#### ****Техническое задание (ТЗ)****

**Название проекта**: Telegram-бот «Расписание АИТ»  
**Исполнитель**: Артур Хандожко, студент 3 курса  
**Цель**: Разработка Telegram-бота для предоставления расписания занятий студентов АИТ в удобной форме.

**1. Общие положения**:

* **Наименование программы**: Telegram-бот «Расписание АИТ».
* **Область применения**: Использование студентами и преподавателями для быстрого доступа к расписанию занятий.
* **Основание для разработки**: Учебное задание, предоставленное в рамках практической работы.

**2. Функциональные требования**:

* **Основные функции**:
  + Вывод расписания занятий на конкретный день недели.
  + Возможность запроса расписания для конкретной группы.
  + Вывод расписания на текущую или следующую неделю.
  + Уведомление о ближайших изменениях в расписании.
* **Интерфейс**:
  + Работа через Telegram (чат-бот).
  + Простая и интуитивно понятная структура команд:
    - /start — приветственное сообщение.
    - /расписание — общее расписание на текущую неделю.
    - /группа N — расписание группы N на текущую дату.
    - /завтра — расписание на следующий день.

**3. Нефункциональные требования**:

* **Надежность**: Бот должен корректно обрабатывать некорректные запросы (например, несуществующие группы или неправильный формат даты).
* **Производительность**: Ответ на запросы пользователя не более 1-2 секунд.
* **Совместимость**: Совместимость с последними версиями Telegram API.

**4. Средства разработки**:

* Язык программирования: Python.
* Используемые библиотеки: python-telegram-bot, requests.
* Серверная часть (реализация через Leadteh.ru API).

#### ****Шаг 2: Диаграмма UML взаимодействия клиента с сервером****

**Описание диаграммы:**

* **Акторы**:
  + Пользователь (студент, преподаватель).
  + Telegram API (посредник).
  + Сервер https://aitanapa.ru/ (хранение данных расписания).
* **Процесс взаимодействия**:
  1. Пользователь отправляет команду через Telegram-бота.
  2. Бот обращается к серверу Telegram API.
  3. Telegram API перенаправляет запрос серверу https://aitanapa.ru для получения расписания.
  4. https://aitanapa.ru возвращает данные о расписании.
  5. Telegram-бот формирует ответ и отправляет его пользователю через Telegram API.

**Диаграмма UML (в формате текста):**

rust

Пользователь -> Telegram-бот: Отправляет запрос (/расписание)

Telegram-бот -> Telegram API: Передаёт запрос

Telegram API -> https://aitanapa.ru: Запрашивает данные расписания

https://aitanapa.ru -> Telegram API: Возвращает расписание

Telegram API -> Telegram-бот: Передаёт расписание

Telegram-бот -> Пользователь: Возвращает сообщение с расписанием

#### ****Шаг 3: Алгоритм работы****

1. **Запуск**:
   * Пользователь вводит команду /start — бот отправляет приветственное сообщение.
2. **Получение запроса**:
   * Пользователь выбирает команду (например, /расписание или /группа N).
   * Бот проверяет корректность команды.
3. **Обработка запроса**:
   * Если запрос корректен, бот формирует запрос к серверу https://aitanapa.ru для получения данных расписания.
   * Если запрос некорректен, бот отправляет сообщение об ошибке.
4. **Ответ пользователю**:
   * Сервер возвращает данные, бот форматирует их и отправляет пользователю.
5. **Завершение**:
   * Бот ожидает следующего запроса.

#### ****Шаг 4: Код программы****

**Пример на Python:**

python

from telegram import Update, Botfrom telegram.ext import Updater, CommandHandler, CallbackContextimport requests

API\_TOKEN = "ВАШ\_ТОКЕН\_БОТА"

LEADTEH\_API\_URL = "https://aitanapa.ru/api/schedule"

def start(update: Update, context: CallbackContext):

update.message.reply\_text("Привет! Я бот расписания АИТ. Напиши /расписание или /группа <номер>.")

def get\_schedule(update: Update, context: CallbackContext):

try:

group = context.args[0] if context.args else None

if not group:

update.message.reply\_text("Пожалуйста, укажите номер группы. Например: /группа 1/22")

return

response = requests.get(f"{LEADTEH\_API\_URL}?group={group}")

if response.status\_code == 200:

schedule = response.json()

reply = f"Расписание для группы {group}:\n"

for day, lessons in schedule.items():

reply += f"{day}:\n" + "\n".join(lessons) + "\n\n"

update.message.reply\_text(reply)

else:

update.message.reply\_text("Не удалось получить расписание. Попробуйте позже.")

except Exception as e:

update.message.reply\_text(f"Ошибка: {e}")

def main():

updater = Updater(API\_TOKEN)

dp = updater.dispatcher

dp.add\_handler(CommandHandler("start", start))

dp.add\_handler(CommandHandler("группа", get\_schedule))

updater.start\_polling()

updater.idle()

if \_\_name\_\_ == "\_\_main\_\_":

main()

#### ****Шаг 5: Документация****

**1. Настройка окружения**:

* Установите Python.
* Установите библиотеки: pip install python-telegram-bot requests.

**2. Запуск бота**:

* Вставьте токен своего Telegram-бота в переменную API\_TOKEN.
* Запустите скрипт.

**3. Проверка работы**:

* Откройте Telegram, найдите своего бота и введите команду /start для начала работы.
* Используйте команды /расписание или /группа <номер>.

**4. Примечания**:

* Убедитесь, что сервер https://aitanapa.ru доступен и правильно возвращает данные.
* В случае ошибок в API проверьте логика запросов.