**АНОТАЦІЯ**

Курсова робота присвячена розробці програмного забезпечення для тренування пам’яті.

Робота містить: опис предметної галузі, постановку задачі, моделювання предметної галузі за допомогою функціональної моделі, концептуальну, логічну та фізичну моделі розроблюваної бази даних, специфікації програмних модулів, результати тестування програмних модулів.

Для реалізації використана об’єктно-орієнтована мова програмування Javascript (з використанням фреймворку Nuxt.js) для frontend та php (з використанням фреймворку Laravel) для backend сторін ПЗ та ДБМС MySQL.

Робота викладена на \_\_ сторінках друкованого тексту та містить \_\_ рисунків, \_\_ таблиць, \_\_ додатки та список використаних джерел з \_\_ найменувань.

**ABSTRACT**

The course work is devoted to the development of a software for a memory training.

The work contains the description of the subject area, problem statement, modeling of the subject area using functional model. Conceptual, logical and physical models of the developed database, specifications of the software modules, results of testing of software modules are also included.

The programming language JavaScript (with the Vue.js framework) for a frontend and the php programming language (with the Laravel framework) for a backend and MySQL DBMS were used for the implementation.

The work is set out on \_\_ pages of printed text and contains \_\_ figures, \_\_ tables, \_\_ appendices and a list of used sources with \_\_ titles in it.

Зміст

[Вступ 5](#_Toc28800500)

[1. Аналіз предметної області розвитку пам’яті. Постановка задачі на розробку програмного забезпечення «Memory.pro» з використанням баз даних. 6](#_Toc28800501)

[1.1. Аналіз предметної області тренування пам’яті 6](#_Toc28800502)

[1.2. Аналіз вже існуючого програмного забезпечення 7](#_Toc28800503)

[1.3. Дослідження та опис предметної області 8](#_Toc28800504)

[1.4. Розробка функціональної моделі бази даних. 11](#_Toc28800505)

[2. Розробка програмного забезпечення «Memory.pro» з використанням баз даних 17](#_Toc28800506)

[2.1. Концептуальна модель бази даних програмного забезпечення «Memory.pro» 17](#_Toc28800507)

[2.2 Розробка специфікації програмних модулів 19](#_Toc28800508)

[2.3 Логічна модель бази даних програмного забезпечення «Memory.pro» 21](#_Toc28800509)

[2.4 Розробка алгоритмів та графічного інтерфейсу програмного забезпечення «Memory.pro» 23](#_Toc28800510)

[2.5 Вибір Системи управління базами даних (СУБД) та інших програмних засобів 29](#_Toc28800511)

[2.6 Кодування та тестування програмного забезпечення «Memory.pro» 37](#_Toc28800513)

[ВИСНОВКИ 43](#_Toc28800514)

[Список використаної літератури 44](#_Toc28800515)

[Додаток А. Інструкція користувача 45](#_Toc28800516)

[Додаток Б. Код програмних модулів 53](#_Toc28800517)

# **Вступ**

Пам’ять – невід’ємна частина життя людини. Кожного для ми зустрічаємось з незліченною кількістю інформації, і з кожним роком її кількість тільки зростає. А людині, щоб бути успішною, потрібно постійно вчитися, постійно отримувати все нові і нові знання та навички. Але скільки людина дійсно запам’ятовує? А скільки того, що запам’ятовує вона забуває через день, місяць, рік?

Метою роботи є розробка сервісу для тренування пам’яті «Memory.pro», який призначений допомогти користувачам покращити свою пам’ять. А саме – тренування з запам’ятовування різних типів інформації (таких як числа, слова тощо), ведення статистики тренувань користування та рейтингу користувачів сервісу.

Для досягнення мети необхідно вирішити наступні завдання:

* Провести аналіз предметної області та вже існуючих аналогів програмного забезпечення.
* Розробити проект програмного забезпечення, який містить в собі:

1. Розробку, концептуальної, логічної, фізичної моделі бази даних.
2. Розробку специфікації програмних модулів.
3. Опис алгоритмів програми та розробку інтерфейсу користувача.
4. Опис СУБД та засобів програмної реалізації.
5. **Аналіз предметної області розвитку пам’яті. Постановка задачі на розробку програмного забезпечення «Memory.pro» з використанням баз даних.**
   1. **Аналіз предметної області тренування пам’яті**

За основу будемо брати вже існуючі сервіси для тренування пам’яті та світові чемпіонати з розвитку пам’яті.

Проаналізувавши вже існуючі сервіси для тренування пам’яті та світові чемпіонати з розвитку пам’яті можна зробити висновок, що для розвитку пам’яті можна тренуватися у запам’ятовуванні різноманітних простих типів інформації, таких як числа, слова, гральні карти, картинки, тощо.

Тому основна функція сервісу Memory.pro – це тренування з запам’ятовування даних різних типів, таких як цифри, слова, картинки, імена, дати, тощо. При виконанні курсової роботи будуть реалізовані два – цифри та слова.

Під час тренування користувачу необхідно запам’ятати максимальну кількість даних вибраного типу за мінімальний час, після чого відновити їх у тому ж порядку.

Після запам’ятовування формується результат тренування – що користувач запам’ятав правильно, що не правильно, скільки часу він запам’ятовував запропоновану інформацію тощо (детальніше у розділі «результати тренування»

На основі результатів користувачів формується статистика – статистика конкретного користувача та загальна статистика.

Таким чином маємо наступні основні сутності: Користувачі, Тренування, та Результати. У базі даних при цьому необхідно зберігати Користувачів та Результати тренувань.

Основним користувачем сервісу є «Користувач» (або «User»). Він характеризується наступними атрибутами:

* Id
* Прізвище
* Ім’я
* Username
* Email
* Дата реєстрації
* Дата народження

Користувачем може стати будь-який бажаючий (будь-якого віку, громадянства, національності). У кожного користувача повинен бути особистий кабінет (профіль), в якому він може бачити свої дані, а також може змінювати їх. Користувачі можуть бачити профілі інших користувачів.

* 1. **Аналіз вже існуючого програмного забезпечення**

При розробці сервісу для тренувань надзвичайно важливо проаналізувати все існуючі подібні сервіси, виявити їх позитивні та негативні сторони. Ость список основних сервісів для тренування пам’яті:

* + 1. memoryleague.com

Міжнародний сервіс для змагань із запам'ятовування Online Memory League Championship. Ним користуються багато кращих мнемоністів світу. Має всі базові типи тренувань, досить зручний інтерфейс.

Сервіс повністю англомовний. Є платним ($ 25 в рік, або $ 10 на три місяці). Сервісом можна користуватися безкоштовно, але тільки в демонстраційному режимі.

В першу чергу розрахований на швидкісне запам'ятовування невеликої кількості даних (наприклад, 80 цифр, 30 картинок). Має хорошу систему змагань (через цей сервіс проводиться один з чемпіонатів світу із запам'ятовування).

* + 1. memoryman.ru

Відносно новий сервіс по тренуванню пам'яті. Сервіс російськомовний. Містить всі основні типи тренувань, досить гнучке налаштування показу даних, кількості елементів. Має простий і досить зручний інтерфейс, мінімалістичний дизайн.

На даний момент найбільш популярний в російськомовному співтоваристві мнемоністів. Але має велику кількість помилок і недоробок.

* + 1. Powermemory.ru

Один з перших російськомовних сервісів по тренуванню пам'яті (в контексті мнемоніки). Надійний і стабільний, але містить тренування тільки із запам'ятовування чисел, карт і слів. Слабка можливість настройки тренування. З 2017 року сервіс закритий на оновлення.

* 1. **Дослідження та опис предметної області**

Основним користувачем сервісу є «Користувач» (або «User»). Він характеризується наступними атрибутами:

* Id
* Прізвище
* Ім’я
* Username
* Email
* Дата реєстрації

Тренування

Тренування - основна функція сервісу. Тренування складається з наступних етапів:

1. Вибір тренування. На цьому етапі користувачу необхідно вибрати необхідний вид тренування і вказати кількість елементів інформації у тренуванні.
2. Підготовка до запам’ятовування (показується зворотній відлік до початку запам’ятовування). Є необов’язковим, і може бути вимкнений при налаштуванні тренування.
3. Запам'ятовування випадкових даних відповідно до типу тренування. Користувачеві показується випадкова послідовність даних зазначеного типу відповідно до встановлених на попередньому етапі параметрам. Користувач послідовно передивляється кожен елемент та запам’ятовує його.
4. Підготовка до згадування (аналогічно до підготовки до запам’ятовування).
5. Згадування. Після запам'ятовування користувачеві пропонується відтворити дані, які вдалося запам’ятати. Відтворити їх необхідно в тому ж порядку в якому вони були показані під час запам'ятовування.
6. Результати. Після того, як користувач закінчив етап згадування йому показуються результати тренування. Саме результати тренування зберігаються у базі даних і враховуються в особистій і загальній статистиці.

Типи тренувань

1. Числа – мета користувача запам’ятати задану послідовність цифр. У рамках курсової роботу користувачу будуть показуватись по 2 цифри. Таким чином набір можливих даних для запам’ятовування складається с чисел від «00» до «99».
2. Слова – мета користувача запам’ятати задану послідовність слів. Користувачу слова показуються по одному.

Набори даних для тренування у систему вводить адміністратор. У рамках курсової роботи фізично адміністратор реалізований не буде.

Результати тренувань

Результати тренування кожного типу зберігаються в базі даних у відповідних таблицях (для кожного типу тренування окрема таблиця). За результатами будується особиста статистика користувача і загальна статистика. Якщо користувач зробив більше 10% помилок – результат оцінюється в 0 балів.

Всі результати тренувань характеризуються такими атрибутами:

* Id користувача
* Дата тренування
* Час запам’ятовування
* Час згадування
* Оцінка

Результат тренування оцінюється за формулою:

Де correct - кількість правильних відповідей,

total - загальна кількість даних,

time - час запам'ятовування в секундах.

Якщо користувач зробив більше 10% помилок, результат оцінюється у 0 балів.

Статистика

Існує два види статистики – статистика користувача та загальна статистика.

У статистиці користувача відображається список усіх його тренувань за певний період та по кожному виду тренувань.

У загальній статистиці відображаються найкращі результати користувачів по кожному виду тренування.

* 1. Розробка функціональної моделі бази даних.

DFD – це нотація, що призначена для моделювання систем з точки зору зберігання, обробки та передачі даних. DFD – основний засіб моделювання функціональних вимог до систем, що розроблюються. Основними елементами DFD є:

1. Процес – функція або послідовність дій, за допомогою яких можна з вхідних даних отримати вихідні. Для назви процесу слід використовувати дієслова.
2. Зовнішня сутність – будь-який об’єкт за межами системи, що є джерелом інформації для системи або отримувачем інформації від системи після обробки.
3. Сховище даних – сховище даних для процесів системи, інакше кажучи, це і є база даних.
4. Потік даних - в цій нотації це іменовані стрілки, що відображають, інформацію, що надходить до системи і виходить з неї.

Змоделюємо поведінку системи за допомогою контекстної діаграми (DFD 0-рівня)

Змоделюємо поведінку системи за допомогою контекстної діаграми (DFD 0-рівня)



Рисунок 1 – Контекстна діаграма системи

Словник даних контекстної діаграми представлений у наступній таблиці:

Таблиця 1 – Словник даних контекстної діаграми

|  |  |
| --- | --- |
| **Назва потоку** | **Складові потоку** |
| Дані про користувача | Вхідний потік: Ім’я, прізвище, псевдонім, дата народження, країна, місто, пароль, email.  Вихідний потік: анкета користувача |
| Дані тренувать | Вхідний потік: id користувача, параметри тренування (тип тренування, шаблон показу даних для запам’ятовування), дані тренування (дані для запам’ятовування, правильно названі дані, час запам’ятовування, час відтворення).  Вихідний потік: результати тренування |
| Статистика | Вхідний потік: тип статистики, тип тренування, id користувача, період для формування статистики.  Вихідний потік: статистика заданого типу. |
| Набори даних | Вхідний потік: тип набору даних, набір даних  Вихідний потік: список наборів даних |
| Звіти про набори даних | Вхідний потік: тип набору даних  Вихідний потік: звіт про набір даних |

Виконаємо декомпозицію контекстної діаграми та побудуємо DFD 1-рівня

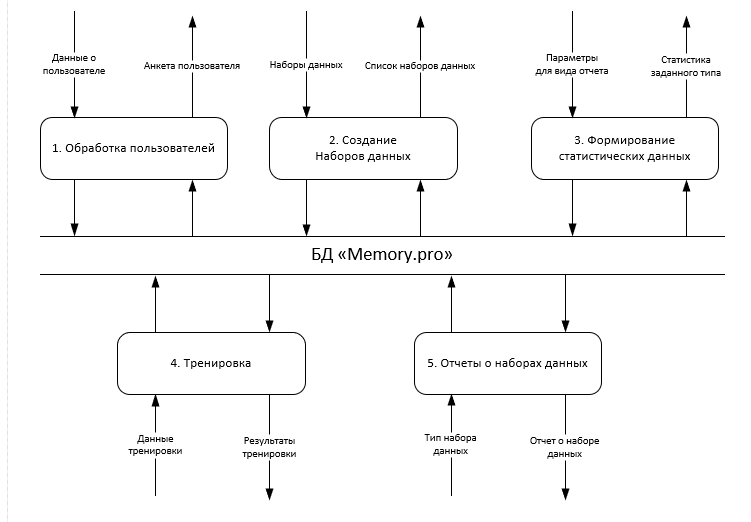


Рисунок 2 – DFD 1-рівня

Формування звітів

Розкриємо детальніше процес отримання статистики: DFD 2 рівня



Рисунок 3 – DFD 2-рівня (статистика)

Для усунення будь-яких розбіжностей при побудові DFD 0, 1, 2, 3 рівнів побудую таблицю відповідності потоків даних на усі рівнях:

Таблиця 2 ­– Таблиця відповідності потоків даних на усіх рівнях

|  |  |  |  |
| --- | --- | --- | --- |
| **Потік даних**  **0 рівня** | **Потік даних**  **1 рівня** | **Потік даних**  **2 рівня** | **Атрибути** |
| Дані про користувача | Дані про користувача |  | Ім’я, прізвище, псевдонім, дата народження, країна, місто, пароль, email |
| Набори даних | Набори даних |  | Тип набору даних, набір даних |
| Статистика | Дані статистики | Дані статистики користувача | Тип статистики, тип тренування, id користувача, період для формування статистики |
| Дані загальної статистики | Тип статистики, тип тренування, період для формування статистики |
| Дані статистики за рейтингом | Тип тренування, id користувача |
| Тренування | Дані тренувать |  | Id користувача, параметри тренування (тип тренування, шаблон показу даних для запам’ятовування), дані тренування (дані для запам’ятовування, правильно названі дані, час запам’ятовування, час відтворення) |
| Звіти про набори даних | Звіти про набори даних |  | Тип набору даних |

1.3 Деталізація вимог щодо розробки програмного забезпечення

Необхідно розробити програмне забезпечення для тренування пам’яті користувачів.

Задача програмного забезпечення надати користувачу можливість тренуватися онлайн. При цьому, користувач має мати можливість виконувати наступні дії:

* Тренуватися за різними типами інформації
* Переглядати результат свого тренування
* Мати можливість отримати свої минулі результати
* Переглядати результати інших користувачів
* Переглядати список користувачів
* Переглядати кабінет інших користувачів (при цьому користувач не має мати доступу до приватної інформації інших користувачів, такої як email та дата народження.

Таким чином кожен користувач фарактеризується атрибутами:

* Id
* Прізвище
* Ім’я
* Username
* Email
* Дата реєстрації

Крім користувача найважливішою сутністтю є тренування результатом якого є Результат. Саме результат зберігається у базі даних.

Якщо, користувач при запам’ятовуванні зробив більше 10% помилок, то результат відразу отримує 0 балів. А сам результат через деякий час буде видалено з бази даних.

Важливо відмітити яким чином організовані результати тренування.

По-перше при тренуванні користувач може вибирати «шаблон» показу елементів тренування, тобто, показувати одночасно кілька елементів інформації.

Тому Результати зберігаються у такій ієрархії:

1. Безпосередньо результат
2. Набір елементів даних тренування
3. Набір даних елементів кожного набору елементів даних тренування

(нижче приведено ER діаграму та логічну модель системи, на яких це наглядно видно).

Таким чином можна виділити наступні сутності результату:

1. Результат – містить основні відомості про тренування:

* Час початку
* Час підготовки до запам’ятовування
* Час запам’ятовування (може бути вирахованний на основі інших сутностей Результату).
* Шаблон показу елементів або кількість одночасно показуваних елементів (може залежати від типу інформації)
* Оцінку результату

1. Набір елементів результату (кожен елемент цього набору відповідає певному результату

* Час запам’ятовування 1 блоку інформації (так як елементи інформації можуть показуватись по декілька відразу, то неможливо вирахувати час на запам’ятовування кожного конкретного елементу, але можна кожної групи елементів.

1. Дані тренування – містять безпосередньо дані тренування та відповіді користування. Також є набором даних що відносяться до конкретного набору даних.

* Правильна відповідь
* Відповідь що дав користувач

Також необхідно формувати статистику результатів тренувань, яка формується на основі результатів та даних користувача.

1. **Розробка програмного забезпечення «Memory.pro» з використанням баз даних**
   1. Концептуальна модель бази даних програмного забезпечення «Memory.pro»

Концептуальне проектування - побудова змістовної моделі предметної галузі. Подібна модель створюється без орієнтації на модель СУБД та модель даних. Для створення такої моделі можна використати діаграму «Сутність-Зв’язок» (Entity Relation, ER). Концептуальна модель включає в себе опис понять предметної галузі та зв’язків між ними.

ER-діаграму для предметної галузі «Memory.pro» можна представити наступним чином:



Рисинок 4 – ER діаграма ПЗ «Memory.pro»

2.2 Розробка специфікації програмних модулів

Виконаємо декомпозицію при розробці функціональної моделі до специфікацій програмних модулів.

1. Опрацювання користувачів

* Призначення: зареєструвати користувача у сервісі.
* Вхідні дані: ім’я, прізвище, псевдонім, дата народження, країна, місто, пароль, email.
* Вихідні дані: анкета користувача
* Умова виконання: коректність введених користувачем вхідних даних
* Процедура виконання: користувач вводить дані, вони перевіряються на коректність; відбувається занесення користувача у базу даних з присвоєнням унікального id користувача та збереженням дати реєстрації.

1. Створення наборів даних

* Призначення: завантаження нових даних для тренувань у базу даних
* Вхідні дані: тип набору даних, набір даних
* Вихідні дані: список наборів даних
* Умова виконання: коректність вхідних даних
* Процедура виконання: адміністратор вводить тип даних, та завантажує данні, вони перевіряються, і додаються до бази даних

1. Формування статистики користувача

* Призначення: формування статистики конкретного користувача
* Вхідні дані: тип статистики, тип тренування, id користувача, період для формування статистики
* Вихідні дані: статистика відповідного типу
* Умова виконання: коректність вхідних даних
* Процедура виконання: після перевірки коректності вхідних даних формується статистика вказаного типу за вказаний період

1. Формування загальної статистики

* Призначення: формування загальної статистики користувачів
* Вхідні дані: тип статистики, тип тренування, період для формування статистики
* Вихідні дані: статистика заданого типу
* Умова виконання: коректність вхідних даних
* Процедура виконання: після перевірки коректності вхідних даних формується статистика вказаного типу за вказаний період

1. Формування статистики за рейтингом

* Призначення: формування статистики за рейтингом
* Вхідні дані: Тип тренування, id користувача
* Вихідні дані: статистика заданого типу
* Умова виконання: коректність вхідних даних
* Процедура виконання: після перевірки коректності вхідних даних формується статистика вказаного типу

1. Тренування

* Призначення: обробка даних тренувань користувачів
* Вхідні дані: Id користувача, параметри тренування (тип тренування, шаблон показу даних для запам’ятовування), дані тренування (дані для запам’ятовування, правильно названі дані, час запам’ятовування, час відтворення)
* Вихідні дані: результати тренування
* Умова виконання: коректність вхідних даних
* Процедура виконання: після перевірки коректності вхідних даних відбувається обробка даних тренувань користувачів

1. Звіти про набори даних

* Призначення: формування звітів про набори даних
* Вхідні дані: тип набору даних
* Вихідні дані: звіт про набір даних заданого типу
* Умова виконання: коректність вхідних даних
* Процедура виконання: після перевірки коректності вхідних даних формується звіт про набір даних заданого типу
  1. Логічна модель бази даних програмного забезпечення «Memory.pro»

Логічне проектування – створення схеми даних на основі конкретної моделі даних. На цьому етапі можна не враховувати специфіку конкретної СУБД, але необхідно враховувати специфіку обраної моделі даних.

Для розробки бази даних в межах цієї курсової роботи будемо використовувати реляційну модель даних. Реляційна модель даних використовує організацію даних у вигляді зв’язних двовимірних таблиць, кожна з яких називається реляційною таблицею або відношенням. Таблиця реляційної бази даних має наступні властивості:

1. Кожен стовпець має унікальне ім’я;
2. Усі елементи в одному стовпці таблиці мають однаковий тип та максимально допустимий розмір;
3. В таблиці не має бути однакових рядків;
4. Порядок рядків та стовпців у таблиці може бути довільним.

Основі структурні елементи реляційної бази даних – поле та запис. Поле – стовпець реляційної таблиці, що відповідає конкретному атрибуту. Запис – рядок реляційної таблиці, що відповідає конкретному екземпляру інформаційного об’єкта.

Логічна модель відрізняється від концептуальної тим, що логічна модель відображає набір відношень з первинними та зовнішніми ключами, а також усуває такий вид зв’язку як «Багато до багатьох», розбиваючи його на два зв’язки «Один до багатьох», створюючи додаткову проміжну таблицю.

Логічну модель предметної області «Memory.pro» можна представити наступним чином (звевши до одного типу інформації)



Рисунок 5 – логічна модель ПЗ «Memory.pro»

* 1. Розробка алгоритмів та графічного інтерфейсу програмного забезпечення «Memory.pro»

Програмне забезпечення Memory.prо складається з двох незалежних частин:

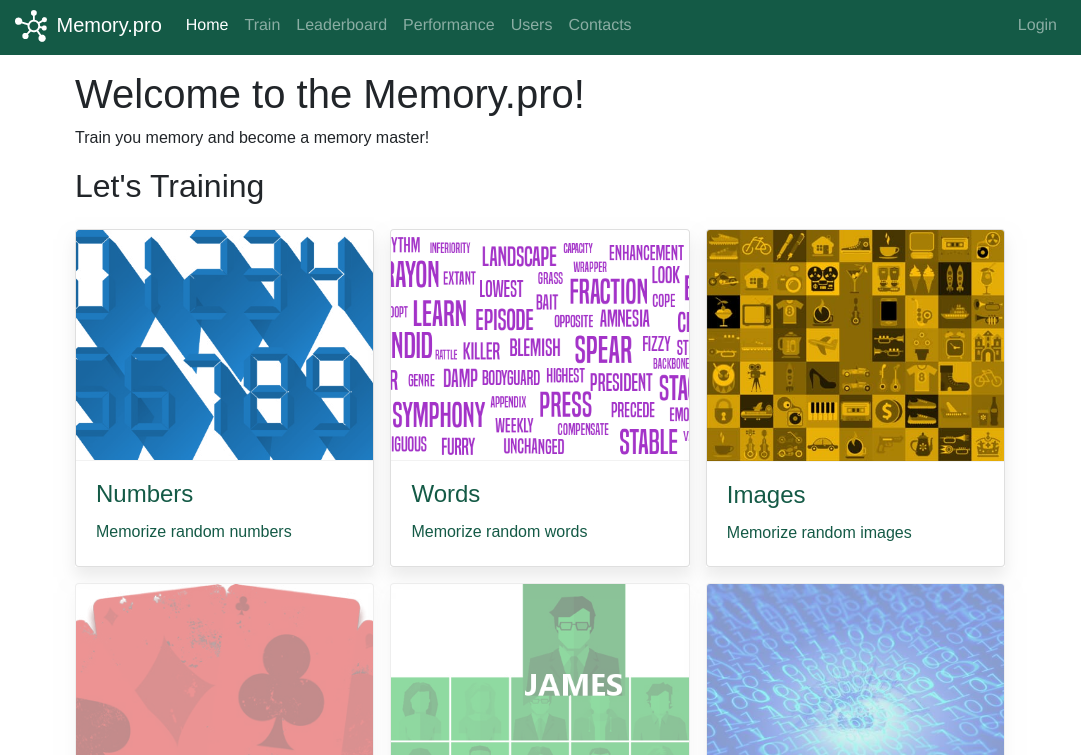
* Frontend.memory.pro
* API.memory.pro

Таким чином користувацька частина лише реалізує можливості API.

Для графічного інтерфейсу виконистана бібліотека Bootstrap

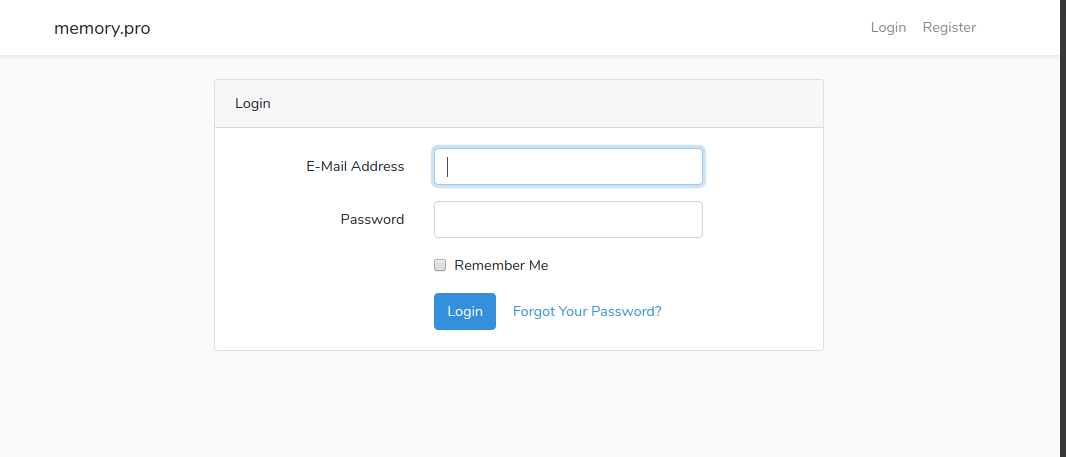
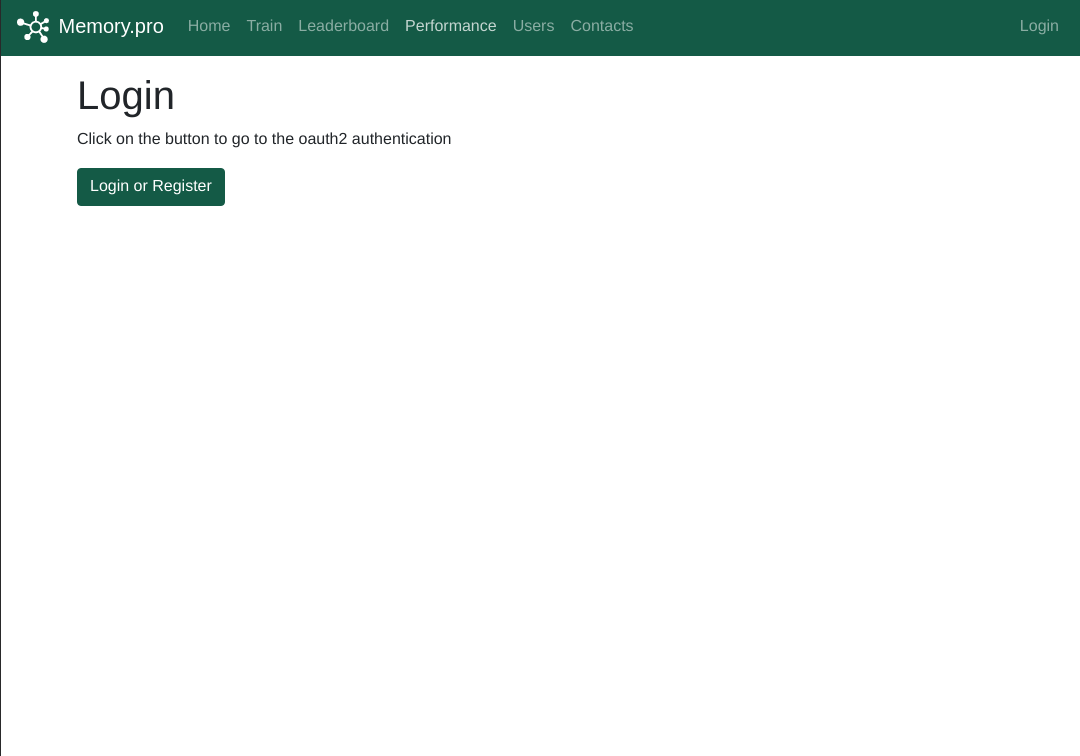
Сервіс Memory.pro – це веб сервіс. Тобто ним можна користуватись будь де и з будь-якого сучасного пристрою, що має браузер. Сервіс знаходиться за посиланням. [**https://online-memory.pro**](https://online-memory.pro)

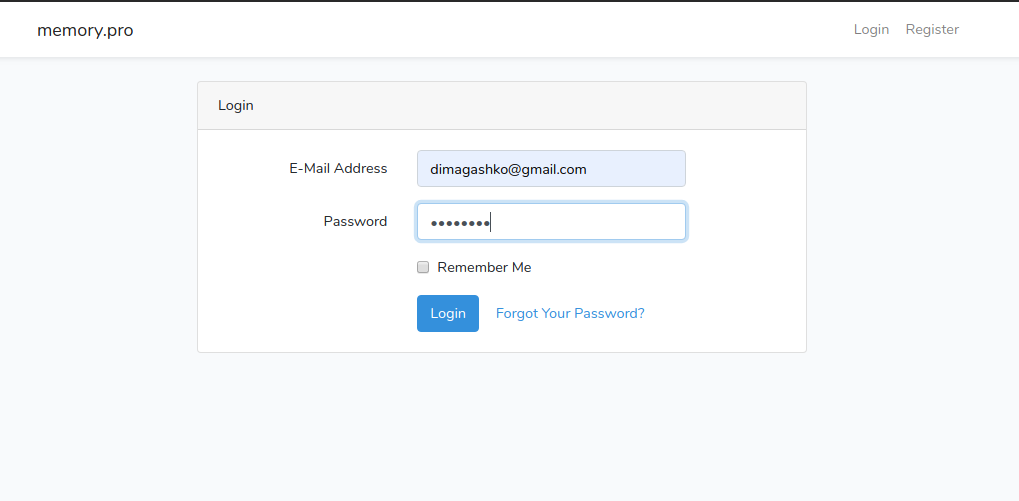
Коли користувач вперше заходить на веб сервіс його зустрічає домашня сторінка:

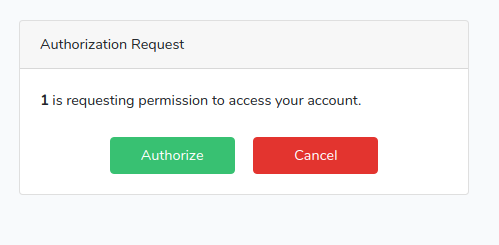


Тут можна бачити список тренувань та трохи нижче кілька останній тренувань користувачів по кожному типу тренувань.

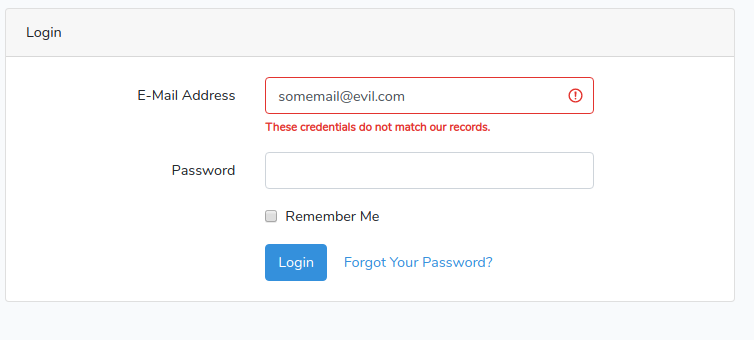
Більшість сторінок доступні усім. Навіть тренування. Але якщо користувач хоче слідкувати за своїм прогресом він має зареєструватися натиснувши на кнопку Login и потім Register or Login.



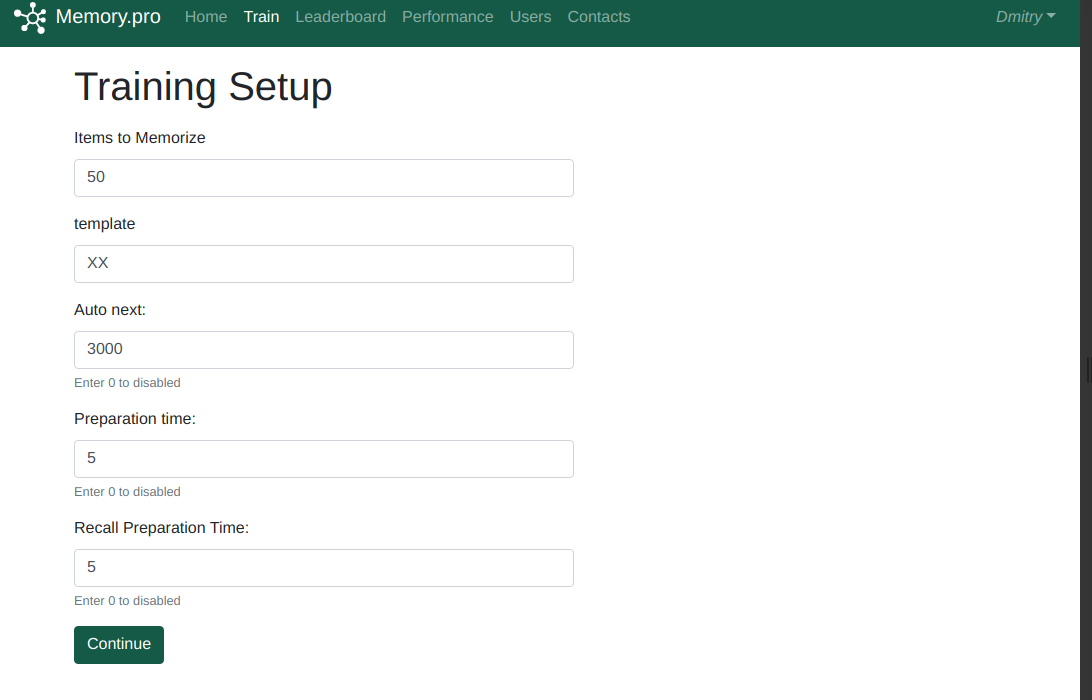




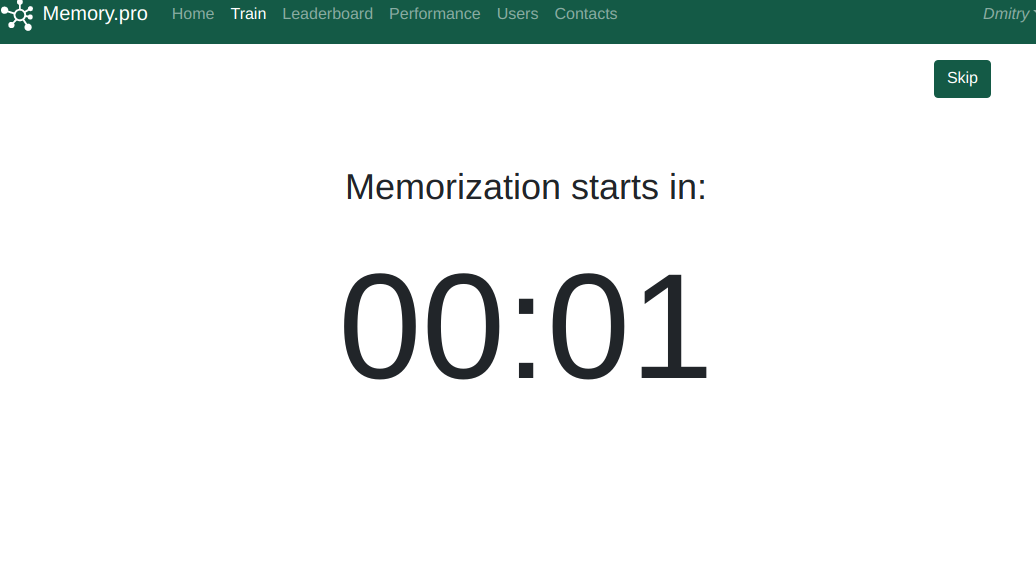
Звичайно, треба вводити лише справжні дані



Для авторизації сервіс використовує OAuth2 (як реєстрація через google на різноманітних сайтах) (за домомогою бібліотеки Laravel.Passport).

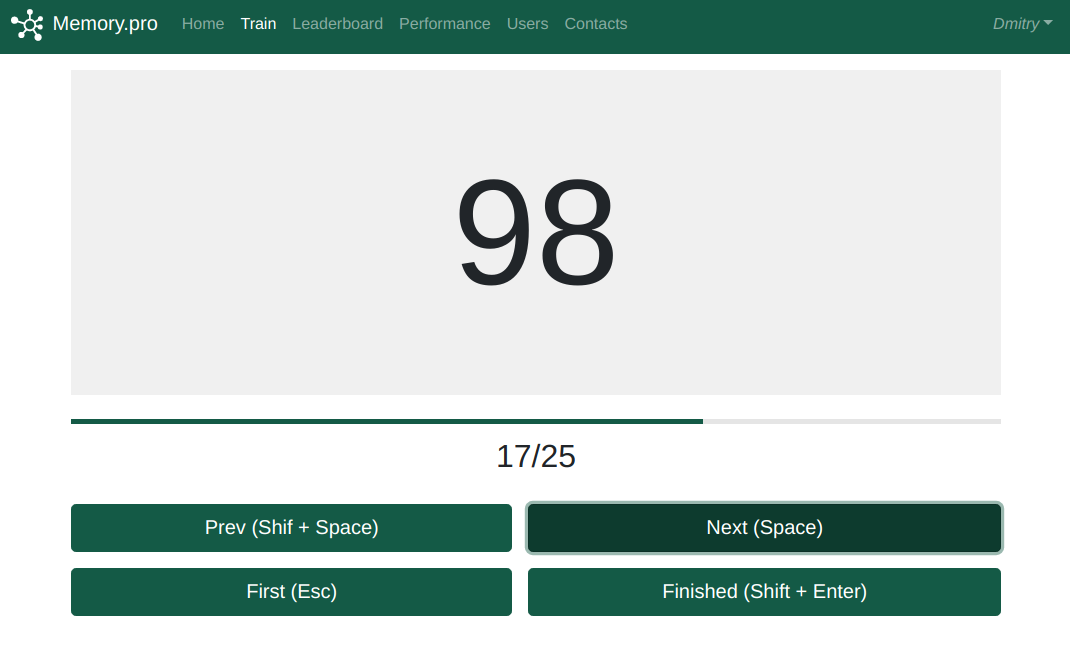
Користувачі можуть бачити профілі інших користувачив (але без приватної інформації)

Система запропонує підготуватися до запам’ятовування (цей етап можна відключити ввевши 0 у поле Preparation time):

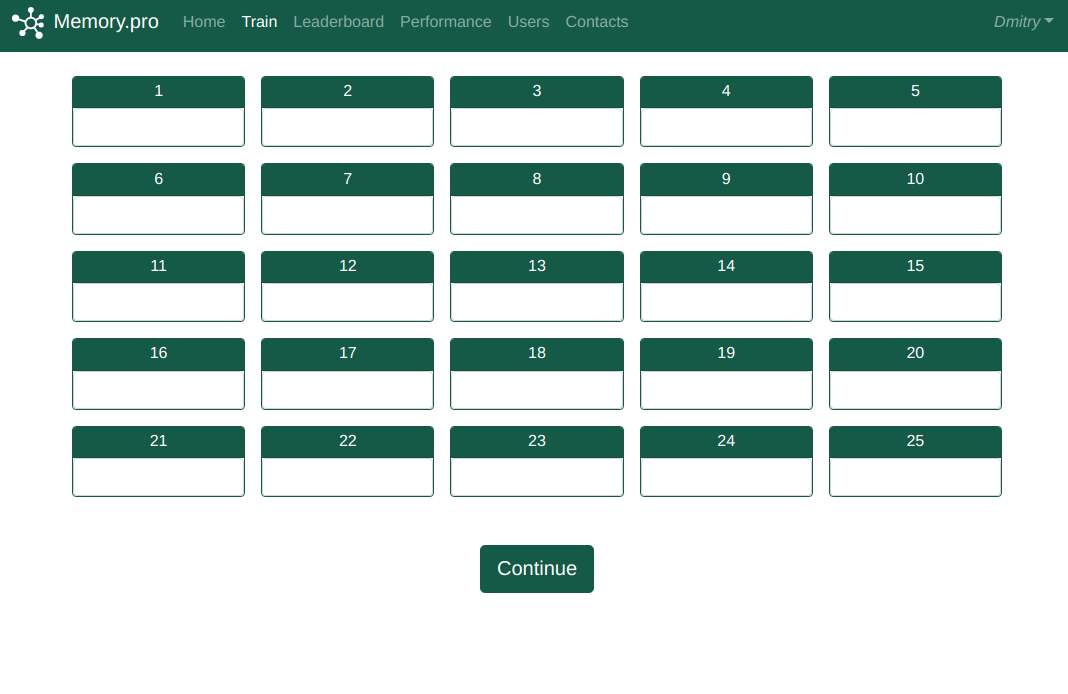


Тепер задача користувача запам’ятати якомога більше інформації.

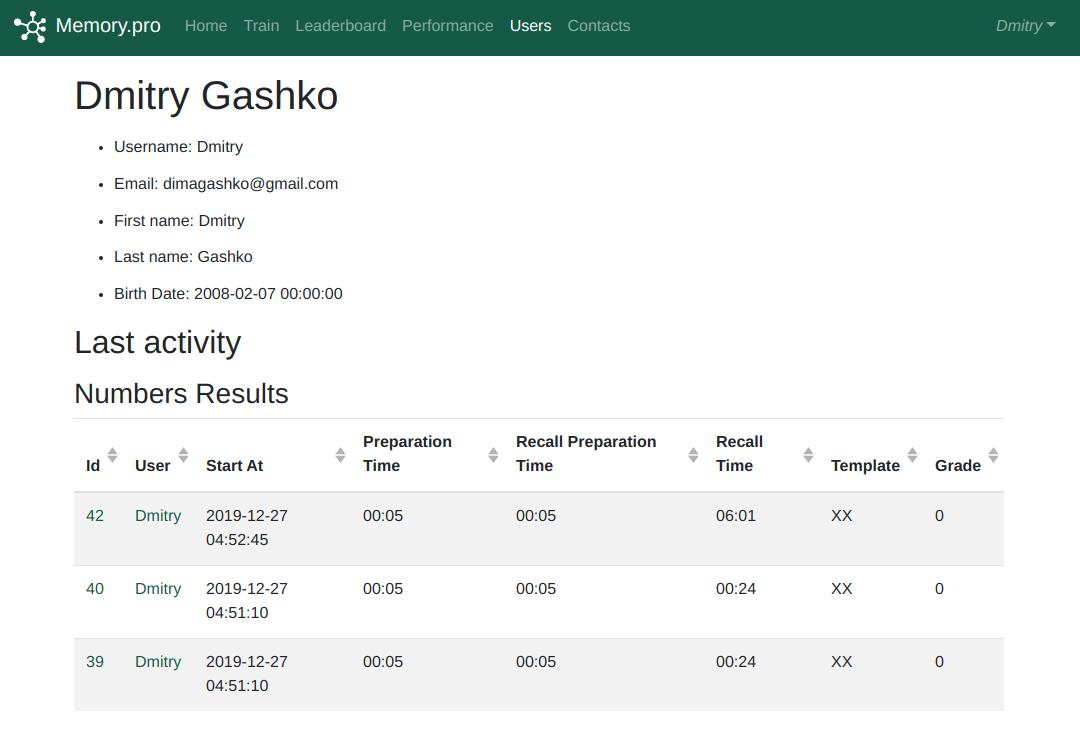
Запам’ятовування буде виглядити так:



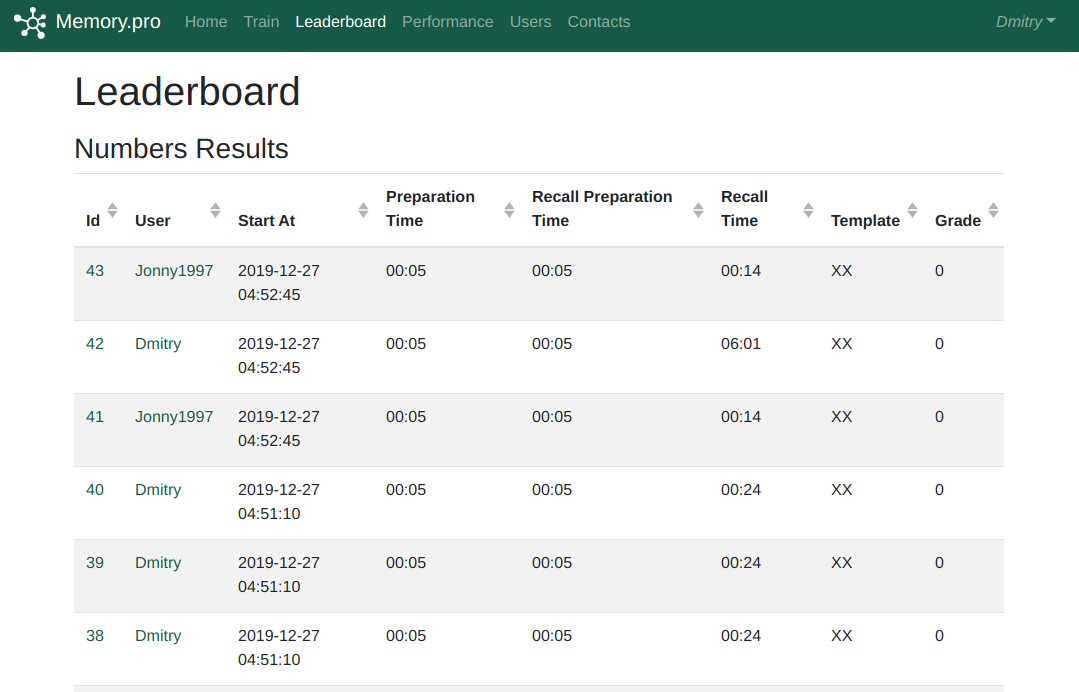
Згадування:



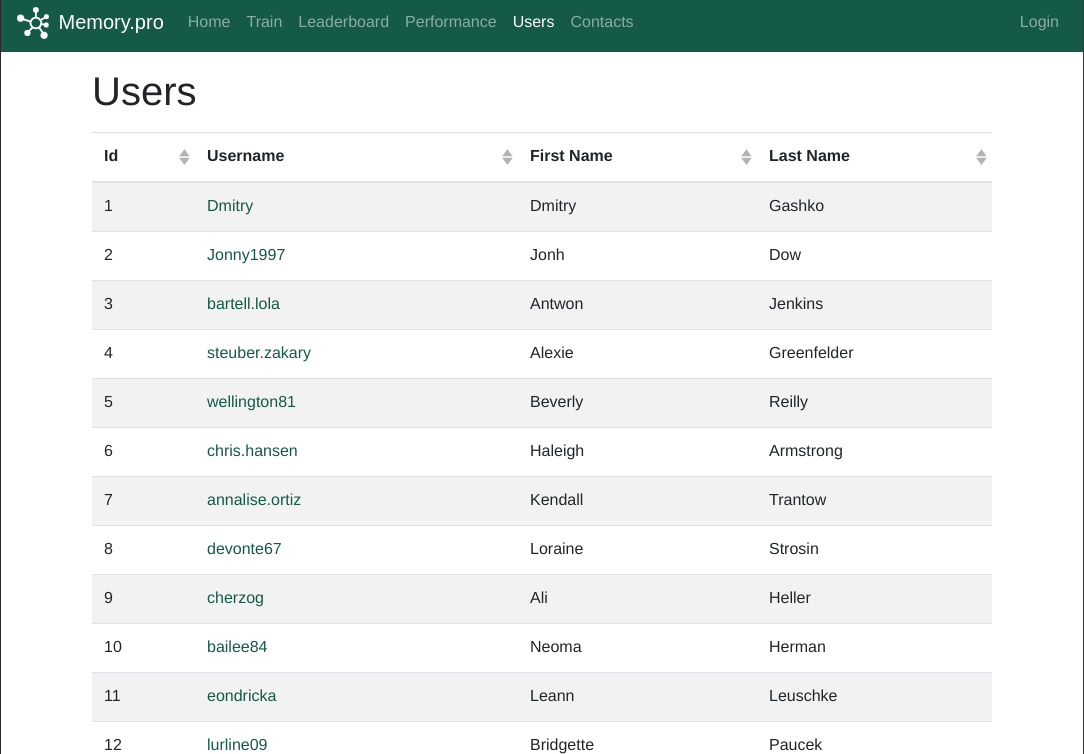
Результат тренування зберігається та його можна передивитись у будь-який час.



На сторінці Leaderboard користувачі можуть бачити остані результати усіх користувачів:



А на вкладці Users усіх зареєстрованих користувачів:



* 1. Вибір Системи управління базами даних (СУБД) та інших програмних засобів

Розглянемо програмні засоби та СУБД, що будуть використані при розробці ПЗ у рамках цієї курсової роботи.

ПЗ «Memory.pro» буде розроблено у вигляді web сервісу. Сервіс буде складатись з двох самостійних частин: Memory.pro REST API, та Memory.pro Frontend. Таким чином і мови конструювання та інші програмні засоби діляться на дві групи: для api та для frontend частин.

Для цього є такі причини:

* Незалежність від платформи

На даний момент web – єдиний спосіб розробки ПЗ, що є дійсно кросплатформеним.

Так, також є JavaFx, Electron, тощо. Але той же Electron – це теж web технології, а JavaFx хоча і є кросплатформеною, але це не завжди так. Крім того у наш час Java зазвичай використовується не для графічних додатків.

В той же час будь-який web сервіс можна запустити всюди де є браузер (будь це комп’ютер, ноутбук, планшет чи телефон), при цьому код один і той же. Його не треба компілювати для конкретних систем, тощо.

Крім того, використовуючи такі технології як ServiceWorkers, PWA (progressive web applications) дають можливість «завантажити» веб сервіс, після чого він буде виглядати як звичайний додаток (та навіть, працювати без Інтернету).

* Швидкість розробки

Сучасні web мови та frameworks дозволяють розробляти досить складні додатки за порівняно невеликі терміни.

* Можливості розширення

Завдяки тому, що сервіс розділено на дві частини: API та Frontend при необхідності можна розробити, наприклад, мобільний додаток, використовуючи той самий API.

* + 1. Вибір та обґрунтування мови конструювання та інших програмних засобів для “Memory.pro Frontend”

Було вирішено використовувати наступні мови конструювання для Frontend частини сервісу:

* JavaScript, Typescript
* VueJs
* Vuex
* Nuxt.js
* Nuxt Auth
* Scss, html
* Axios
* Bootstrap
* Yarn

Vue – це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня відображення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA, Single-Page Applications).

Vuex – бібліотека для програм Vue що реалізує патерн управління станами. Він служить централізованим сховищем для всіх компонентів програми, з правилами, які гарантують, що стан можна мутувати лише передбачувано.

TypeScript – мова програмування зі статичною типізацією що транслюється у JavaScript і при цьому дуже схожа на JavaScript (завдяки чому код після трансляції дуже схожий на код до трансліції. Завдяки статичній типізації дозволяє створювати більш надійний код, покращує можливості автозавершення.

CSS (SCSS) – Cascading Style Sheets, це мова призначена для стилізації веб сторінок (як і де відображати елементи веб-сторінки), написаних на HTML.

SCSS – препроцесор мови CSS, до додає до мови CSS можливості, яких у неї нема, але при цьому, транслюється у CSS. Додає такі можливості як змінні (вже є і в самому CSS), вкладуваність (скоро буде і в самому CSS), mixins, code splitting, цикли, умовні оператори тощо.

Pug (або Jade) – препроцесор мови HTML та шаблонизатор. Транслюється у HTML, легка у вивчені. Додає такі можливості як: більш наглядний вигляд коду, mixins, code splitting, javascript.

Webpack – сучасні веб додатки є досить великими, та зазвичай вони використовують досить велику кількість додаткових інструментів (SCSS, Pug, TS, icons, image optimization тощо), тому деякі рутинні дії хочеться автоматизувати. Webpack – один з інструментів, що дозволяють це зробити. Webpack – система для розумної збірки JavaScript-коду. Дозволяє розробляти веб додатки для максимальної зручності розробки, а під час збирання, наприклад, перетворює scss в css, pug у html, ts у js, може об’єднувати код у різних файлах в один, та багато іншого.

* + 1. Вибір та обґрунтування мови конструювання та інших програмних засобів для “Memory.pro REST API”

Було вирішено використовувати наступні мови конструювання для API частини сервісу:

* PHP
* Laravel
* Laravel Passport
* MySql
* Vue

PHP – одна з найрозповсюдженіших мов для розробки серверної частити веб додатків. Це досить просто мова, тобто її досить легко «вивчити» (що є одночасно і її плюсом і мінусом). Цю мову часто недолюблюють, через те що код написаний на ній часто може бути досить поганим. Але це лише тому, що мова дозволяє писати по різному. Сучасний PHP код напротив досить надійний. Існує багато фреймворків, що дозволяють одночасно розробляти з великою швидкістю і в той же час створювати надійні веб сервіси.

У якості систему управління базою даних було вирішено використовувати MySql – open-source система управління реляційними базами даних. MySQL є однією з найпопулярніших СУБД і підходить у більшості випадків (в тому числі і для Memory.pro), при цьому може забезпечувати достатні можливості для розширення.

Laravel – безкоштовний open-source PHP фреймворк для створення веб додатків створений Тейлором Отвелом. Призначений для створення веб додатків за архітектурним патерном MVC (model–view–controller). Базується на фреймворку Symfony.

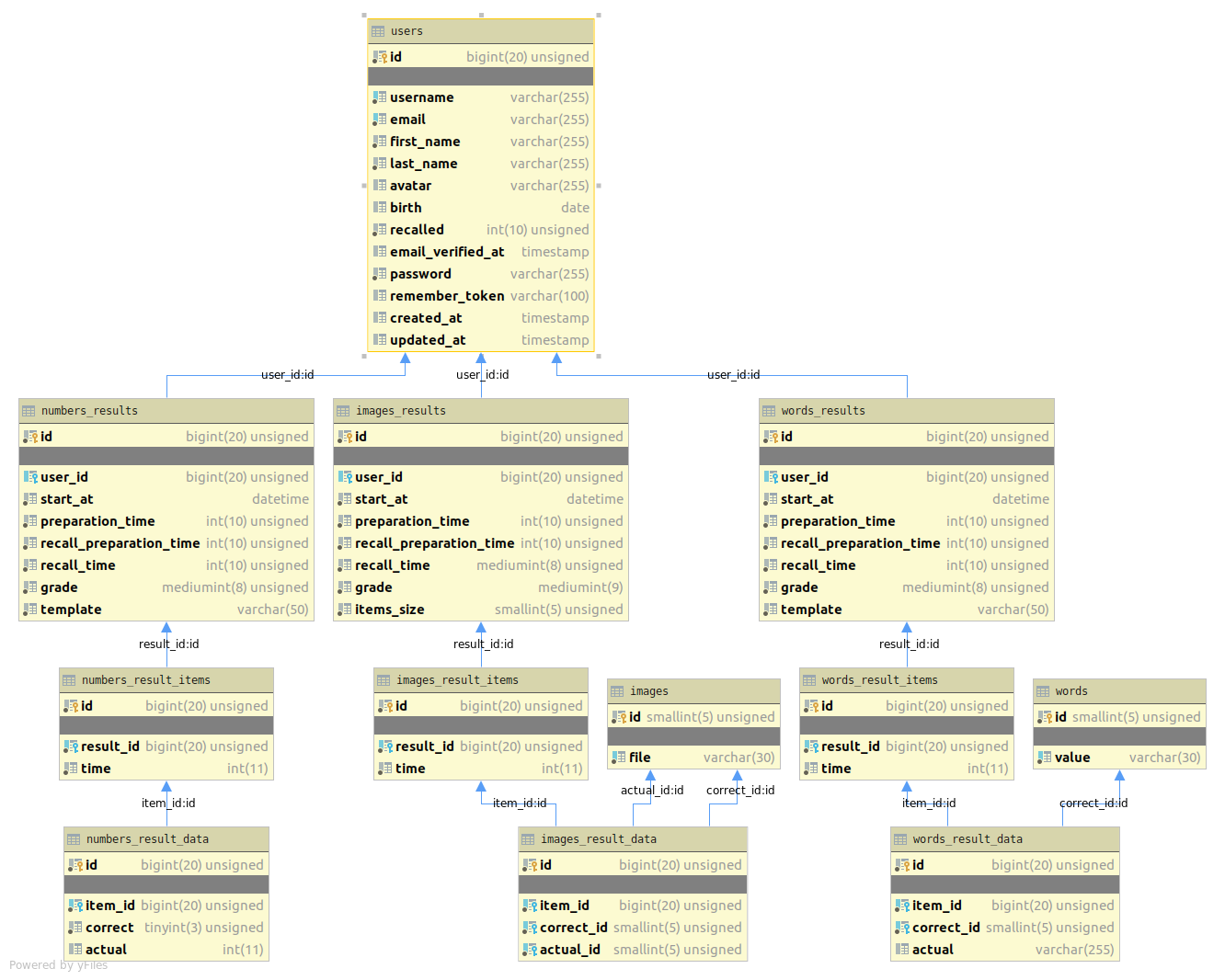


Рисунок 6 – логічна модель ПЗ «Memory.pro»

Для усунення розбіжностей в іменуванні сутностей та їх атрибутів побудуємо таблицю відповідності логічної та фізичної моделей.

Таблиця 3 – Таблиця відповідності логічної та фізичної моделей (користувач)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ім’я атрибуту в фізичній моделі | Ім’я атрибуту в логічній моделі | Тип даних | Розмір | Ключ | Обов’язковість поля |
| 1 | 2 | 3 | 4 | 5 | 6 |
| id | id | bitInt | 10 | PK | NOT NULL |
| username | username | varchar | 50 |  | NOT NULL |
| Email | email | varchar | 255 |  | NOT NULL |
| First\_name | Ім’я | varchar | 50 |  | NOT NULL |
| Last\_name | Прізвище | varchar | 50 |  | NOT NULL |
| birth | Дата нородження | date |  |  | NOT NULL |

Таблиця 4 – Таблиця відповідності логічної та фізичної моделей (результати)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ім’я атрибуту в фізичній моделі | Ім’я атрибуту в логічній моделі | Тип даних | Розмір | Ключ | Обов’язковість поля |
| 1 | 2 | 3 | 4 | 5 | 6 |
| id | id | bitInt | 10 | PK | NOT NULL |
| Start\_at | Час початку | varchar | datatime |  | NOT NULL |
| Preparation\_time | Час підготовки | Int | 4 |  | NOT NULL |
| Recall Preparation\_time | Час підготовки до згадування | Int | 4 |  | NOT NULL |
| Recall\_tome | Час згадування | int | 4 |  | NOT NULL |
| grade | Дата нородження | int |  |  | NOT NULL |

Таблиця 5 – Таблиця відповідності логічної та фізичної моделей (набір результатів)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ім’я атрибуту в фізичній моделі | Ім’я атрибуту в логічній моделі | Тип даних | Розмір | Ключ | Обов’язковість поля |
| 1 | 2 | 3 | 4 | 5 | 6 |
| id | id | bitInt | 10 | PK | NOT NULL |
| time | Час | datatime |  |  | NOT NULL |

Таблиця 6 – Таблиця відповідності логічної та фізичної моделей (дані тренування)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ім’я атрибуту в фізичній моделі | Ім’я атрибуту в логічній моделі | Тип даних | Розмір | Ключ | Обов’язковість поля |
| 1 | 2 | 3 | 4 | 5 | 6 |
| id | id | bitInt | 10 | PK | NOT NULL |
| Correct\_id | Час початку | int | 4 |  | NOT NULL |

Таблиця 7 – Таблиця відповідності логічної та фізичної моделей (набір даних для запам’ятовувавання)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ім’я атрибуту в фізичній моделі | Ім’я атрибуту в логічній моделі | Тип даних | Розмір | Ключ | Обов’язковість поля |
| 1 | 2 | 3 | 4 | 5 | 6 |
| id | id | bitInt | 10 | PK | NOT NULL |
| Value | значення | varchar | datatime |  | NOT NULL |

* 1. Кодування та тестування програмного забезпечення «Memory.pro»

Тестування - це процес виконання програми з метою виявлення помилок.

Тестовий прогін є вдалим, якщо в процесі його виконання виявлена помилка, і невдалим, якщо отримано коректний результат.

Добрим вважається тест, що має високу ймовірність виявлення ще не виявленої помилки.

Вдалим вважається тест, що виявляє ще не виявлену помилку.

Методи виявлення помилок: ручні та автоматизовані

Принципи тестування:

* Варто уникати тестування програми її автором.
* Програмуюча організація не повинна сама тестувати розроблені нею програми.
* Необхідно досконально вивчати результати застосування кожного тесту.
* Необхідно перевіряти не тільки, чи робить програма те, для чого вона призначена, але й чи не робить вона те, що не повинна робити.
* Слід зберігати тести, навіть якщо програма вже відтестована.
* Не можна планувати тестування в припущенні, що помилки не будуть виявлені.
* Імовірність наявності невиявлених помилок у частині програми пропорційна числу помилок, вже виявлених у цій частині.

Тестування “чорного ящика” - якщо відомі конкретні функції, які повинен виконувати даний продукт, можна прогнати тести, що підтверджують повну працездатність кожної з функцій. Термін “чорний ящик” значить, що при розробці тестових випадків тестувальники нічого не знають про внутрішню структуру або код. Технології, що зазвичай застосовуються під час тестування “чорного ящика” називають технологіями динамічного тестування.

Тестування “білого ящика” - якщо відомі особливості роботи всередині продукту, можна виконати тести, які підтверджують, що внутрішня робота продукту проходить відповідно специфікаціям, а всі внутрішні компоненти використовуються правильно. Термін “білий ящик” означає, що при розробці тестових випадків тестувальники використовують будь-які доступні відомості про внутрішню структуру або код. Технології, що використовуються під час тестування “білого ящика”, зазвичай називають технологіями статичного тестування.



Рисунок 7 – види тестування

TDD (test driven development) та BDD (behavior driven development) – методології розробки програмного забезпечення основна ідея яких заснована на тому що розробники самостійно пишуть unit тести, при цьому розробці кожної функції, класу, модуля тощо за наступною схемою:

* Написання мінімальної спеціфікації (що завідомо неповна)
* Написання мінімальної реалізації функції з метою «обійти» специфікацію (щоб упав тест)
* Розширення специфікації
* Вдосконалення реалізації

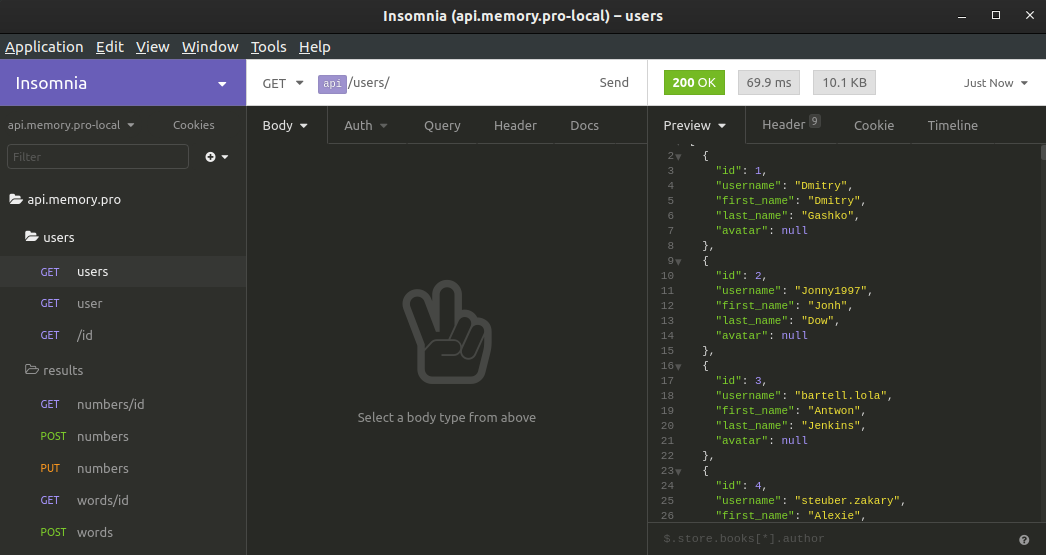
Всі ці кроки можуть повторюватись поки не буде досягнуто достатнього покриття функції тестом.

Так як програмне забезпечення Memory.pro складаесть з двох частин:

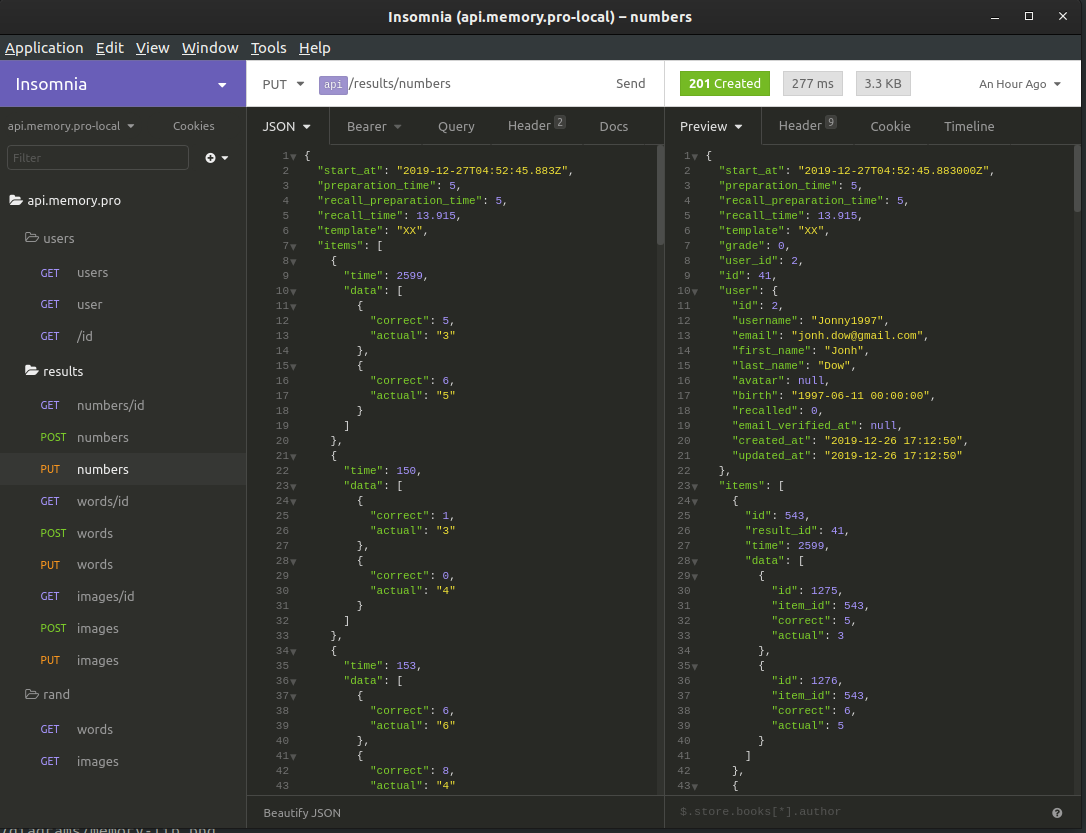
* Frontent.memory.pro
* API.memory.pro

То для тестування бізнес логіки найбільшу увагу приділемо «API.memory.pro»

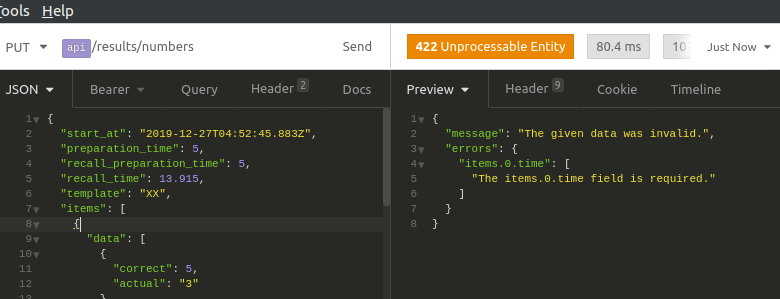
Отримання даних користувача:

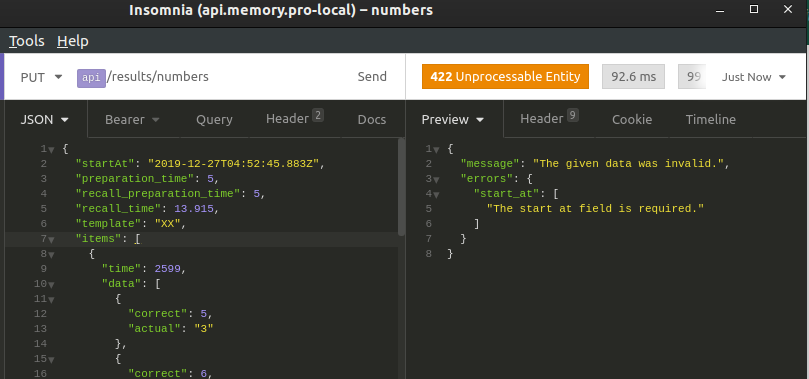


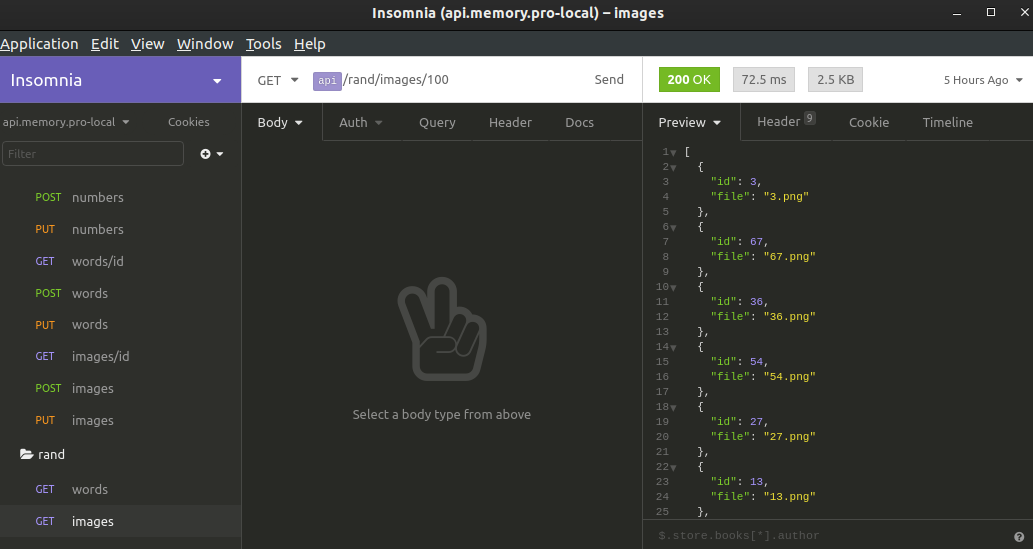
Збереження даних тренуваня

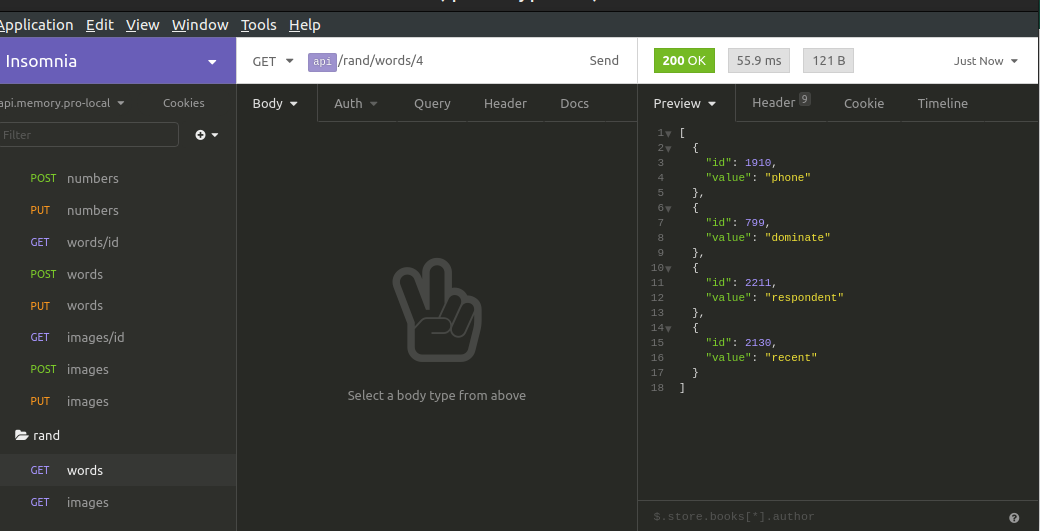


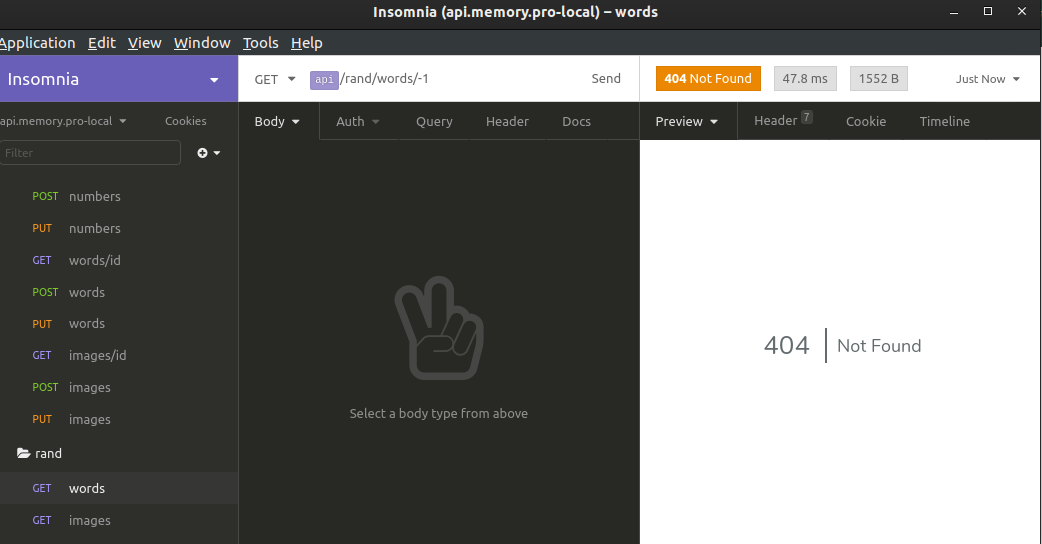
Якщо введені некоректні данні система має повідомляти про це:





Отримання випадкових даних для запам’ятовування:



Якщо дані некоректні має генеруватися помилка 404:

# **ВИСНОВКИ**

Виконавши курсову роботу, було розроблено програмне забезпечення «Memory.pro» - веб сервіс для тренування пам’яті. Розроблене ПЗ повністю відповідає поставленій задачі та вимогам до нього. У процесі розробки було досягнуто виконано наступні етапи:

1. Проаналізовано предметну галузь та виявлено істотні характеристики предметної області, що в подальшому дозволило змоделювати поведінку системи за допомогою функціональної моделі предметної галузі;
2. Розроблено детальну специфікацію програмних модулів;
3. Розроблено концептуальну модель ПЗ, що відображає зв’язки та відношення між виділеними на попередніх етапах сутностями БД;
4. В якості програмних засобів було обрано такі мови, фреймворки та бібліотеки: JavaScript, TypeScript, Vue.js, Vuex, Nuxt.js, Auth.nuxt.js, axios, html, Sass; PHP, Laravel, Laravel.passport, MySQL, Lavarel Eloquent ORM, що дали можливість розробити програмне забезпечення досить гручким та надійним
5. На основі обраної мови програмування та СУБД змодельовано поведінку системи за допомогою логічної та фізичної моделей;
6. Виконано кодування програмних модулів, текст яких наведено у додатку Б, та їх тестування за допомогою методу розбиття на класи еквівалентності;
7. Розроблено інструкцію користувача, що висвітлює усі деталі роботи з розробленим ПЗ.

# **Список використаної літератури**

1. Карпова Т. С. Базы данных: Разработка, модели, реализация [Текст] / Т. С. Карпова. – СПб.: Питер, 2001. – 304 с.
2. Андон, Ф. И. Основы инженерии качества программных систем [Текст] / Ф. И. Андон, Г. И. Коваль, Т. М. Коротун, Е. М. Лаврищева, В. Ю. Суслов –К: Академпереодика, 2007. – 672 с.
3. Офіційна документація фреймворку Laravel [Електронний ресурс] : [https://laravel.com/].
4. Офіційна документація фреймворку Vue.js [Електронний ресурс] [http://vuejs.org/].

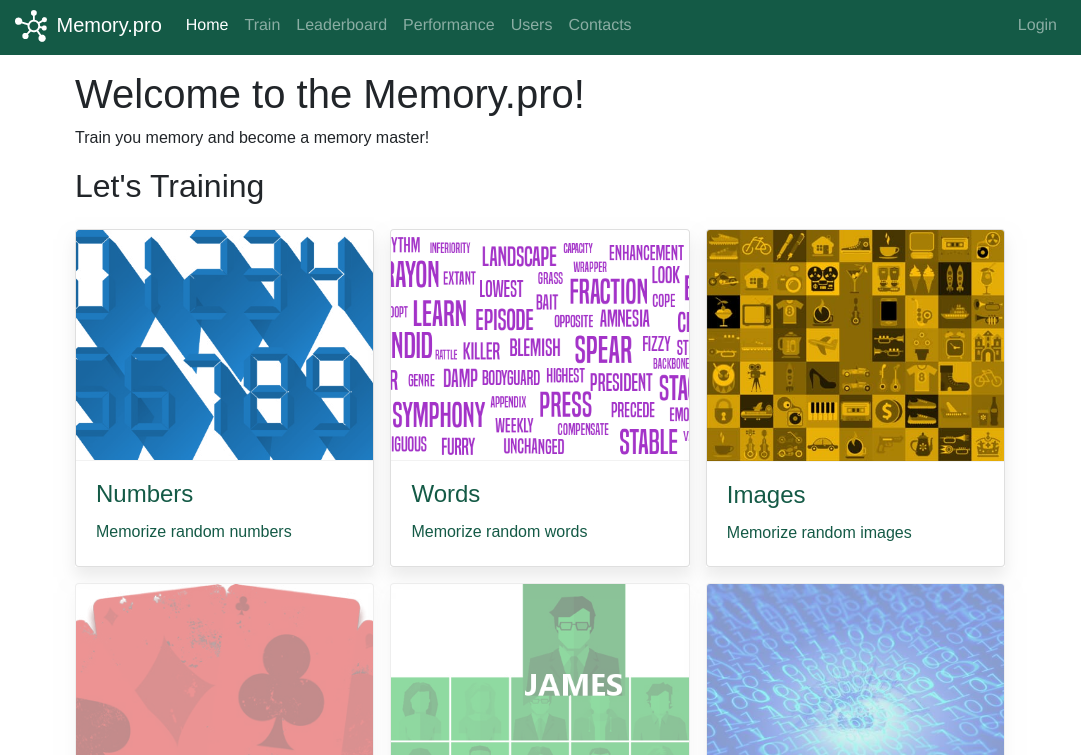
# **Додаток А. Інструкція користувача**

Дана інструкція призначена для ознайомлення користувачів з функціями ПЗ «Memory.pro» та полегшення роботи з ним.

Сервіс Memory.pro – це веб сервіс. Тобто ним можна користуватись будь де и з будь-якого сучасного пристрою, що має браузер.

Ви можете знайти його за посиланням: [**https://online-memory.pro**](https://online-memory.pro)

Коли користувач вперше заходить на веб сервіс його зустрічає домашня сторінка:

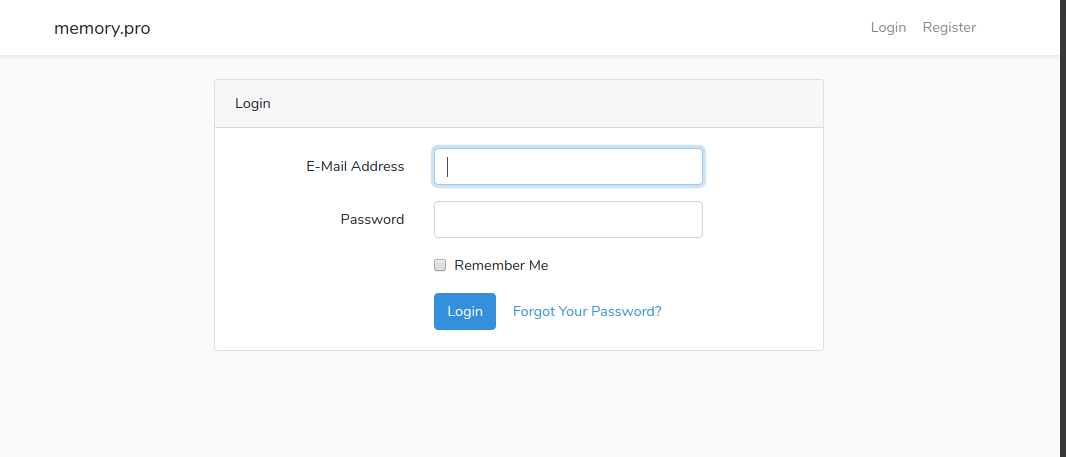
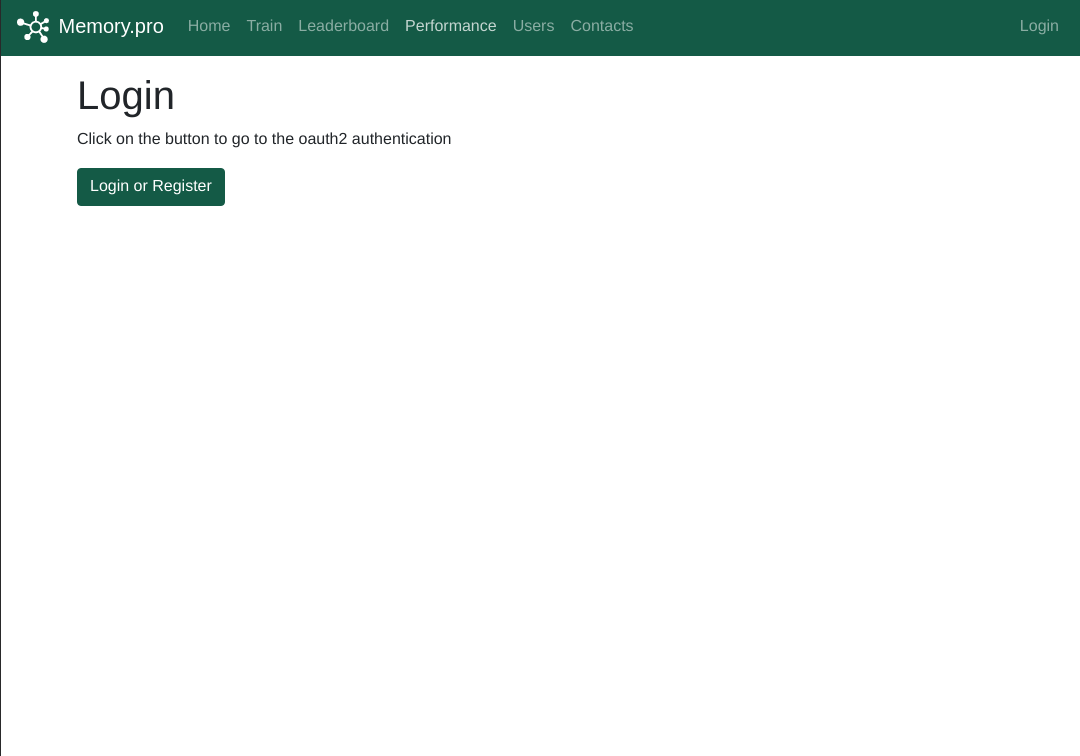


Тут можна бачити список тренувань та трохи нижче кілька останній тренувань користувачів по кожному типу тренувань.

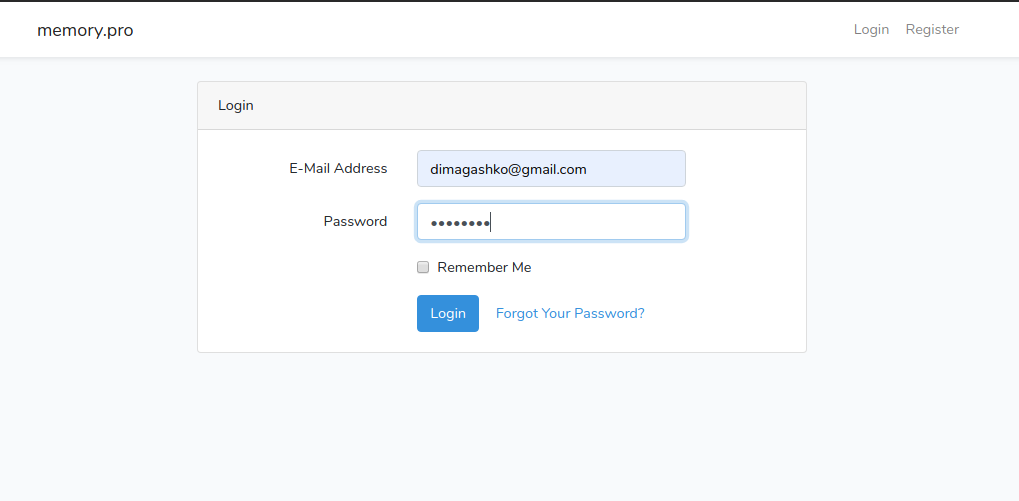
Більшість сторінок доступні усім. Навіть тренування. Але якщо ви хочете вести свою статистику та бачити свій прогрес, то ремомендуємо зареєструватися.

Для цього треба натиснути кнопку «login». Та виконувати инструкції сервісу.

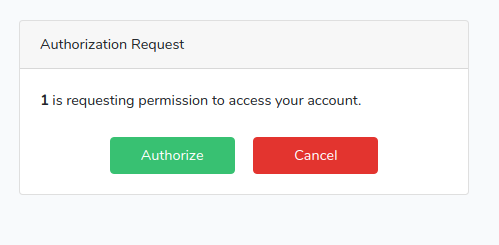
Натисність «Login or Register»:



Введіть свій email та пароль:

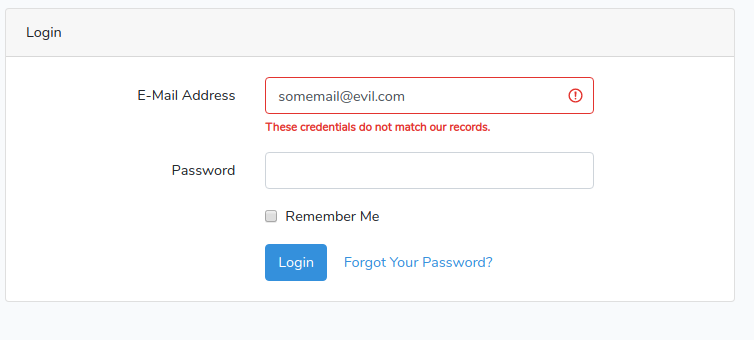


Підтвердіть авторизацію



Якщо ви ще не зареєстровані натисність «Register» (якщо не відбулося автоматичного перенаправлення на сайт сервісу, зайдіть на нього ще раз, и повторно натисніть «Login».

Звичайно, якщо під час авторизації ввести неправильні дані, то система повідомить про це:

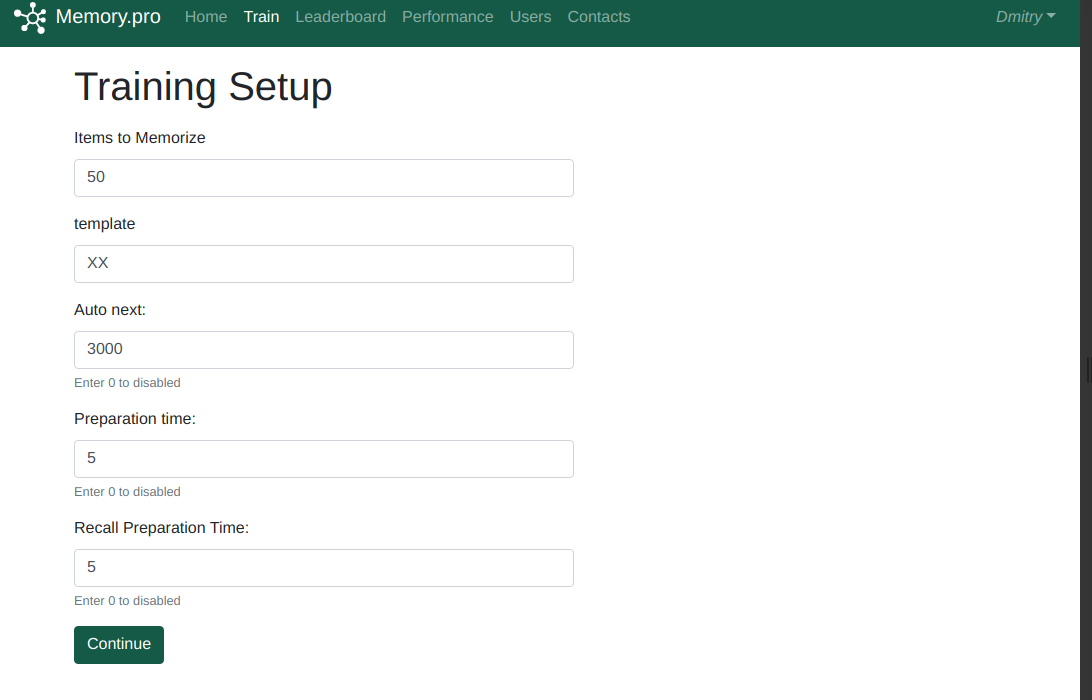


Для авторизації сервіс використовує OAuth2 (як реєстрація через google на різноманітних сайтах)

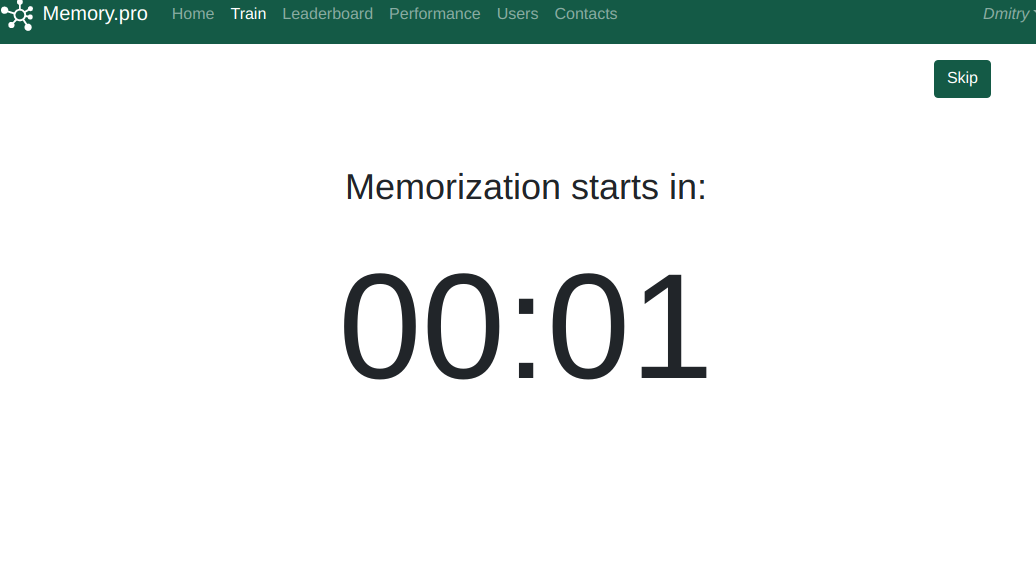
Після авторизації або реєстрації отримує повні можливості у сервісі.

Для того, щоб почати тренування, треба натиснути на один з його видів на сторінці <https://online-memory.pro/train> або на домашній сторінці.

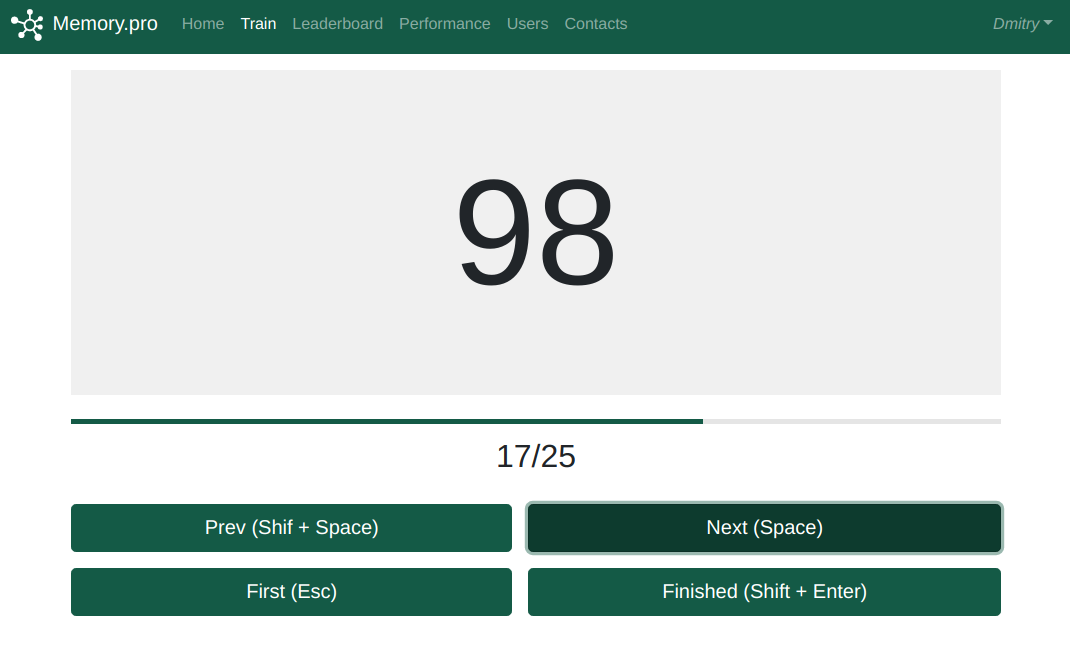
Коли тренування почнеться, перим ділом треба встановити параметри тренування:



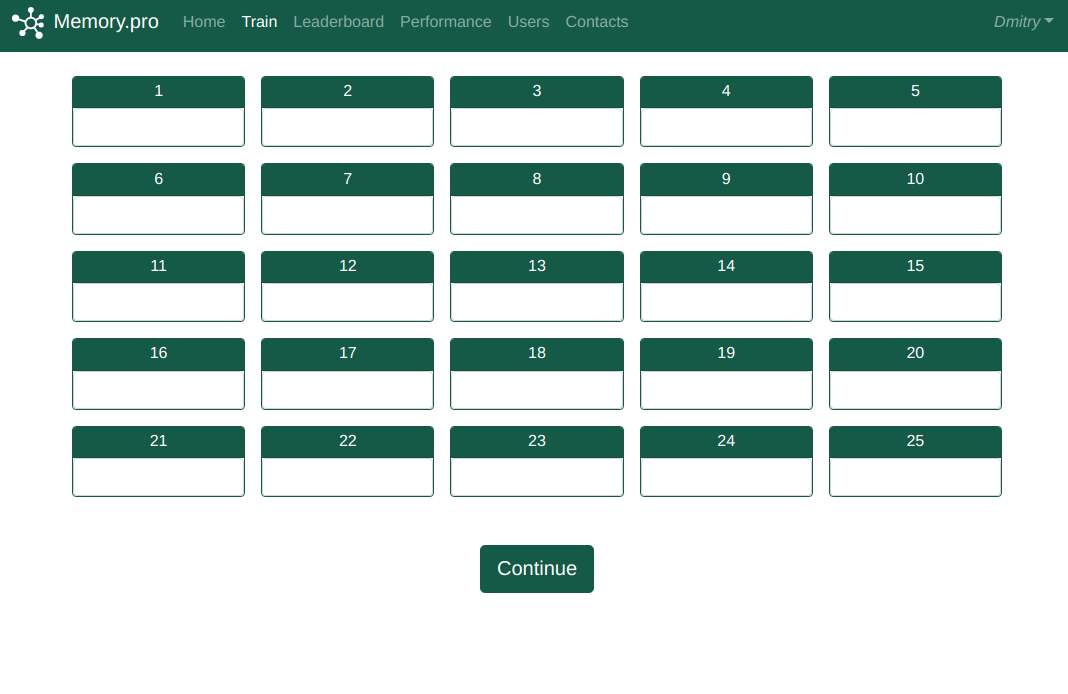
Система запропонує підготуватися до запам’ятовування (цей етап можна відключити ввевши 0 у поле Preparation time):



Тепер задача користувача запам’ятати якомога більше інформації:

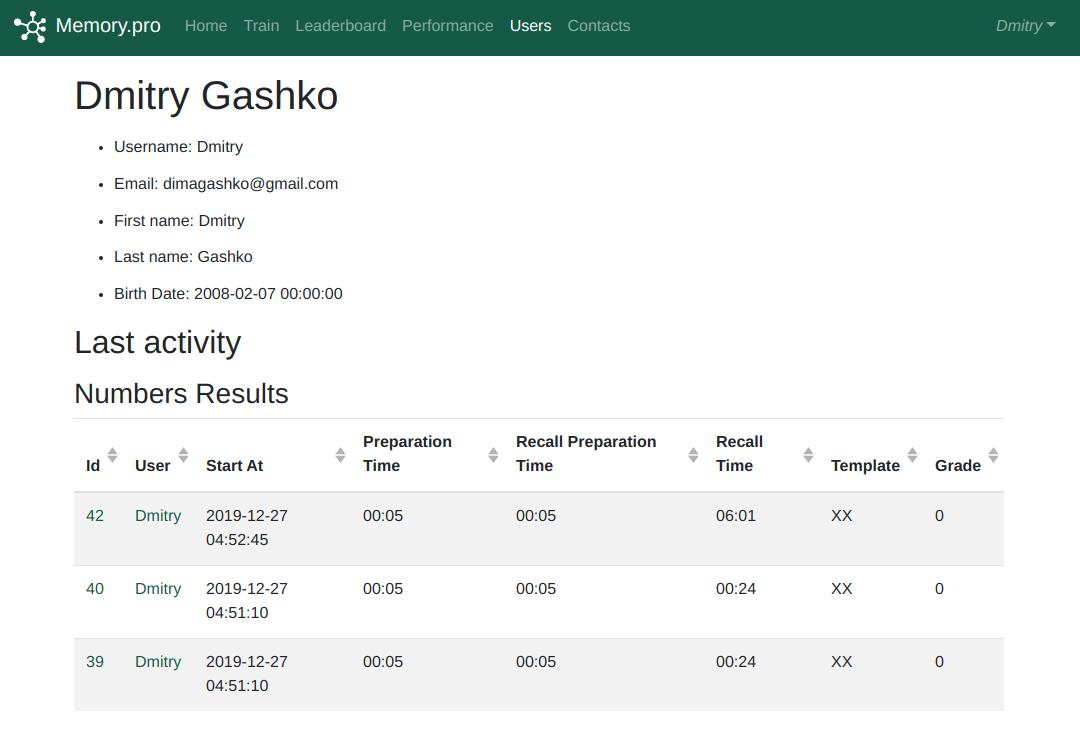


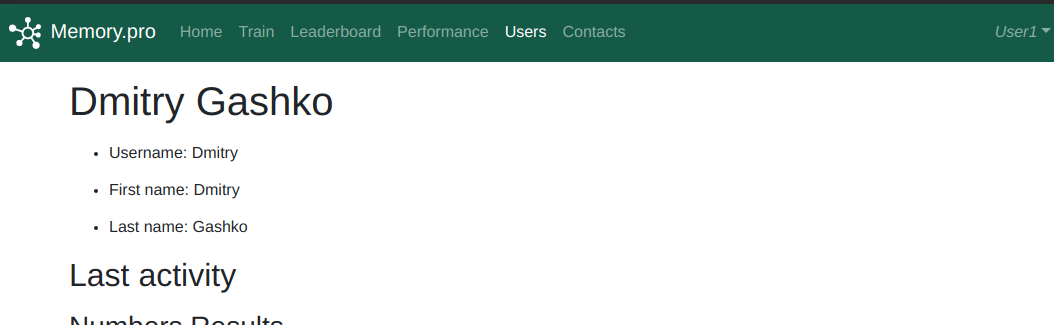
Та відтворити її у точно такій послідовності:



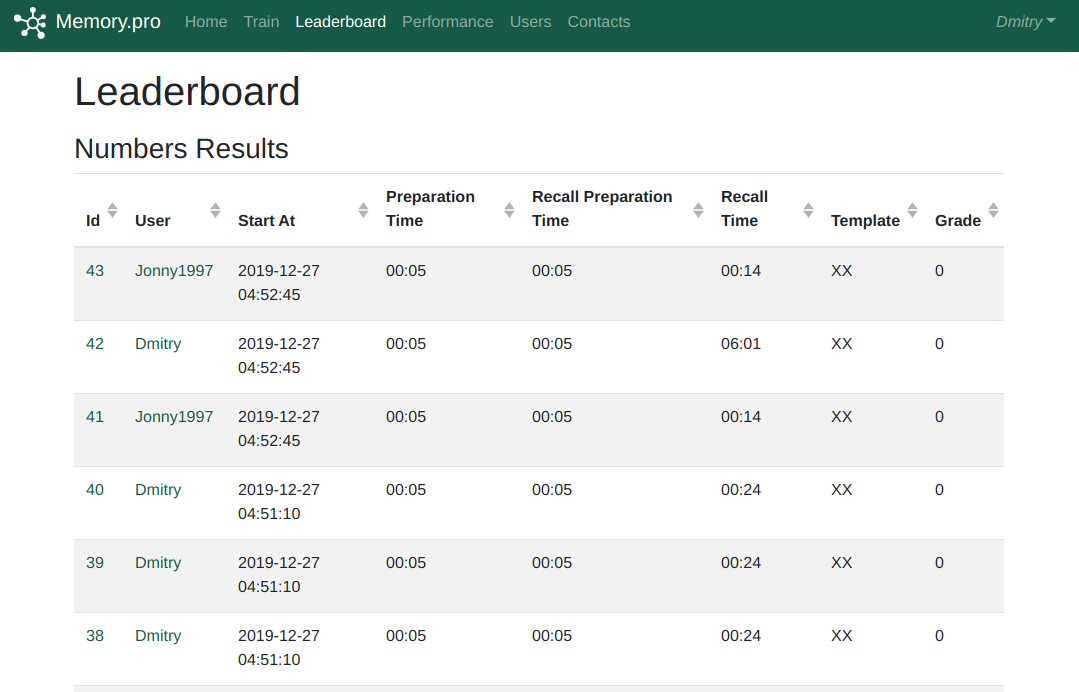
Коли ви завершите згадування, натисніть кнопку Continue щоб побачити результати.

Кожен результат користувача зберігається. Їх можна побачити на вкладці Performance, або кілька останніх у кабінеті користувача:

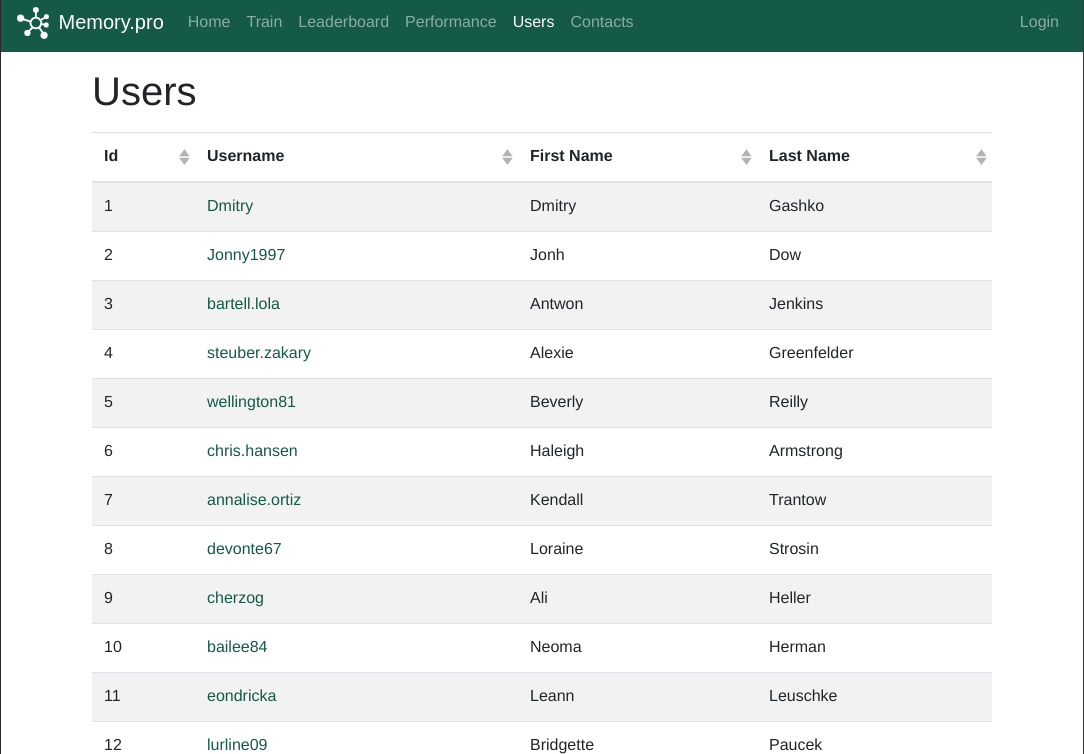
До речі, інші користувачі також можуть бачити ваш профіль та слідкувати за вашим прогресом, але вони не будуть бачити приватну інформацію. Наприклад, той же профіль інший користувач бачить так:

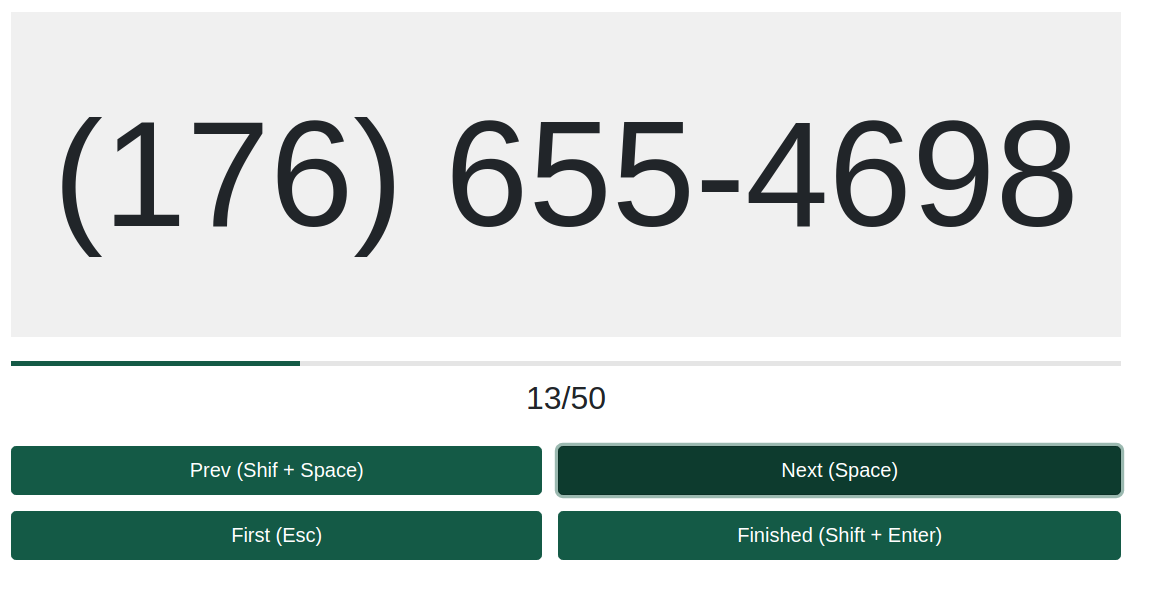


На сторінці Leaderboard можна бачити остані результати усіх користувачів:

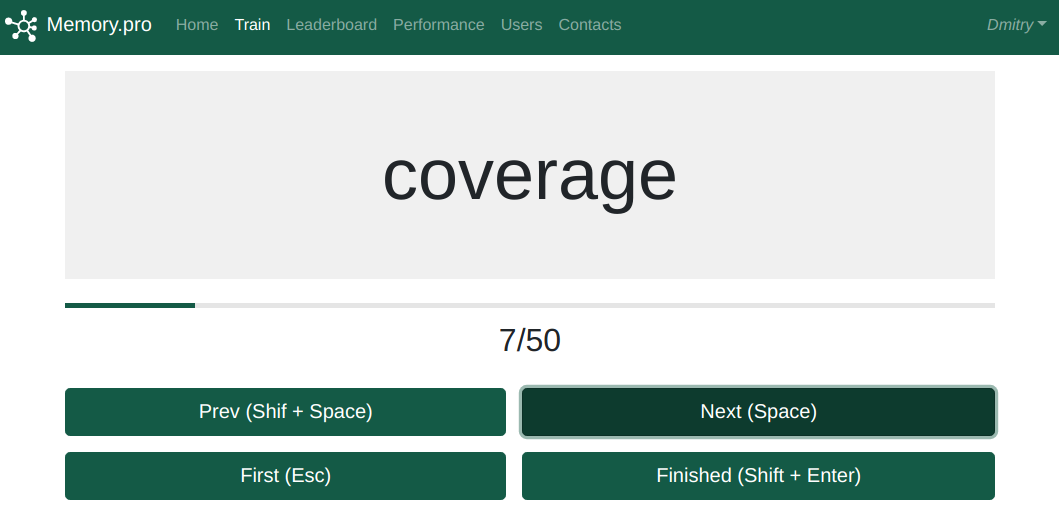


А на вкладці Users усіх зареєстрованих користувачів:



При встановлені параметрів тренування можна вибирати «Template» - шаблон показу елементів. Наприклад, якщо ввести «(XXX) XXX-XXXX» можна запам’ятовувати номери телефонів:

А запам’ятовування слів виглядає так:



# **Додаток Б. Код програмних модулів**

Частина 1. API.memory.pro

<?php

namespace App\Http\Controllers;

use App\Services\RandService;

class RandController extends Controller

{

    public function randWords(RandService $getRandService, int $len) {

        return $getRandService->getWords($len);

    }

    public function randImages(RandService $getRandService, int $len)

    {

        return $getRandService->getImages($len);

    }

}

<?php

namespace App\Services;

use App\Image;

use App\Word;

class RandService

{

   private int $maxLen = 50000;

   public function getWords(int $len)

   {

      $len = $this->formatLen($len);

      $words = Word::limit($len)->inRandomOrder()->get();

      if (count($words) === 0) {

         return abort('204', 'No words in out db');

      }

      return $this->fitToLen($words->toArray(), $len);

   }

   public function getImages(int $len)

   {

      $len = $this->formatLen($len);

      $images = Image::limit($len)->inRandomOrder()->get();

      if (count($images) === 0) {

         return abort('204', 'No images in out db');

      }

      return $this->fitToLen($images->toArray(), $len);

   }

   private function fitToLen(array $data, int $len)

   {

      $additionalLen = $len - count($data);

      $additional = [];

      for ($i = 0; $i < $additionalLen; $i++) {

         $additional[] = $data[array\_rand($data)];

      }

      return array\_merge($data, $additional);

   }

   private function formatLen($len)

   {

      return min(max($len, 0), $this->maxLen);

   }

}

<?php

namespace App\Http\Controllers;

use App\Http\Requests\SaveImagesResultRequest;

use App\Http\Requests\SaveResultRequest;

use App\Http\Requests\WordsResultsListRequest;

use App\ImagesResult;

use App\NumbersResult;

use App\Services\ResultsListService;

use App\Services\SaveResultService;

use App\WordsResult;

use Carbon\Carbon;

class ResultsController extends Controller

{

    public function saveNumbersResult(SaveResultRequest $req, SaveResultService $saveResultService)

    {

        return $saveResultService->saveNumbersResult([

            'start\_at' => new Carbon($req->start\_at),

            'preparation\_time' => $req->preparation\_time,

            'recall\_preparation\_time' => $req->recall\_preparation\_time,

            'recall\_time' => $req->recall\_time,

            'template' => $req->template,

            'items' => $req->items,

        ],  $req->user('api'));

    }

    public function saveWordsResult(SaveResultRequest $req, SaveResultService $saveResultService)

    {

        return $saveResultService->saveWordsResult([

            'start\_at' => new Carbon($req->start\_at),

            'preparation\_time' => $req->preparation\_time,

            'recall\_preparation\_time' => $req->recall\_preparation\_time,

            'recall\_time' => $req->recall\_time,

            'template' => $req->template,

            'items' => $req->items,

        ], $req->user('api'));

    }

    public function saveImagesResult(SaveImagesResultRequest $req, SaveResultService $saveResultService)

    {

        return $saveResultService->saveImagesResult([

            'start\_at' => new Carbon($req->start\_at),

            'preparation\_time' => $req->preparation\_time,

            'recall\_preparation\_time' => $req->recall\_preparation\_time,

            'recall\_time' => $req->recall\_time,

            'items\_size' => $req->items\_size,

            'items' => $req->items,

        ], $req->user('api'));

    }

    public function getWordsResultsList(WordsResultsListRequest $req, ResultsListService $resultsListService)

    {

        return $resultsListService->wordsList([

            'user\_id' => $req['user\_id'],

            'limit' => $req['limit'],

        ]);

    }

            $result->user()->associate($user);

            $user->save();

        }

        $result->save();

        foreach ($resultData['items'] as $itemData) {

            $item = new WordsResultItem();

            $item->time = $itemData['time'];

            $item->result()->associate($result);

            $item->save();

            foreach ($itemData['data'] as $dataData) {

                $data = new WordsResultData();

                $data->correct\_id = $dataData['correct'];

                $data->actual = $dataData['actual'];

                $data->item()->associate($item);

                $data->save();

            }

        }

        DB::commit();

        return $result->loadMissing('items.data.correct');

    }

    /\*\*

     \* @param array {

     \*   start\_at: Carbon,

     \*   recall\_time: int,

     \*   items\_size: string,

     \*   items: {

     \*       time: number,

     \*       data: {

     \*          correct: number,

     \*          actual: number,

     \*       }[],

     \*   }[],

     \* } $resultData

     \*/

    public function saveImagesResult(array $resultData, User $user = null)

    {

        DB::beginTransaction();

        $grade = $this->gradeService->gradeImagesResult($resultData);

        $result = new ImagesResult();

        $result->start\_at = $resultData['start\_at'];

        $result->preparation\_time = $resultData['preparation\_time'];

        $result->recall\_preparation\_time = $resultData['recall\_preparation\_time'];

        $result->recall\_time = $resultData['recall\_time'];

        $result->items\_size = $resultData['items\_size'];

        $result->grade = $grade['grade'];

        if ($user) {

            $result->user()->associate($user);

            $user->save();

        }

        $result->save();

        foreach ($resultData['items'] as $itemData) {

            $item = new ImagesResultItem();

            $item->time = $itemData['time'];

            $item->result()->associate($result);

            $item->save();

            foreach ($itemData['data'] as $dataData) {

                $data = new ImagesResultData();

                $data->correct\_id = $dataData['correct'];

                $data->actual\_id = $dataData['actual'];

                $data->item()->associate($item);

                $data->save();

            }

        }

        DB::commit();

        return $result->loadMissing(['items.data.correct', 'items.data.actual']);

    }

}

<?php

namespace App\Services;

use App\ImagesResult;

use App\NumbersResult;

use App\WordsResult;

class ResultsListService

{

    public function numbersList(array $options)

    {

        $res = NumbersResult::query();

        $res = $res->orderBy('id', 'desc');

        if (isset($options['user\_id'])) {

            $res = $res->where('user\_id', '=', $options['user\_id']);

        }

        if (isset($options['limit'])) {

            $res = $res->limit($options['limit']);

        }

        return $res->get()->loadMissing('user');

    }

     \* @param array $resultData (as in SaveResultService::save\*Result)

     \* @return int

     \*/

    public function gradeImagesResult(array $resultData)

    {

        $total = 0;

        $correct = 0;

        $time = 0;

        foreach ($resultData['items'] as $itemData) {

            $total += count($itemData['data']);

            $time += $itemData['time'];

            foreach ($itemData['data'] as $dataData) {

                if ($dataData['correct'] === $dataData['actual']) {

                    $correct++;

                }

            }

        }

        if ($correct / $total < 1 - $this->MAX\_IMAGES\_ERROR) {

            $grade = 0;

        } else {

            $grade =  $this->calcGradeCommonly($total, $correct, $time);

        }

        return [

            "grade" => $grade,

            "total" => $total,

            "correct" => $correct,

            "time" => $time,

        ];

    }

    /\*\*

     \* Calculate grade using common algorithm.

     \* Uses in trainings witch doesn't requite separate grade algorithm

     \*/

    private function calcGradeCommonly(int $total, int $correct, int $time)

    {

        $grade = ($correct + ($total - $time / 3000)) / 5;;

        return $this->prepareGrade($grade);

    }

    private function prepareGrade($grade)

    {

        return round(min(max($grade, 0), $this->MAX\_GRADE));

    }

}

<?php

namespace App\Http\Controllers;

use App\User;

use Illuminate\Http\Request;

class UsersController extends Controller

{

    public function getUser(Request $req, $username)

    {

        /\*\* @var User \*/

        $user = User::where('username', '=', $username)->firstOrFail();

        $curUser = $req->user('api');

        if ($curUser && $user->username == $curUser->username) {

            return $user;

        }

        return [

            'id' => $user->id,

            'username' => $user->username,

            'first\_name' => $user->first\_name,

            'last\_name' => $user->last\_name,

            'avatar' => $user->avatar,

        ];

    }

    public function getUsersList()

    {

        return User::all()->map(function ($user) {

            return [

                'id' => $user->id,

                'username' => $user->username,

                'first\_name' => $user->first\_name,

                'last\_name' => $user->last\_name,

                'avatar' => $user->avatar,

            ];

        });

    }

}

<?php

namespace App\Services;

use App\User;

use Illuminate\Http\Request;

use Illuminate\Validation\Rule;

class UserService

{

   /\*\*

    \* Update user info

    \*

    \* @param User $user

    \* @param array $params = {

    \*   username: string,

    \*   email?: string,

    \*   first\_name?: string,

    \*   last\_name?: string,

    \*   birth?: Carbon,

    \* } r

    \*

    \* @return User

    \*/

   public function updateUserData(User $user, Request $r)

   {

      $this->validateUpdateRequest($r);

      if (isset($r->username)) $user->username = $r->username;

      if (isset($r->email)) $user->email = $r->email;

      if (isset($r->first\_name)) $user->first\_name = $r->first\_name;

      if (isset($r->last\_name)) $user->last\_name = $r->last\_name;

      if (isset($r->birth)) $user->birth = $r->birth;

      $user->save();

      return $user;

   }

   public function updateAvatar(User $user, Request $request)

   {

      $this->validateAvatarUpdateRequest($request);

      $user->uploadImage($request->avatar, 'avatar');

      $user->save();

      return $user;

   }

   private function validateAvatarUpdateRequest(Request $request)

   {

      $request->validate([

         'avatar' => 'required|mimes:jpeg,png,gif,webp|max:' . 20 \* 1024,

      ]);

   }

   private function validateUpdateRequest(Request $request)

   {

      $request->validate([

         'username' => 'nullable|string|min:2|max:255',

         'email' => 'nullable|string|email|max:255',

         'first\_name' => 'nullable|string|min:2|max:255',

         'last\_name' => 'nullable|string|min:2|max:255',

         'birth' => 'nullable|date',

      ]);

   }

}

Частина 2. frontend.memory.pro

<template>

<div>

<Setup v-if="step === 'setup'" @done="setupDone" template="X" />

<Preparation

v-else-if="step === 'preparation'"

:time="preparation"

@done="step = 'memorization'"

title="Memorization starts in:"

/>

<Memorization

v-else-if="step === 'memorization'"

:autoNext="autoNext"

:data="data"

@done="memorizationDone"

/>

<Preparation

v-else-if="step === 'recallPreparation'"

:time="recallPreparation"

@done="step = 'recall'"

title="Recall starts in:"

class="step"

/>

<Recall :len="len" v-else-if="step === 'recall'" @done="recallDone" />

</div>

</template>

<script>

import { mapActions } from 'vuex';

import chunk from 'lodash/chunk';

import Setup from '@@/train/steps/Setup';

import Preparation from '@@/train/steps/Preparation';

import Memorization from '@@/train/steps/Memorization';

import Recall from '@@/train/steps/Recall';

import splitAndFormatByTemplate from '@/assets/scripts/splitByTemplate';

export default {

components: { Setup, Preparation, Memorization, Recall },

data: () => ({

data: [],

ids: [],

times: [],

answers: [],

step: 'setup',

preparation: 0,

recallPreparation: 0,

recallTime: 0,

autoNext: 2000,

template: '',

len: 0,

itemSize: 0,

startAt: null

}),

created() {

this.startAt = new Date();

},

methods: {

async setupDone(data) {

this.len = +data.len;

this.template = data.template;

this.autoNext = data.autoNext;

this.preparation = data.preparation;

this.recallPreparation = data.recallPreparation;

await this.fetchData();

this.step = this.preparation > 0 ? 'preparation' : 'memorization';

},

memorizationDone(times) {

this.step = this.recallPreparation > 0 ? 'recallPreparation' : 'recall';

this.times = times;

},

recallDone(answers, time) {

this.answers = chunk(answers, this.itemSize);

this.recallTime = time;

this.done();

},

async fetchData() {

const rawData = await this.rand(this.len);

const data = (window.data = rawData.map(w => w.value));

const ids = rawData.map(w => w.id);

this.itemSize = this.template.split('X').length - 1;

this.ids = chunk(ids, this.itemSize);

this.data = splitAndFormatByTemplate(this.template, data);

},

async done() {

const trainResult = {

start\_at: this.startAt,

preparation\_time: this.preparation,

recall\_preparation\_time: this.recallPreparation,

recall\_time: this.recallTime,

template: this.template,

items: this.getTrainigDataResult()

};

const { id } = await this.save(trainResult);

this.$router.push(`/results/words/${id}`);

},

getTrainigDataResult() {

return this.ids.map((ids, itemIndex) => {

const r = this.answers[itemIndex];

return {

time: this.times[itemIndex],

data: ids.map((id, dataIndex) => ({

correct: id,

actual: r ? r[dataIndex] : -1

}))

};

});

},

...mapActions({

rand: 'trainingData/randWords',

save: 'results/saveWordsResult'

})

}

};

</script>

<style scoped>

.step {

flex: 1;

}

</style>

<template>

<div>

<Setup v-if="step === 'setup'" @done="setupDone" template="XX" />

<Preparation

v-else-if="step === 'preparation'"

:time="preparation"

@done="step = 'memorization'"

title="Memorization starts in:"

/>

<Memorization

v-else-if="step === 'memorization'"

:autoNext="autoNext"

:data="data"

@done="memorizationDone"

>

<template #item="{item, even}">

<TextItem :value="item" :even="even" size="150" />

</template>

</Memorization>

<Preparation

v-else-if="step === 'recallPreparation'"

:time="recallPreparation"

@done="step = 'recall'"

title="Recall starts in:"

class="step"

/>

<Recall :len="recallLen" v-else-if="step === 'recall'" @done="recallDone" />

</div>

</template>

<script>

import { mapActions } from 'vuex';

import chunk from 'lodash/chunk';

import Setup from '@@/train/steps/Setup';

import Preparation from '@@/train/steps/Preparation';

import Memorization from '@@/train/steps/Memorization';

import Recall from '@@/train/steps/Recall';

import TextItem from '@@/train/TextItem';

import splitAndFormatByTemplate from '@/assets/scripts/splitByTemplate';

export default {

components: { Setup, Preparation, Memorization, Recall, TextItem },

data: () => ({

data: [],

ids: [],

times: [],

answers: [],

step: 'setup',

preparation: 0,

recallPreparation: 0,

recallTime: 0,

autoNext: 2000,

template: '',

len: 0,

itemSize: 0,

startAt: null

}),

computed: {

recallLen() {

return Math.ceil(this.len / this.itemSize);

}

},

created() {

this.startAt = new Date();

},

methods: {

async setupDone(data) {

this.len = +data.len;

this.template = data.template;

this.autoNext = data.autoNext;

this.preparation = data.preparation;

this.recallPreparation = data.recallPreparation;

await this.fetchData();

this.step = this.preparation > 0 ? 'preparation' : 'memorization';

},

memorizationDone(times) {

this.step = this.recallPreparation > 0 ? 'recallPreparation' : 'recall';

this.times = times;

},

recallDone(answers, time) {

this.answers = chunk(

answers.flatMap(n => n.split('')),

this.itemSize

);

this.recallTime = time;

this.done();

},

async fetchData() {

const data = (window.data = await this.rand(this.len));

const ids = data.map(n => n);

this.itemSize = this.template.split('X').length - 1;

this.ids = chunk(ids, this.itemSize);

this.data = splitAndFormatByTemplate(this.template, data);

},

async done() {

const trainResult = {

start\_at: this.startAt,

preparation\_time: this.preparation,

recall\_preparation\_time: this.recallPreparation,

recall\_time: this.recallTime,

template: this.template,

items: this.getTrainigDataResult()

};

const { id } = await this.save(trainResult);

this.$router.push(`/results/words/${id}`);

},

getTrainigDataResult() {

return this.ids.map((ids, itemIndex) => {

const r = this.answers[itemIndex];

return {

time: this.times[itemIndex],

data: ids.map((id, dataIndex) => ({

correct: id,

actual: r ? r[dataIndex] : -1

}))

};

});

},

...mapActions({

rand: 'trainingData/randNumbers',

save: 'results/saveNumbersResult'

})

}

};

</script>

<style scoped>

.step {

flex: 1;

}

</style>

import {

  processRequestError,

  removeAuthHeader

} from '~/assets/scripts/utils.ts';

export const state = () => ({});

export const actions = {

  async getNumbersResult(\_, id) {

    try {

      return await this.$axios.$get(`/results/numbers/${id}`, {

        transformRequest: [removeAuthHeader]

      });

    } catch (e) {

      processRequestError(e);

    }

  },

  async getWordsResult(\_, id) {

    try {

      return await this.$axios.$get(`/results/words/${id}`, {

        transformRequest: [removeAuthHeader]

      });

    } catch (e) {

      processRequestError(e);

    }

  },

  async getImagesResult(\_, id) {

    try {

      return await this.$axios.$get(`/results/images/${id}`, {

        transformRequest: [removeAuthHeader]

      });

    } catch (e) {

      processRequestError(e);

    }

  },

  async saveNumbersResult(\_, data) {

    try {

      return await this.$axios.$put(`/results/numbers`, data);

    } catch (e) {

      processRequestError(e);

    }

  },

  async saveWordsResult(\_, data) {

    try {

      return await this.$axios.$put(`/results/words`, data);

    } catch (e) {

      processRequestError(e);

    }

  },

  async saveImagesResult(\_, data) {

    try {

      return await this.$axios.$put(`/results/images`, data);

    } catch (e) {

      processRequestError(e);

    }

  },

  async userNumbersResults(\_, data) {

    try {

      return await this.$axios.$post(`/results/numbers`, data);

    } catch (e) {

      processRequestError(e);

    }

  },

  async userWordsResults(\_, data) {

    try {

      return await this.$axios.$post(`/results/words`, data);

    } catch (e) {

      processRequestError(e);

    }

  },

  async userImagesResults(\_, data) {

    try {

      return await this.$axios.$post(`/results/images`, data);

    } catch (e) {

      processRequestError(e);

    }

  }

};

<template>

  <div>

    <h1 class="mb-3">Training Setup</h1>

    <b-form @submit.prevent="submit" class="form">

      <b-form-group label="Items to Memorize" label-for="setup-let">

        <b-form-input

          id="setup-len"

          v-model="data.len"

          type="number"

          required

          min="10"

          max="100000"

        ></b-form-input>

      </b-form-group>

      <b-form-group label="template" label-for="setup-template">

        <b-form-input

          id="setup-template"

          v-model="data.template"

          required

        ></b-form-input>

      </b-form-group>

      <b-form-group

        label="Auto next:"

        label-for="setup-auto-next"

        description="Enter 0 to disabled"

      >

        <b-form-input

          id="setup-auto-next"

          v-model="data.autoNext"

          type="number"

          step="100"

          min="0"

          required

        ></b-form-input>

      </b-form-group>

      <b-form-group

        label="Preparation time:"

        label-for="setup-preparation"

        description="Enter 0 to disabled"

      >

        <b-form-input

          id="setup-preparation"

          v-model="data.preparation"

          type="number"

          min="0"

          required

        ></b-form-input>

      </b-form-group>

      <b-form-group

        label="Recall Preparation Time:"

        label-for="setup-recall-preparation"

        description="Enter 0 to disabled"

      >

        <b-form-input

          id="setup-recall-preparation"

          v-model="data.recallPreparation"

          type="number"

          min="0"

          required

        ></b-form-input>

      </b-form-group>

      <b-button type="submit" variant="primary">Continue</b-button>

    </b-form>

  </div>

</template>

<script>

export default {

  props: ['template'],

  data() {

    return {

      data: {

        len: 50,

        template: '',

        autoNext: 3000,

        preparation: 5,

        recallPreparation: 5

      }

    };

  },

  created() {

    if (this.template) {

      this.data.template = this.template;

    }

  },

  methods: {

    submit() {

      this.$emit('done', this.data);

    }

  }

};

</script>

<style lang="scss" scoped>

.form {

  max-width: 500px;

}

</style>

<template>

  <div>

    <slot :item="item" :even="even" name="item">

      <TextItem :value="item" :even="even" />

    </slot>

    <Status :cur="index + 1" :total="data.length" class="status" />

    <Controls

      ref="controls"

      @next="next"

      @prev="prev"

      @first="first"

      @finish="done"

      :auto-next="autoNext"

      class="controls"

    />

  </div>

</template>

<script>

import TextItem from '@@/train/TextItem';

import Controls from '@@/train/Controls';

import Status from '@@/train/Status';

export default {

  components: { TextItem, Controls, Status },

  props: ['autoNext', 'data'],

  data: () => ({

    index: 0,

    startedAt: null,

    times: []

  }),

  computed: {

    item() {

      return this.data[this.index];

    },

    even() {

      return this.index % 2 === 0;

    }

  },

  mounted() {

    this.showAt = Date.now();

    this.times = this.data.map(\_ => 0);

  },

  methods: {

    done() {

      this.$emit('done', this.times);

    },

    prev() {

      this.setIndex(this.index - 1);

    },

    next() {

      this.setIndex(this.index + 1);

    },

    first() {

      this.setIndex(0);

    },

    finish() {

      // Last index + 1 (go outside)

      this.setIndex(this.data.length);

    },

    setIndex(value) {

      if (value < 0) value = 0;

      if (value === this.index) {

        return;

      }

      const now = Date.now();

      const diff = now - this.showAt;

      this.times[this.index] += diff;

      if (value > this.data.length - 1) {

        value = this.data.length - 1;

        this.done();

      }

      this.showAt = now;

      this.index = value;

      this.$refs.controls.restartAutoNext();

    }

  }

};

</script>

.<style lang="scss" scoped>

.controls,

.status {

  margin-top: 24px;

}

</style>

<template>

  <div class="mt-2">

    <div class="items-container">

      <RecallItem ref="items" v-for="n in len" :key="n" :index="n" />

    </div>

    <div class="controls mt-5 mb-5">

      <b-button @click="done" variant="primary" size="lg">Continue</b-button>

    </div>

  </div>

</template>

<script>

import RecallItem from '@@/train/RecallItem';

export default {

  components: { RecallItem },

  props: ['len'],

  data: () => ({

    startAt: 0

  }),

  computed: {

    answers() {

      return this.$refs.items.map(item => item.value);

    }

  },

  created() {

    this.startAt = Date.now();

  },

  methods: {

    done() {

      const time = (Date.now() - this.startAt) / 1000;

      this.$emit('done', this.answers, time);

    }

  }

};

</script>

<style lang="scss" scoped></style>

<style lang="scss" scoped>

.controls {

  text-align: center;

}

.items-container {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));

  grid-auto-flow: dense;

  gap: 1em;

}

</style>