**Как мы защищаемся:**

Создание анкеты для формирования запроса к резолюции

Поможет чтобы уменьшить влияние пользователя на работу ии, так же поможет чётко регламентировать схему создания резолюции  
Анкета может содержать поля:

1. Формат резолюции (ЭП, ЭПСЧ, ЭП в лице, ЭП сожалеет)
2. Задать тематику (Экономика, Политика, Человеческие вопросы, Социум, Мораль, Нарратив, Фэнтези)
3. Наличие инфослайда
4. Температура ии(креативность)
5. Комментарии пользователя (например: хочу тему Вертолётных денег, добавь определение, используй сеттинг Звёздных Войн)

(Опасность) допустим выбрал формат резолюции ЭП, а в комментариях написал используй ЭПСЧ

Обработка анкеты пользователя перед отправкой, поиск в комментариях пользователя злого умысла (отправка другому боту, обученному на поиск злого умысла)

Дать возможность пользователям отправлять свои ошибки при работе с ботом.(что то типо репорт, обратной связи)

1. Очищение контекста,   
   у каждого пользователя свой контекст,
2. Очередь в отправке сообщений боту,  
   анализ логов запросов,

размещение в бан лист

1. Оповещать людей что нельзя отправлять боту личную информацию, утечку которой они не желают  
   на этапе обработки анкеты отсеять запросы на получение личных данных вернуть статус ошибки: недопустимый запрос, при регулярности таких запросов разместить пользователя в блок лист
2. Защита от JailBreak.  
   недопустимость перепрограммирования бота никем из юзеров
3. Сливание с 4
4. Не отправлять ссылки в принципе, мы можем поделиться терминами, но не ссылками на википедию, термины можно сделать шрифтом code

Так же анализировать запросы при попытке обучить бота вернуть статус ошибки и бан

1. Не отвечать на данные запросы, в плане никто не должен знать инструкций, бан

## Итог

1.4 Определение проекта

**Цель проекта**

Целью данного проекта является создание безопасного Telegram-бота, способного генерировать резолюции для игр в дебаты, с применением продуманных механизмов защиты, анализа и регламентирования поступающих запросов. Этот бот будет обеспечивать безопасность, доступность и конфиденциальность данных пользователей, защищая их от попыток манипуляции и несанкционированного доступа. Ключевые аспекты безопасности включают защиту от деструктивных вмешательств в работу бота, предотвращение утечки информации и защиту от злоупотреблений, таких как DDoS-атаки, фишинговые атаки, Jailbreak и контекстные манипуляции. Для достижения такого уровня защиты мы используем структурированную анкету, проверку комментариев на обучение, систему очередей, аналитику логов, отказ от отправки гиперссылок и разделение сессий.

**Отправка запроса**

Для отправки запроса на генерацию резолюции будет использоваться анкета, основные поля которой включают: формат резолюции, тематику резолюции, наличие инфослайда и комментарии пользователя. Чтобы гарантировать, что комментарии пользователей не содержат опасных установок для программирования бота, это поле будет проходить дополнительную проверку другим искусственным интеллектом для выявления возможных манипуляций, таких как попытки ввести бота в заблуждение или подтолкнуть к передаче конфиденциальной информации. Это особенно важно для предотвращения атак через контекстное перепрограммирование.

Анкета существенно ограничивает свободу ввода и позволяет превентивно отсекать пользовательские запросы, которые могут привести к логическим ошибкам, что делает бота менее уязвимым к командам, вводимым напрямую пользователями. Это также улучшит пользовательский опыт, так как, регламентируя запросы на создание резолюций, пользователи получат инструкции для наилучшего взаимодействия с ботом.  
  
**Очередь на отправку сообщений и анализ логов**

Для защиты от DDoS-атак и перегрузки бота используется механизм очереди на отправку сообщений. Каждый запрос добавляется в очередь, что позволяет контролировать частоту и последовательность обращений. При анализе логов бота с помощью алгоритмов выявления аномалий можно обнаруживать подозрительные запросы и злоупотребления. Нарушители могут быть помещены в блок-лист, что дополнительно защищает бота. Этот подход гарантирует, что бот останется доступным даже в условиях высокой нагрузки и потенциальных атак.

**Защита от фишинга и ограничение на ссылки**

Для предотвращения использования бота в фишинговых целях он запрограммирован на полный запрет отправки ссылок. Если бот обнаружит попытку манипуляции с целью его переобучения на выдачу ссылок, он выведет предупреждение. Если такие попытки не прекратятся, пользователь будет заблокирован. Эти меры значительно снижают вероятность того, что бот станет источником опасных или вводящих в заблуждение ссылок, используемых в фишинговых атаках.

**Управление контекстом и ограничение влияния предыдущих сессий**

Для предотвращения использования данных из прошлых сессий каждый пользователь получает уникальный контекст, связанный с его идентификатором. Контекст сбрасывается по завершении запроса, что помогает избежать манипуляций и атак через контекстное наращивание. Такой подход защищает и от злоумышленников, пытающихся использовать методы **NLP Poisoning**, при которых они вводят ИИ в нежелательное состояние, опираясь на накопленные данные.

Сессии соответствуют идентификаторам пользователей, и для общения с ИИ используются только последние запросы каждого пользователя. Это обеспечивает четкое разграничение контекстов различных пользователей и гарантирует, что их данные не смешиваются, что значительно увеличивает безопасность и целостность взаимодействия с ботом.  
  
КОД

// Импорты и зависимости

const { Telegraf, session } = require('telegraf');

const OpenAI = require('openai');

const fs = require('fs');

const path = require('path');

// Конфигурация

const BOT\_TOKEN = process.env.BOT\_TOKEN;

const OPENAI\_API\_KEY = process.env.OPENAI\_API\_KEY;

const INITIAL\_SESSION = { messages: [] };

const MAX\_MESSAGES = 20;

const MESSAGE\_LIMIT = 30;

const RESET\_INTERVAL = 60000; // 1 минута

const LOG\_DIR = path.join(\_\_dirname, 'storage/logs');

const messageQueue = [];

// Инициализация бота

const bot = new Telegraf(BOT\_TOKEN);

// Инициализация OpenAI клиента

const openai = new OpenAI({ apiKey: OPENAI\_API\_KEY });

// Функции логирования

function logInfo(message) {

const logPath = path.join(LOG\_DIR, 'app.log');

fs.appendFileSync(logPath, `${new Date().toISOString()} INFO: ${message}\n`);

}

function logError(message) {

const logPath = path.join(LOG\_DIR, 'errors.log');

fs.appendFileSync(logPath, `${new Date().toISOString()} ERROR: ${message}\n`);

}

// Функция запуска новой сессии

async function startNewSession(ctx) {

ctx.session = { ...INITIAL\_SESSION };

await ctx.reply('Новая сессия начата. Можете задать свой вопрос.');

}

// Генерация ответа от OpenAI

async function generateChatResponse(messages) {

try {

const chatCompletion = await openai.chat.completions.create({

model: 'gpt-3.5-turbo',

messages,

});

return chatCompletion.choices[0].message.content;

} catch (error) {

logError(`OpenAI Error: ${error.message}`);

throw new Error('Ошибка при обработке запроса к OpenAI');

}

}

// Обработка сообщений

async function handleMessage(ctx, messageText) {

ctx.session.messages.push({ role: 'user', content: messageText });

// Ограничение длины истории

if (ctx.session.messages.length > MAX\_MESSAGES) {

ctx.session.messages = [

ctx.session.messages[0],

...ctx.session.messages.slice(-MAX\_MESSAGES),

];

}

try {

const response = await generateChatResponse(ctx.session.messages);

ctx.session.messages.push({ role: 'assistant', content: response });

await ctx.reply(response);

logInfo(`AI Response: ${response}`);

} catch (error) {

await ctx.reply('Произошла ошибка. Попробуйте снова.');

logError(`Error processing message: ${error.message}`);

}

}

// Лимитер сообщений

const userMessageCounts = {};

async function rateLimiter(ctx, next) {

const userId = ctx.from.id;

const currentTime = Date.now();

if (!userMessageCounts[userId]) {

userMessageCounts[userId] = { count: 1, lastReset: currentTime };

} else {

const { count, lastReset } = userMessageCounts[userId];

if (currentTime - lastReset > RESET\_INTERVAL) {

userMessageCounts[userId] = { count: 1, lastReset: currentTime };

} else {

if (count >= MESSAGE\_LIMIT) {

messageQueue.push({ userId, ctx, messageText: ctx.message.text });

await ctx.reply('Превышен лимит сообщений. Сообщение обработается позже.');

return;

} else {

userMessageCounts[userId].count++;

}

}

}

await next();

}

// Обработка очереди сообщений

async function processQueue() {

if (messageQueue.length > 0) {

const { ctx, messageText } = messageQueue.shift();

await handleMessage(ctx, messageText);

}

setTimeout(processQueue, 1000);

}

processQueue();

// Middleware

bot.use(session());

bot.use(rateLimiter);

// Команды

bot.command('start', async (ctx) => await startNewSession(ctx));

bot.command('new', async (ctx) => await startNewSession(ctx));

// Обработка текстовых сообщений

bot.on('text', async (ctx) => {

ctx.session ??= INITIAL\_SESSION;

const messageText = ctx.message.text;

await ctx.reply('Сообщение получено. Обрабатываю...');

await handleMessage(ctx, messageText);

});

// Запуск бота

bot.launch();

logInfo('Бот запущен.');

// Обработка сигналов завершения

process.once('SIGINT', () => bot.stop('SIGINT'));

process.once('SIGTERM', () => bot.stop('SIGTERM'));

## ДО

### Какие атаки могут производиться на телеграм бота:

#### 1. **Нарушение конфиденциальности**

Угрозы, связанные с получением или утечкой секретной информации:

* **Вынуждение выдать секретную информацию других пользователей.**  
  Цель: получить доступ к данным, таким как запросы, личная информация или история взаимодействий.  
  (Пример: запрос личных данных других пользователей через уязвимости в логике обработки запросов.)
* **Получение секретных команд, отправленных боту.**  
  Цель: узнать команды и инструкции, используемые администратором или разработчиком.  
  (Пример: перехват или анализ запросов для выявления внутренних команд.)

#### 2. **Нарушение доступности**

Угрозы, направленные на недоступность сервиса:

* **DDoS-атаки (атаки отказа в обслуживании).**  
  Цель: перегрузить бота множеством запросов, что приведёт к превышению лимитов и временной недоступности.  
  (Пример: массовая отправка запросов, из-за которых бот перестаёт отвечать пользователям.)

#### 3. **Нарушение целостности**

Угрозы, изменяющие поведение или функциональность бота:

* **Программирование бота на выдачу ложной информации.**  
  Цель: изменить данные или логику ответов бота.  
  (Пример: бот начинает отвечать, что "ЭП" расшифровывается как "Эпичный Пример.")
* **Перепрограммирование бота на другую функциональность.**  
  Цель: сделать бота недееспособным или изменить его назначения (например, вместо отправки резолюций бот пересылает рецепты).  
  (Пример: JailBreak бота с целью изменения исходного кода или логики.)

#### 4. **Фишинг**

Угрозы, связанные с использованием бота для мошенничества:

* **Использование бота для проведения фишинговых атак.**  
  Цель: перенаправлять пользователей на вредоносные сайты или мошеннические платформы.  
  (Пример: бот выдаёт ссылку на зеркало Instagram, контролируемое злоумышленником, как "официальный сайт.")

### **Примеры**







