2.1 Функціональні та нефункціональні вимоги до програми

|  |  |
| --- | --- |
| Функціональні вимоги |  |
| Процесно-орієнтовані  Інформаційно-орієнтовані | * система повинна дозволяти користувачу зареєструватись в додаток двома способами: за допомогою використання Google пошти, або прямо через додаток * система повинна дозволяти користувачу вводити данні про свої витрати декількома способами:  1. Сканування чеку та розпізнавання суми витрат та адреси місця, де покупка була здійснена, з можливістю редагувати данних при неправильному зчитуванні. 2. Введення даних про суми витрат вручну з обранням категорії, до якої відносяться витрати.  * система при зборі інформації за допомогою сканування чеку повинна за місцем здійснення витрати визначати категорію витрат та вносити суму до даної категорії, при неможливості визначити категорію повинна дати можливість користувачу вибрати потрібну. * система повинна перевіряти данні введені користувачем на коректність (суми повинні бути представлені цілими числами або числами з плаваючою крапкою) * система повинна дати змогу переглядати список усіх витрат, які були внесені за весь час (час, сума витрат, категорія), та можливість видаляти витрати за бажанням користувача. * система повинна обробляти данні за категоріями, де для кожної категорії буде створена статистика про витрати за певний період часу (якщо категорія - їжа, то за тиждень, місяць, рік; якщо категорія - навчання\курси, то період - рік) * система повинна створювати загальну статистику по усіх витратах за день, тиждень, місяць, рік * система повинна надати користувачу можливість переглянути будь-яку статистику за відповідними запитами, обраними з меню * система повинна надсилати користувачу повідомлення про неочікувано великі витрати, нехарактерні користувачу, також повідомлення про економію коштів (зекономлена сума порівняно з минулим періодом), повідомлення з порадами щодо користування коштами (зменшити\витрати на певну категорію і т. д.). * система повинна надати користувачеві можливість відмовитись від розсилки з порадами та іншим або відновити розсилку у процесі користування, а також при реєстрації заздалегідь питати про надсилання даної інформації * система повинна зберігати інформацію про особисті данні користувача (і’мя, пошта, вік, стать, електронна пошта) * система повинна зберігати данні про всі витрати користувача з часом та категорією * система повинна зберігати згенеровану статистику * система повинна мати базу з повідомленнями для користувачів |

|  |  |
| --- | --- |
| Нефункціональні вимоги |  |
| Операційні  Представлення  Безпека  Культурні та політичні | * додаток повинен працювати на мобільних телефонах Android та Apple * будь-який взаємодія між користувачем та системою має здійснюватись не більше ніж за 5 секунд * система повинна бути доступною 24 години 365 днів у році * будь-який користувач з коректними особистими даними може зареєструватись та переглядати будь-яку інформацію про витрати та статистику у будь-який момент часу * персональна інформація захищена * можливість користуватись з будь-якою валютою |

2.3 Можливості пакетів json, xml та html.

|  |  |
| --- | --- |
| HTML Parser | HTML парсинг - це процес, який приймає html код, читає його та генерує структуру об’єктів з отриманої інформації  Завдяки особливій структурі написання html сторінок програміст має змогу з малими затратами часу спарстити сторінку та отримати необхідну інформацію з неї.  HTML парсер дає змогу отримати інформацію: заголовки, параграфи, ссилки, коментарі, чистий текст і т. д.   * Створення парсеру   from html.parser import HTMLParser  class MyHTMLParser(HTMLParser):  Якщо *convert\_charrefs=True* (по замовчуванню True), всі знаки автоматично конвертуються у відповідні знаки в UNICODE.  \*зауважимо: цей парсер не перевіряє кінцеві теги на відповідність початковим   * HTMLParser.feed(*data*)   передає текст у форматі str парсеру, данні зберігаються в буфурі поки інші данні не будуть передані чи не буде викликаний метод close()   * методи:   get\_starttag\_text  handle\_starttag  handle\_endtag  handle\_startendtag  handle\_data  handle\_entityref  handle\_charref  handle\_comment  handle\_decl  handle\_pi  unknown\_decl  використовуються для опрацювання та отримання даних за певним їх типом, наприклад, початкових і кінцевих блоків, чистого тексту, коментарів, декларацію DOCType, інструкцій і т. д.   * є передбаченния опрацбвання кусків данних тобто кількох рядків * при некоректних данних також зчитує їх незважаючи на неправильних порядок |
| json | json це тип данних, які зазвичай отримуються за допомогою API. Це схоже на словники в пайтоні, які складаюсься з інших словників, ключів та значень.  Ця бібліотека надає можливість використовувати данні з json, конвертуючи їх у відповідні словники в Python  Для обробки інформації з json використовуються наступні методи різні методи наприклад:   * JSONDecoder()   конвертує типи з json на мову Python(object - dict, array -list, string - str, number - int/float)  У бібліотеці передбачено обробку вийнятків JSONDecodeError.  Використання бібліотеки передбачає кодування json файлу у таких кодуваннях: UTF-8, UTF-16, or UTF-32, UTF-8  Бібліотека сприймає байти в стрінгах і не викликає помилок, але це може спричинити помилки в майбутньому.  Можливі обмеження:  -розмір json файлу  -довжина рядків  -діапазон чисел |
| lxml | lxml це перша Python XML бібліотека, яка забезпечує підтримку XPath 1.0, XSLT 1.0.  lxml забезпечує простий API для парсингу XML та HTML.  Вона підтримує покроковий парсинг.   * для початку роботи:   from **lxml** import etree  from **io** import StringIO, BytesIO   * для читання файлу можна використовувати метод parse() який повертає ElementTree об’єкт * створення XMLпарсеру та отримання даних з нього   **>>>** parser = etree.XMLParser(ns\_clean=True) **>>>** tree = etree.parse(StringIO(xml), parser)   * створення HTMLпарсеру та отримання даних з нього   **>>>** parser = etree.HTMLParser() **>>>** tree = etree.parse(StringIO(broken\_html), parser) |

2.4

Програма використовує Google authentification API для реєстрації користувачів у додаток.

Після створення проекту в Google Console було згенеровано GOOGLE\_CLIENT\_ID та GOOGLE\_CLIENT\_SECRET, за допомогою який програма зможе отримувати особисті дані користувача від Google.

Дані будуть повернені у форматі json. Наприклад, при реєстрації мене за поштою [yezerska@ucu.edu.ua](mailto:yezerska@ucu.edu.ua) програма отримає наступні дані:

**req = Request('https://www.googleapis.com/oauth2/v1/userinfo',**

**None, headers)**

**res = urlopen(req)**

**json\_string = res.read().decode('utf-8')**

**json\_obj = json.loads(json\_string)**

**print(json\_obj)**

>>>

{

"id": "110336905728859290890",

"email": "yezerska@ucu.edu.ua",

"verified\_email": true,

"name": "",

"given\_name": "",

"family\_name": "",

"picture": "https://lh4.googleusercontent.com/-3FUychsBdjY/AAAAAAAAAAI/AAAAAAAAABI/86kcstZZcM8/photo.jpg",

"hd": "ucu.edu.ua"

}

Програма обробляє дані за допомогою бібліотек json.

Інформацію можна отримати за наступними ключами:

“id”, “email”, "verified\_email", "name", "given\_name", "family\_name", "picture", "hd"

Щоб доступитись до даних, наприклад емайлу, виконуємо наступні кроки:

name\_user = json\_obj[“email”]

>>> [yezerska@ucu.edu.ua](mailto:yezerska@ucu.edu.ua)

* У модулі app.py репозиторію курсової роботи реалізований запис даних у форматі json у файл json\_data\_example.txt для прикладу даних які буде отримувати програма.

Програма використовуватиме інформацію про користувача за такими ключами:"email", "name", "picture".

Ця інформація буде використана для реєстрації в додаток.

Також в перспективі буде використовуватись інший GoogleAPI для визначення типу магазину за адресою, отриманою при скануванні чеку.