ПРАКТИЧЕСКОЕ ЗАДАНИЕ 2

Необходимо доработать программу, сделанную в первом задании, в отдельной ветке репозитория, создавать новый проект для этого задания не требуется

Необходимо создать отдельную ветку

git checkout -b <имя_новой_ветки>

Установить необходимые зависимости

Создать виртуальное окружение. venv

pip install openpyxl python-docx

Создать файл. gitignore с указанием. venv, а также папок потенциальных которые будет генерировать ваша программа, как запретные для хранения на git

Файловая структура и пути

Хранить Excel в data/users.xlsx (добавить папку в .gitignore).
Все выгрузки — в dumps/<UUID>/ (добавить папку в .gitignore).
Только относительные пути внутри проекта (никаких абсолютных — иначе программа теряет переносимость).

Что нужно сделать (функциональные требования)

1) После запуска программы показывать меню:

1. **«Добавить новую учётную запись»** (повторяет сценарий регистрации с новыми полями смотри пункт 2). Запись с данными нового пользователя помещается в excel документ (см. пункт 3)
2. **«Вывести список пользователей»**

   В поле консоли выводится полный перечень пользователей UUID и НИК

   2.1 Появляется снизу списка два поля

   1 – «Удалить выбранного пользователя» (см. пункт 2.2.)

   2 – «Закрыть меню» (просто выходим на верхнее меню)

   2.2 Удалить запись

   Пользователь выбирает точное позицию путем ввода номера пользователя, далее производится удаление строки с этими данными и сохраняется файл. (Удаление строк через ws.delete_rows по индексу, идти снизу-вверх.)

   При выборе удалить пользователя из документа excel удаляется строка с данными этого пользователя
3. **«Выгрузить данные пользователя по UUID»**

   Создать папку dumps/<UUID>/ (если нет — создать; использовать os.makedirs(..., exist_ok=True) или аналог на pathlib).

Сформировать **таблицу в `.docx`** (2 колонки: «Поле»/«Значение») с данными пользователя и сохранить в `dumps/<UUID>/<UUID>.docx`. Использовать `python-docx` → `Document()`, `add_heading`, `add_table`, стиль `Table Grid`.

4. **«Выгрузить дамп (JSON) по UUID»**

   Сохранить JSON-дамп в dumps/<UUID>/<UUID>.json (использовать with open(..., encoding='utf-8'), json).

   Типовая структура JSON

```
}
 "uuid": "7f8d1f0a-4e4a-4b3f-9c74-4a6b9c6b1f1a",
 "profile": {
  "last_name": "name",
  "phone": "+79991234567",
  "email": "ivan@example.com",
  "password ": "password",
 },
 "payment": {
  "card_mask": "411111******1111",
  "card_expiry": "03/29"
 },
 "meta": {
  "created_at": "2025-09-18T12:34:56Z",
 }
}
```

2) Новые поля + валидации

Добавить форму ввода:

Номер банковской карты

Проверять по алгоритму **Луна (Luhn)**.

Краткий алгоритм: идём справа налево, каждую вторую цифру удваиваем, если ≥10 — вычитаем 9; суммируем все цифры; валидно, если сумма % 10 == 0.

Дополнительно проверяйте длину (обычно 12–19 цифр) и что остались только цифры.

Срок действия карты

Форматы `MM/YY` или `MM/YYYY`; месяц — `01..12`.

3) Сообщение об успехе + запись в Excel

После успешного ввода всех данных:

Показать сообщение «Регистрация успешна».

Добавить строку в Excel data/users.xlsx **ровно с такими столбцами и порядком**:

ID	Ник	Номер телефона	Email	Пароль	Номер банковской карты

«ID» — **UUID** (`uuid.uuid4()`), генерируется автоматически. Используйте `openpyxl`: открыть/создать книгу, взять активный лист, при первом запуске — записать заголовки и далее **append**-ить строки, затем **save**.

4) Нефункциональные требования (качество кода)

Отдельные функции для проверок (luhn_valid, validate_card_expiry, validate_phone, validate_email).
Работа с файлами — через **контекстные менеджеры** (with open(...)), везде указывать encoding='utf-8'.
Код структурирован: функции, «точка входа» с меню (`main()`), никакой «магии» в глобальной области.
Сообщения об ошибках — понятные, без «traceback» в консоль.
При первом запуске — создать книгу и заголовки; далее — дописывать строку и сохранять.
UUID сгенерировать и поместить в столбец «ID».

5) Ограничения и замечания

Пароль в этом задании можно хранить как строку (для учебных целей). *Опционально*: добавьте хеширование (SHA-256/bcrypt) как «звёздочку+».

Номер карты **храните как строку**, без пробелов/дефисов.

Хеш строки (SHA-256)

```python
import hashlib


def sha256_text(text: str) -> str:
    return hashlib.sha256(text.encode("utf-8")).hexdigest()


print(sha256_text("hello"))  # 2cf24dba5fb0a...
```