**Тестове завдання на позицію fullstack розробника (PHP, JS/TS, HTML/CSS):**

1. **Backend**

**Реалізувати web application на PHP**

Вхідні параметри (тип запиту post, формат даних json ):

**idClient** -- ідентифікатор клієнта (обов’язкове поле, number)

**dateBirthday** --дата народження клієнта (обов’язкове поле, string)

**phone** -- номер телефону клієнта (не обов’язкове поле, string)

**mail** – e-mail клієнта (не обов’язкове поле, string)

**address** -- адреса клієнта (не обов’язкове поле, string)

**monthSalary** -- сума доходу клієнта (не обов’язкове поле, якщо дані відсутні то вважаємо що дохід = 0, number)

**currSalary** -- валюта доходу клієнта (не обов’язкове поле, якщо дані відсутні то вважаємо що валюта = UAH, string)

**requestLimit** – бажана сума кредитного ліміту (обов’язкове поле, number)

Реалізувати валідацію вхідних даних на відповідність заданим типам даних, та перевірити наявність обов’язкових полів.

Якщо перевірка не пройдена, повернути у відповідь помилку з рекомендацією.

Якщо перевірка пройдена успішно, сгенерувати та надати у відповідь Ref (унікальний ідентифікатор заявки, string).

Сконвертувати суму доходу (monthSalary) в національну валюту, викликавши будь-яке публічно доступне апі з курсом валют (наприклад <https://api.privatbank.ua/p24api/pubinfo?json&exchange&coursid=5>)

Розрахувати limitItog (кредитний ліміт, який ми можемо надати) по формулі:

limitItog = k\*сума доходу клієнта в національній валюті, де

k = 0.95 - для клієнтів з мобільним оператором kyivstar

k = 0.94 - для клієнтів з мобільним оператором vodafone

k = 0.93 - для клієнтів з мобільним оператором lifeCell

k = 0.92 - для клієнтів з іншими мобільними операторами

Якщо limitItog більше ніж requestLimit, то обмежити значенням requestLimit.

Якщо вік клієнта менше 18 років, то requestLimit = 0.

Якщо limitItog більше 0 - вважати що поле decision = “accept”, інакше “decline”.

Виконати запис параметрів до БД в таблицю Client з рішенням по кредиту.

Структура таблиці:

CREATE TABLE Client (

Ref varchar(30) NOT NULL, -- унікальний ідентифікатор заявки

idClient int NOT NULL, -- ідентифікатор клієнта

dateCurr timestamp NOT NULL, -- поточна дата та час на момент надходження запиту до апі

phone varchar(45) NULL, -- номер телефону клієнта

mail varchar(45) NULL, -- email

address varchar(45) NULL, -- адреса клієнта

monthSalary numeric(15,2) NULL, -- сума доходу клієнта

currSalary char(3) NULL, -- валюта доходу клієнта

decision varchar(45) NULL, -- рішення по кредиту

limitItog numeric (15,2) NULL, -- кредитний ліміт, який ми можемо надати

primary key (Ref )

);

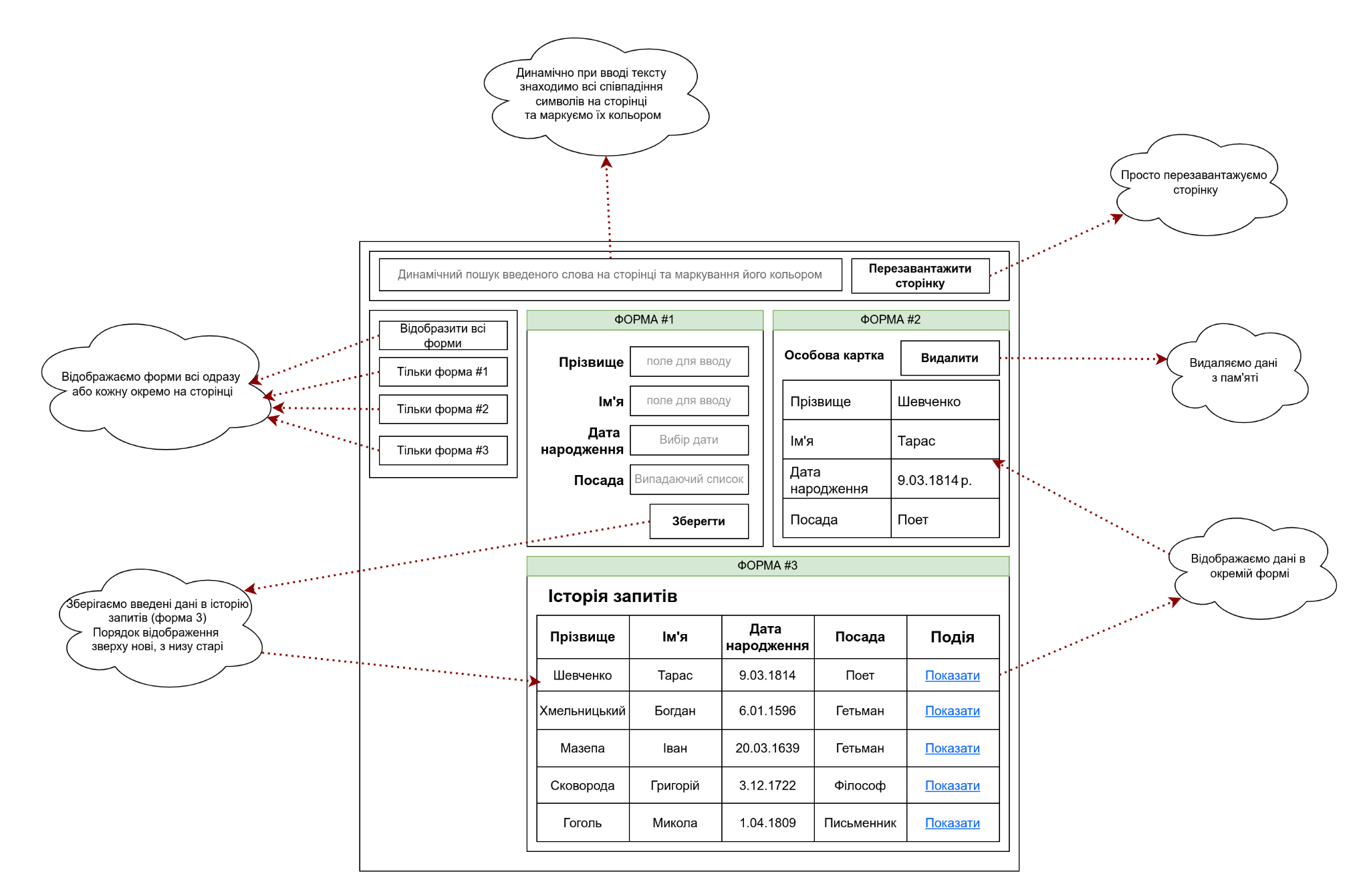
1. **Frontend частина:**

**Реалізація сторінки**

Створити сторінку згідно наведеної структури в додатку до цього ТЗ.  
Реалізувати дизайн на власний смак але дотримуватись структури макета.

Додаткові умови:

* Бажано реалізація з використанням фреймворку VueJs, Pinia, TypeScript, якщо ні то аргументувати чому
* Бажано бібліотека стилів Tailwind, в крайньому випадку Bootstrap (але щось одне)
* Мінімізувати використання кастомних стилів
* Бажано зберігання даних в Redis, в крайньому випадку кеш-пам’яті
* Динамічні події, сторінка не має перезавантажуватись
* Не використовувати jQuery

****

1. **Задача на логіку:**

**Завдання до виконання мовою JS/TS (або прописати алгоритм виконання)**

Зернова угода діє. Дано масив, що містить висоту рівнів баржі на 2d площині, причому кожен рівень має ширину 1. Яка максимальна кількість одиниць зерна, яку можна завантажити на баржу?

Кожен рівень знаходиться безпосередньо поруч із рівнем, що стоїть поруч із ним в масиві, за винятком випадків, коли показана висота 0, тоді це дно баржі.

Окрему одиницю можна уявити як двомірний квадрат шириною 1.

Слідкуйте за швидкодією: вам знадобиться рішення, лінійне кількості рівнів (кількість ітерацій має бути кратною кількості рівнів, а не збільшуватися експоненційно).

| **4** | **1** | **3** |  |  |  |  | **4** | **0** | **1** | **3** |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | **2** |  |  |  |  |  |  |  | **5** |
|  |  |  |  |  |  |  |  |  |  |  |  |

| **Початковий код**  function loadGrain(levels) {  // your code here    return 0;  } | **Приклади тестів**  loadGrain([4, 1, 3]) // 2  loadGrain([2, 1, 5, 2, 7, 4, 10]) // 7  loadGrain([2, 0, 1, 5, 2, 7]) // 6  loadGrain([2, 4, 2]) // 0  loadGrain([7, 4]) // 0  loadGrain([]) // 0 |
| --- | --- |