# 📄 Технічне завдання

## 🎯 Мета проєкту

Розробити веб-застосунок, який дозволяє користувачу перекладати текст через API DeepL після оплати через LiqPay. Усі дані мають зберігатися в базі даних PostgreSQL. Результат перекладу надсилається на вказану електронну пошту. У системі передбачена реєстрація користувача з шифруванням пароля. Додатково реалізується сторінка статистики з використанням pandas.

## ⚙️ Технології

**Бекенд**

• Python 3.11+  
• Django  
• PostgreSQL (моделі Django)  
• LiqPay API  
• DeepL API  
• Pandas

**Фронтенд**

• React

## 🧩 Основний функціонал

### 1. Реєстрація та авторизація користувача

- Реєстраційна форма: email (з валідацією), username (з перевіркою на унікальність), пароль (з валідацією: не менше 6 символів, обов’язкове число, символ, велика літера, маленька літера), підтвердження пароля.  
- Пароль хешується одним з доступних методів хешуванням (на вибір)  
- Авторизація: username, пароль.  
- Можливість вийти із акаунта (окрема кнопка абощо).  
- Передбачити наявність root користувача, який зможе дивитися статистику.

### 2. Головна сторінка (для зареєстрованих користувачів)

- Форма:  
 - Текст для перекладу.  
 - Вибір мови перекладу (якийсь селектор із заздалегідь захаркодженими мовами).   
 - Кнопка "Оплатити та перекласти". (співвідношення кількості символів і ціни – 1 грн за 10 символів)

### 3. Оплата через LiqPay\*

- Інтеграція через LiqPay API (sandbox).  
- Якщо оплата успішна:  
 - Надіслати запит до DeepL API.  
 - Перекласти текст.  
 - Зберегти результат у БД.  
 - Надіслати лист з результатом на email користувача.

- Якщо оплата неуспішна:  
 - Повідомити про це користувача, показавши snackbar на сторінці замовлення.

### 4. Надсилання листа

- Для відправки листа на пошту використати вбудовану в django функцію.  
- Щоб мати змогу надіслати листа з gmail, можна взяти будь-яку пошту, увімкнути там подвійну аутентифікацію і знайти в налаштуваннях безпеки “Паролі додатків”. Створивши новий пароль, можна буде під’єднатися за допомогою django mail, вказавши дані в конфізі.  
- Зміст листа: початковий текст, мова, переклад. Дозволяється використовувати звичайний текст. Якщо зробити вміст листа через html, буде плюсом.

### 5. Статистика (тільки для адміністратора)

- Сторінка /stats (якщо користувач зайшов як root):  
- ВСІ дані на цій сторінці повинні обраховуватися з допомогою бібліотеки pandas. (для більшої наглядності рекомендується заповнити БД вручну рандомними даними). Самі дані повинні братися з ORM Django, а не з прямих запитів до БД.  
- Показує загальну кількість перекладів, загальну кількість доходів, середній чек на замовлення (кількість прибутку / кількість замовлень), кількість зареєстрованих юзерів, кількість юзерів, які уже перекладали щось.  
 - Забезпечити можливість перегляду замовлень, їх фільтрацію за довільним ключем, сортування.

## 📑 Структура бази даних

### Таблиця: users

id – SERIAL PRIMARY KEY – Унікальний ідентифікатор  
email – TEXT UNIQUE – Електронна пошта  
username – TEXT UNIQUE – Псевдонім  
password\_hash – TEXT – Хешований пароль  
created\_at – TIMESTAMP – Дата реєстрації

### Таблиця: payments

id – SERIAL PRIMARY KEY – ID платежу  
user\_id – INTEGER (FK → users.id) – Хто платив  
amount – NUMERIC – Сума  
status – TEXT – Статус (успішний / неуспішний)  
created\_at – TIMESTAMP (або TEXT з парсом на пітоні в datetime) – Час створення  
closed\_at – TIMESTAMP (або TEXT з парсом на пітоні в datetime) – Час закриття

### Таблиця: translations

id – SERIAL PRIMARY KEY – ID перекладу  
user\_id – INTEGER (FK → users.id) – Хто перекладав  
payment\_id – INTEGER (FK → payments.id) – ID платежу  
source\_text – TEXT – Початковий текст  
translated\_text – TEXT – Результат перекладу  
source\_lang – TEXT – Початкова мова  
target\_lang – TEXT – Мова перекладу  
created\_at – TIMESTAMP – Дата і час

## 🔐 Ролі

• Звичайний користувач:  
 - Може реєструватися, авторизуватися.  
 - Може перекладати тексти після оплати.  
 - Отримує результати на email.  
  
• Адміністратор:  
 - Має доступ ЛИШЕ до сторінки /stats (на якій повинні бути усі переклади, які зробили юзери).

**🏳️‍🌈 Фронтенд роути**

Отже в кінці повинен вийти сайт з наступними сторінками:

– Авторизація / Реєстрація (зробити SPA)   
– Сторінка користувача з вводом тексту для перекладу і кнопкою лог аута  
– Сторінка адміністратора з можливістю перевірки статистики  
– Сторінка усіх перекладів користувача (з фільтрацією і сортуванням через pandas)  
– Сторінка окремого перекладу (доступна лише адміну та юзеру, який замовляв переклад)

**🇺🇦 Бекенд роути**

Зазвичай такі роути прописуються не на пряму, а через окремий шлях у посилання: не ‘/translate’, а ‘/api/translate’. Продумування назв ендпоінтів, дизай його внутрішнього функціоналу та методи (GET, POST…) повністю залежить від розробника.

**🤝 Взаємодія фронтенда та бекенда**

Фронтенд повинен відповідати відповідати за рендер сторінок за роутами (‘/auth’, ‘/stats’…) та отримку даних від бекенда.  
Бекенд відповідатиме за всі серверні обчислення, відправку пошти, зв’язок із LiqPay та DeepL.   
Тобто в загальному фронт керує виглядом та розфасовує дані, отримані від бека за допомогою REST API.  
Наприклад, при реєстрації фронт перевіряє валідність введених даних та відправляє їх на бек. Бек перевіряє чи існує уже юзер з такою поштою / юзернеймом. Якщо існує, то повідомляє про це фронт. Якщо не існує – зберігає нового користувача у базі (попередньо зашифрувавши пароль), та дає зелене світло фронту.

# Примітки

**\* LiqPay Callback**Для того, щоб зловити відповідь від LiqPay зазвичай потрібен окремий сервер, на який LiqPay надсилав би відповідь. Однак, враховуючи те, що такими технологіями володіє не кожен, рекомендується використати безкоштовний сервіс **ngrok**. Він надасть вам змогу “запостити” вашу localhost у мережу за наданим посиланням, яке і потрібно буде вказати при зверненні до LiqPay.