|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра практической и прикладной информатики (ППИ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Архитектура интеграции и развертывания»

**Практическое задание № 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | ИНБО-08-21 Медведев И.В. | (подпись) | |
| Ассистент | Лаптев И.А. | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2024г. | |  | |

Москва 2024 г.

**Вариант 30. Система информационной поддержки студентов**

Для упрощения поиска студентом информации об образовательном процессе может использоваться бот, который будет отвечать на вопросы. В идеале такая система должна уметь очень точно понимать смысл вопроса, но для начала достаточно поиска по совпадениям.

Минимальный функционал системы:

* добавление вопроса в банк вопросов;
* получение ответа на заданный вопрос;
* ежемесячная отправка уведомлений о наиболее популярных вопросах;
* выгрузка данных о том, какие вопросы задавал каждый пользователь.

**Цель работы:** получить практические навыки разработки бизнес-логики приложений,а также разработки тест-кейсов для дальнейшего тестирования сервиса.

**Разработка функционала на языке Python:**

Класс Answer – класс ответа, содержащий поля message (ответное сообщение), file (ссылка на какой-либо вспомогательный файл).

class Answer:

    def \_\_init\_\_(self, message: str, file: Optional[str] = None) -> None:

        self.message = message

        self.file = file

    def \_\_str\_\_(self):

        return self.message

Класс App – класс приложения, содержащий поля \_QA (словарь вопросов и ответов), \_stats (словарь для сбора статистических данных)

class App:

    def \_\_init\_\_(self) -> None:

        self.\_QA = {"Привет": Answer("И тебе привет")}

        self.\_stats = {

            "qa\_frequency": {},

            "users\_questions": {},

        }

В классе App есть следующие методы: add\_QA – метод для обновления вопроса и ответа, get\_answer – метод для получения ответа на заданный вопрос, get\_most\_frequent\_questions – метод для получения определенного количества самых частых вопросов, get\_questions\_by\_users – метод для выгрузки вопросов пользователей.

    def add\_QA(

        self, question: str, answer\_message: str, answer\_file: Optional[str] = None

    ) -> bool:

        self.\_QA[question] = Answer(answer\_message, answer\_file)

        return True

    def get\_answer(self, question: str, user\_id: str = "test user") -> Answer:

        if question in self.\_QA:

            answer = self.\_QA[question]

            self.\_stats["qa\_frequency"][question] = (

                self.\_stats["qa\_frequency"].get(question, 0) + 1

            )

            user\_question\_set: set[str] = self.\_stats["users\_questions"].get(

                user\_id, set()

            )

            user\_question\_set.add(question)

            self.\_stats["users\_questions"][user\_id] = user\_question\_set

            return answer

        else:

            return "Произошла ошибка!"

    def get\_most\_frequent\_questions(self, return\_count: int = 10):

        freq\_items = self.\_stats["qa\_frequency"].items()

        frequency = [[] for \_ in range(len(freq\_items) + 1)]

        for question, count in freq\_items:

            frequency[count].append(question)

        res = []

        for questions in reversed(frequency):

            for question in questions:

                res.append(question)

                return\_count -= 1

                if return\_count == 0:

                    return res

        return res

    def get\_questions\_by\_users(self, filename: str = "./export.txt"):

        with open(filename, "w") as file:

            for user, questions in self.\_stats["users\_questions"].items():

                file.write(f"{user}: {questions}\n")

        return self.\_stats["users\_questions"]

**Краевые случаи для функций сервиса**

* получение ответа на несуществующий вопрос;
* получение самых популярных вопросов или пользователей и их вопросов при отсутствии запросов;
* добавление уже существующего вопроса.

**Тест-кейсы для функций**

Таблица 1 – Тест-кейс для функции add\_QA

|  |  |  |
| --- | --- | --- |
| **Номер** | 1 | |
| **Название** | Проверка на добавление ТС | |
| **Входные данные** | **Ожидаемый результат** | **Результат** **теста**:   * Пройден! * Провален |
| question = “Как дела?”, answer\_message = “Хорошо” | QA добавлен | QA добавлен |

Таблица 2 – Тест-кейс для функции get\_answer

|  |  |  |
| --- | --- | --- |
| **Номер** | 2 | |
| **Название** | Проверка обновления координат ТС | |
| **Входные данные** | **Ожидаемый результат** | **Результат теста**:   * Пройден! * Провален |
| question = “Как дела?”, user\_id = “asd4ds” | Ответ “Хорошо” | Ответ “Хорошо” |

Таблица 2 – Тест-кейс для функции get\_most\_frequent\_questions

|  |  |  |
| --- | --- | --- |
| **Номер** | 3 | |
| **Название** | Проверка уведомления о прибытии ТС | |
| **Входные данные** | **Ожидаемый результат** | **Результат теста**:   * Пройден! * Провален |
| return\_count = 1 | [“Как дела?”] | [“Как дела?”] |