команди

- /start - запуск бота та відображення головного меню
- Навігація по меню за допомогою кнопок клавіатури
- Кнопка " Назад" для повернення в головне меню

7.2. Режими роботи

1. Навігаційний режим - вибір пунктів меню
2. Інформаційний режим - відображення статичної інформації
3. ChatGPT режим - інтерактивне спілкування з AI

7.3. Користувацький інтерфейс

- Емоїї іконки для покращення візуального сприйняття
- Markdown форматування для структурованого відображення
- Адаптивна клавіатура з кнопками
- Індикатор набору тексту під час обробки ChatGPT запитів

8. Тестування функціональності

8.1. Тестування меню

1. Перевірка відображення головного меню після /start
2. Тестування всіх пунктів меню
3. Перевірка роботи кнопки "Назад"

8.2. Тестування ChatGPT інтеграції

1. Активація режиму ChatGPT
2. Відправка тестових запитів

3. Перевірка обробки помилок API
4. Тестування повернення в головне меню

8.3. Обробка помилок

1. Неправильні токени API
2. Відсутність мережевого з'єднання
3. Перевищення лімітів API

Структура проекту

lab3/

```
|— bot.js          # Основний файл бота
|— package.json    # Конфігурація проекту
|— .env.example    # Приклад змінних середовища
|— .env            # Змінні середовища (не в git)
|— .gitignore      # Файли для ігнорування
└— README.md       # Документація проекту
```

Результати роботи

Створено повнофункціональний Telegram бот з наступними можливостями:

1. Структуроване меню з чотирма основними розділами
2. Інформаційні блоки про студента, технології та контакти
3. ChatGPT інтеграція для інтерактивного спілкування з AI
4. Користувачський інтерфейс з емої та markdown форматуванням
5. Система станів для управління режимами роботи
6. Обробка помилок та graceful shutdown

7. Документація та інструкції для деплою

Скріншоти роботи бота:

- Головне меню з кнопками навігації
- Відображення інформації про студента
- Список ІТ-технологій
- Контактна інформація
- Режим роботи з ChatGPT
- Відповіді від AI асистента

Можливості деплою

Бот готовий для деплою на різних платформах:

- Heroku - через git deployment
- Railway - підключення GitHub репозиторію
- Glitch - імпорт проекту
- DigitalOcean App Platform - container deployment
- AWS/Google Cloud - через serverless functions

Висновки

В ході виконання лабораторної роботи:

- Вивчено принципи роботи з Telegram Bot API
- Реалізовано структуроване меню з навігацією
- Впроваджено інтеграцію з OpenAI ChatGPT API
- Створено систему станів для управління діалогом
- Отримано практичні навички розробки чат-ботів

- Вивчено методи обробки помилок та graceful shutdown

Telegram боти є потужним інструментом для створення інтерактивних додатків, що можуть інтегруватися з різними сервісами та API. Інтеграція з ChatGPT відкриває широкі можливості для створення розумних асистентів та автоматизації спілкування з користувачами.

Проект демонструє практичне застосування сучасних веб-технологій для створення корисних інтерактивних сервісів.

Скріншоти:



test bot
bot



Будь ласка, виберіть команду з меню 15:04



IT-технології 15:04 ✓✓



IT-технології



Front-end:

- HTML5, CSS3, JavaScript
- React.js, Vue.js
- Bootstrap, Tailwind CSS



Back-end:

- Node.js, Express.js
- Python, FastAPI
- MongoDB, PostgreSQL



Інструменти:

- Git, GitHub
- Docker, AWS
- VS Code, WebStorm

15:04



Назад 15:04 ✓✓

Виберіть відповідну команду 15:04



Студент 15:04 ✓✓



Інформація про студента



Прізвище: Балахон М.О.



Група: ІМ-22



Факультет: Інформатики та обчислювальної техніки



Спеціальність: Інженерія програмного забезпечення



Курс: 4




15:04



Write a message...



Назад

Service ID: `srv-d3dta2je5dus73bvscr0`  mibal-ua / 4-web-labs  main<https://four-web-lab3.onrender.com> 

📘 Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.



[Upgrade now](#)

September 30, 2025 at 3:03 PM

✓ Live

[5862bb3](#) Implement lab3: Telegram Bot with ChatGPT Integration - Create Telegram bot with structured menu system - Implement student...

All logs ▾

 Search Live tail ▾

GMT+2



...

```
Sep 30 03:03:22 PM ⓘ Run 'npm audit' for details.
Sep 30 03:03:25 PM ⓘ ==> Uploading build...
Sep 30 03:03:36 PM ⓘ ==> Uploaded in 8.7s. Compression took 1.7s
Sep 30 03:03:36 PM ⓘ ==> Build successful 🎉
Sep 30 03:03:40 PM ⓘ ==> Deploying...
Sep 30 03:03:53 PM ⓘ ==> Running 'npm start'
Sep 30 03:03:54 PM ⓘ
Sep 30 03:03:54 PM ⓘ > lab3-telegram-bot@1.0.0 start
Sep 30 03:03:54 PM ⓘ > node bot.js
Sep 30 03:03:54 PM ⓘ
Sep 30 03:03:59 PM ⓘ 🎉 Telegram bot started successfully!
Sep 30 03:03:59 PM ⓘ Make sure to set TELEGRAM_TOKEN and OPENAI_API_KEY in environment variables
Sep 30 03:03:59 PM ⓘ 🌐 Health check server running on port 10000
Sep 30 03:04:01 PM ⓘ ==> Your service is live 🎉
Sep 30 03:04:02 PM ⓘ ==>
Sep 30 03:04:02 PM ⓘ ==> //////////////////////////////////////
Sep 30 03:04:02 PM ⓘ ==>
Sep 30 03:04:02 PM ⓘ ==> Available at your primary URL https://four-web-lab3.onrender.com
Sep 30 03:04:02 PM ⓘ ==>
Sep 30 03:04:02 PM ⓘ ==> //////////////////////////////////////
Sep 30 03:04:06 PM ❌ error: [polling_error] {"code":"ETELEGRAM","message":"ETELEGRAM: 409 Conflict: terminated by other getUpdates request; make sure that only one bot instance is running"}
Sep 30 03:04:12 PM ❌ error: [polling_error] {"code":"ETELEGRAM","message":"ETELEGRAM: 409 Conflict: terminated by other getUpdates request; make sure that only one bot instance is running"}
Sep 30 03:04:19 PM ❌ error: [polling_error] {"code":"ETELEGRAM","message":"ETELEGRAM: 409 Conflict: terminated by other getUpdates request; make sure that only one bot instance is running"}
Sep 30 03:04:26 PM ❌ error: [polling_error] {"code":"ETELEGRAM","message":"ETELEGRAM: 409 Conflict: terminated by other getUpdates request; make sure that only one bot instance is running"}
Sep 30 03:09:09 PM ⓘ ==> Detected service running on port 10000
Sep 30 03:09:10 PM ⓘ ==> Docs on specifying a port: https://render.com/docs/web-services#port-binding
```

Need better ways to work with logs? Try the  Render CLI,  Render MCP Server, or set up a log stream integration 