**Анотація:** в курсовій роботі розроблений сервіс для тренування пам’яті «Memory.pro». Для розробки програмного забезпечення було використано мову програмування Java, бібліотеку JavaFX (з використанням FXML та SASS), JPA фреймворк EclipseLink та СУБД MYSQL.

**Аннотация:** в курсовой работе разработан сервис для тренировки памяти «Memory.pro». Для разработки программного продукта был использован язык программирования Java, библиотека JavaFX (с использование FXML и SASS), JPA фреймворк EclipseLink и СУБД MYSQL.

**Annotation:** a software product for memory training named «Memory.pro» is developed during the coursework. The Java programing language, the library JavaFX (with FXML and SASS), the Java Persistence API (by EclipseLink) and MYSQL DBMS have been used.

Зміст

[Вступ 4](#_Toc9889486)

[1. Аналіз та опис предметної області 5](#_Toc9889487)

[1.1. Загальний опис предметної області 5](#_Toc9889488)

[1.2. Аналіз вже існуючого програмного забезпечення 5](#_Toc9889489)

[1.3. Дослідження та опис предметної області 6](#_Toc9889490)

[1.4. Вибір засобів розробки 8](#_Toc9889491)

[2. Проектування 9](#_Toc9889492)

[2.1. Ескізний проект 9](#_Toc9889493)

[2.1.1. Схема взаємодії модулів 9](#_Toc9889494)

[2.1.2. Діаграма варіантів використання 10](#_Toc9889499)

[2.2. Технічний проект 11](#_Toc9889500)

[2.2.1. Контекстна діаграми системи 11](#_Toc9889501)

[2.2.2. DFD 1 рівня 11](#_Toc9889502)

[2.2.3. DFD 2 рівня (Статистика) 12](#_Toc9889503)

[2.2.4. Діаграма сутність-зв’язок (ERD) 13](#_Toc9889504)

[2.2.5. Логічна модель системи 14](#_Toc9889505)

[2.2.6. Діаграма класів 15](#_Toc9889506)

[2.2.10 Діаграма послідовностей 18](#_Toc9889507)

[Додаток А. Технічне завдання 20](#_Toc9889508)

[Додаток Б. Інструкція користувача 26](#_Toc9889509)

[Додаток В. Код програмних модулів 36](#_Toc9889510)

# **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 та опис предметної області**
   1. **Загальний опис предметної області**

Сервіс «Memory.pro» призначений для тренування та розвитку пам’яті. Основна функція сервісу – 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.

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

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.

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. 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.
3. 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.
4. 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. **Вибір засобів розробки**

Для розробки програмного забезпечення у цій роботі використано мову програмування Java. Для розробки графічного інтерфейсу користувача використано бібліотеку JavaFX з використанням FXML та CSS (SASS) а також Java Persistence API framework EclipseLink та СУБД MYSQL.

1. **Проектування**
   1. **Ескізний проект** 
      1. **Схема взаємодії модулів**

Взаємодію модулів системи можна представити в вигляді наступної діаграми:



2. 1. 2. **Діаграма варіантів використання**

За допомогою діаграми варіантів використання окреслимо користувачів системи та її функціонал:



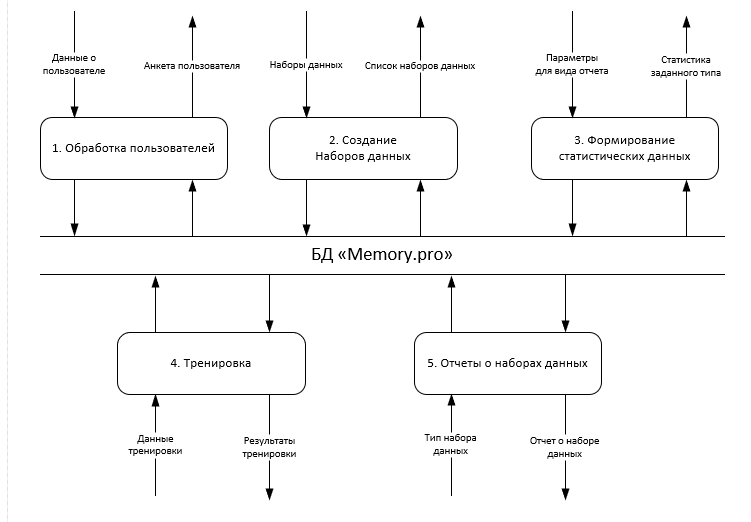
* 1. **Технічний проект**

В результаті опису предметної області та постановки задачі отримано вимоги до БД системи, які можна зобразити такими діаграмами як: DFD, ERD, логічна модель системи.

* + 1. **Контекстна діаграми системи**



* + 1. **DFD 1 рівня**



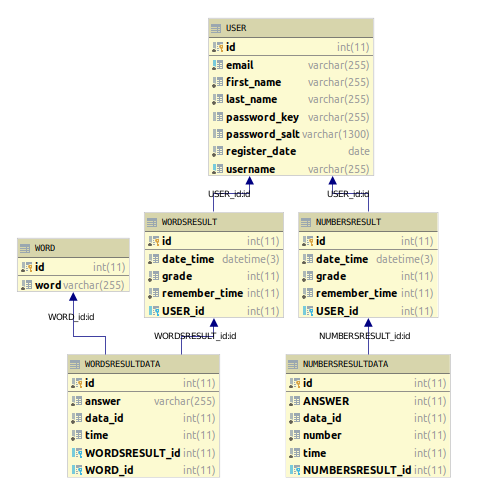
* + 1. **DFD 2 рівня (Статистика)**



* + 1. **Діаграма сутність-зв’язок (ERD)**



* + 1. **Логічна модель системи**



Таблиця User – користувач системи. Усі атрибути зрозумілі з назви. Атрибути password\_key та password\_salt необхідні для авторизації користувача (сам пароль ніде не зберігається, його знає лише користувач). Тобто зберігання паролю користувача основане на механізмі хешування.

Таблиці WordsResult та NumbersResult призначенні для зберігання результатів тренування.

date\_time – час початку тренування

grade – оцінка тренування

remember\_time – час згадування в секундах

Таблиці WordsResultData NumbersResultData зберігають дані тренування. Відповідно слова, та числа. А також час на запам’ятовування кожної одиниці даних у мілісекундах та відповідь користувача (у WordsResultData зберігається як текст, так як, користувач може ввести слово, якого немає в базі даних).

* + 1. **Діаграма класів**

На діаграмі класів зображено взаємодію модулів системи.

У системі усі класи поділяються на такі види:

1. Component – певна візуально та логічно самостійна сутність. Усі компоненті складаються с контролеру (java класу, FXML, та SASS для стилів). Компоненти реалізують лише візуальну складову. Вони не повинні реалізовувати занадто багато логіки, не пов’язаної напряму з відображенням. Назви усіх класів цього виду закінчуються на Component
2. Screen – наслідує від Component. Представляє сутність вікна (сторінки). Screen класи можуть містити у собі компоненти. Один вид компоненту може міститись в різних Screen класах. Назви усіх класів цього виду закінчуються на Screen.
3. Service – класи, що реалізують логічно самостійну сутність. Дані класи реалізують певну задачу (наприклад авторизацію) та надають зручний інтерфейс для роботи з ними. Вони нічого не знають про те, як результати їх роботи будуть представлені користувачу. Дані класи працюють тільки з даними. Назви усіх класів цього виду закінчуються на Service.
4. Entity – описання сутностей для Java Persictence API
5. Dao – класи, що узагальнюють роботу з Entity класами та базою даних в цілому. Тільки в Dao-класах дозволяється працювати з базою даних. Назви усіх класів цього виду закінчуються на Dao.

Кожен вид класів при необхідності може використовувати додаткові класи

Також є два допоміжних класів:

* 1. Common – клас, що дозволяє Screen- та Component-класам спілкуватись між собою, та передавати повідомлення до Bootstrap.
  2. Alerts – надає зручний інтерфейс для виведення повідомлень користувачам (повідомлення про помилки, попередження, інформаційні повідомлення тощо). Даний клас міститься (композиція) у Component і таким чином доступний з усіх Component та Screen класів.

Component класи:

1. HeaderComponent – header, що відображається на усіх сторінках
2. LoginComponent – авторизація користувачів
3. SignupComponent – реєстрація нових користувачів
4. TrainingSetupComponent – вибір параметрів тренування
5. TrainingMemorizeComponent – етап запам’ятовування
6. TrainingRememberComponent – етап згадування
7. ResultComponent – відображає результат тренування за переданими типом тренування на id результату.

Screen класи:

1. Bootstrap – точка входу, головний контролер
2. AuthScreen – сторінка реєстрації та авторизації
3. IndexScreen – головна сторінка
4. ProfileScreen – профіль користувача
5. StatisticScreen – сторінка статистики
6. TrainingScreen – сторінка тренування

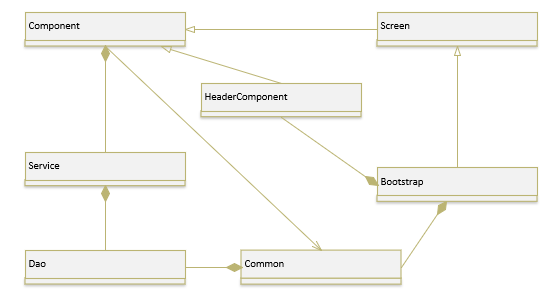
Service класи

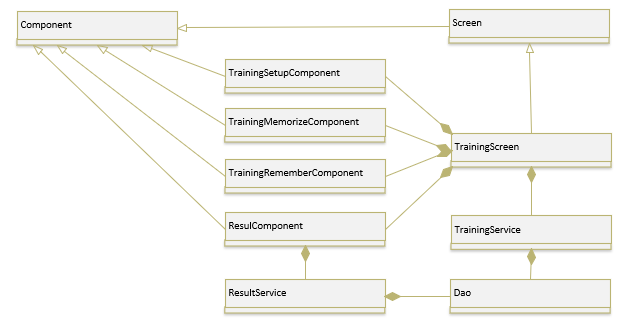
1. AuthService – сервіс авторизації на реєстрації
2. TrainingService – сервіс тренувань (надає дані для тренування, обробляє та зберігає результати.
3. ResultService – сервіс для роботи з результатами тренування (за переданим типом тренування та id результата надає оброблений і готовий для подальшого використання результат тренування (в поверненому результаті вже містяться всі дані такі як – час запам’ятовування, мінімальний та максимальний час на запам’ятовування одного елементу тощо).

Entity та Dao класи (відповідають сутностям на ER діаграмі

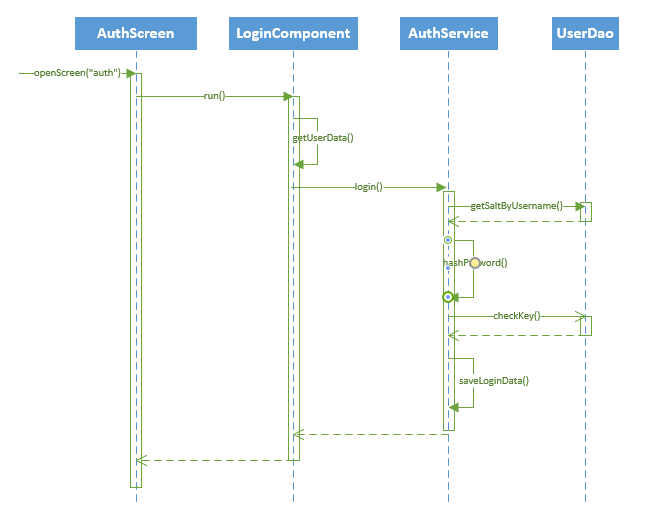
1. User та UserDao
2. NumbersResult та NumbersResultDao
3. NumbersResultData та NumbersResultDataDao
4. WordsResult та WordsResult
5. WordsResultData та WordsResultDataDao
6. Word та WordDao

Так, як сервіс складається з великої кількості класів і при цьому все вони діляться всього на кілька типів, то на діаграмі класів немає сенсу зображати всі класи. Тому буде представлена загальна діаграма класів та діаграма класів відносно TrainingScreen.





* + 1. **Діаграма послідовностей** 
       1. **Діаграма послідовностей процесу «авторизація»**



* + - 1. **Діаграма послідовностей процесу «тренування»**



# **Додаток А. Технічне завдання**

1. Загальні відомості
   1. Найменування системи
      1. Повна назва системи

Сервіс для тренування пам’яті «Memory.pro».

* + 1. Коротка назва

Memory.pro

* 1. Підстави для розробки

Підставою для розробки програмного забезпечення (ПО) є договір між Беркунським Євгенієм Юрійовичем (далі Замовник) та Гашко Дмитро Андрійовичем (далі Виконавець) від 20.05.2019.

* 1. Планові строки початку та закінчення робіт

Початком роботи над програмним забезпеченням слід вважати 21.05.2019, закінчення роботи є 10.06.2019.

* 1. Порядок оформлення та пред’явлення замовнику результатів робіт

Роботи зі створення ПЗ здаються Замовнику в дату закінчення роботи. Виконавець надає замовнику звітні документи, склад яких зазначено у договорі та демонструє результати робіт.

1. Призначення та цілі створення системи
   1. Призначення системи

Сервіс «Memory.pro» призначений для розвитку та тренування пам’яті, аналізу свого прогресу, та прогресу інших учасників.

У рамках проекту будуть реалізовані такі типи тренувань:

1. Запам’ятовування послідовності двозначних чисел
2. Запам’ятовування послідовності слів.
   1. Цілі створення системи

Система створюється з наступними цілями:

1. Надати користувачу інструменти для тренування пам’яті
2. Реалізувати основні види тренувань (числа та слова).
3. Реєстрація/авторизація користувачів.
4. Кабінет користувача
5. Збереження результатів тренувань користувачів
6. Перегляд статистичних даних
7. Короткий опис сервісу

Сервіс «Memory.pro» призначений для тренування та розвитку пам’яті. Основна функція сервісу – 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.

1. Вимоги до системи
   1. Вимоги до системи в цілому
      1. Вимоги до структури та функціонування системи

Сервіс повинен мати систему авторизації, де користувач може увійти у систему, або зареєструватись; особистий кабінет користувача; систему тренувань з запам’ятовування чисел та слів. Усі дані користувачів та результатів тренувань повинні зберігатись у базі даних.

* + 1. Вимоги до надійності
       1. Склад показників надійності для системи в цілому

Рівень надійності повинен досягатись узгодженим застосуванням організаційних, організаційно-технічних заходів та програмно-апаратних засобів. Надійність повинна забезпечуватись за рахунок:

1. Застосування технічних засобів, програмного забезпечення, що відповідають класу вирішуваних задач.
2. Своєчасне виконання процесів адміністрування та технічного обслуговування програмно-апаратних засобів.
   * + 1. Перелік аварійних ситуацій, за яких регламентуються вимоги до надійності

Під аварійною ситуацією слід розуміти аварійне завершення роботи складових сервісу: безпосередньо програми, бази даних тощо.

При роботі системи можливі наступні аварійні ситуації, що впливають на надійність роботи системи:

1. Збій у роботі електропостачання серверу БД
2. Перевантаження серверу БД
3. Помилки роботи ПЗ, що не були виявлені про тестуванні та випробуванні системи;
   * 1. Вимоги до ергономіки до технічної естетики

Інтерфейс системи має бути звучним та інтуїтивно зрозумілим. Усі сторінки та компоненти повинні бути виконанні в єдиному стилі. Палітра основних кольорів має складатись не більше ніж з 4 кольорів.

Сервіс обов’язково має бути доступний з клавіатури.

Сервіс має бути на англійській мові.

Усі помилки та виняткові ситуації мають супроводжуватись відповідними повідомленнями.

* + - 1. Вимоги до захисту інформації від несанкціонованого доступу

Забезпечення інформаційне безпеки сервісу має відповідати таким вимогам:

* Для доступу до сервісу користувачу необхідно увійти до системи.
* Якщо користувач ще не зареєстрований у системі, він повинен мати можливість зареєструватись
* Пароль, введений користувачем не зберігається, і навіть не відправляється на сервер бази даних. Авторизація реалізується на основі механізму хешування
* В базі даних має зберігатись ключ та «сіль» (що генерується для кожного користувача) паролю
* Неавторизовані користувачі не повинні мати доступ до сервісу
  1. Вимоги до видів забезпечення
     + 1. Вимоги щодо застосування систем управління баз даних (СУБД)

Всі дані сервісу (дані користувачів, результати тренувань) повинні зберігатись у базі даних на основі СУБД MYSQL.

* + - 1. Вимоги до структури процесу збору, обробки, передачі даних у системі та представленню даних

Процес збору, обробки і передачі даних в системі визначається регламентом процесів збору, перетворення і завантаження даних, що розробляється в етапі «Проектування. Розробка ескізного проекту. Розробка технічного проекту ».

* + - 1. Вимоги до захисту даних від руйнування при аваріях та сбоях електропостачання системи

Необхідно регулярно робити backup даних сервісу

Розробка програмного забезпечення сервісу повинна супроводжуватись веденням GIT-репозиторію, який також має розміщуватись на GitLab.

* + 1. Вимоги до ПЗ

До забезпечення якості ПЗ ставляться такі вимоги:

Сервіс має працювати стабільно, не закриватися при виникненні помилок без попередження користувача.

Сервіс має мати достатню швидкість роботи, щоб користувачі могли вільно ним користуватись

* + 1. Вимоги до організаційного забезпечення

Основними користувачами системи Користувач. До захисту від помилкових дій користувача ставляться такі вимоги:

1. Усі дії, що у разі випадкового виконання можуть завдати шкоди користувачу мають супроводжуватись підтвердженням.
2. для зниження помилкових дій користувачів має бути розроблена інструкція користувача.
   * 1. Вимоги до методичного забезпечення

Порядок роботи з системою викладено у окремому документі – «Інструкції користувача».

1. Склад робіт по створенню системи

Роботи зі створення системи виконуються в три етапи:

* Проектування. Розробка ескізного проекту. Розробка технічного проекту
* Розробка робочої документації.
* Введення в дію.

1. Порядок контролю та прийому системи
   1. Об’єм та види випробувань системи

Система піддається випробуванням наступних видів:

1. Попередні випробування.
2. Дослідна експлуатація.
3. Приймальні випробування.
4. Вимоги до документування

|  |  |
| --- | --- |
| Проектування. Розробка ескізного проекта. Розробка технічного проекта. | Відомість ескізного проекта |
| Пояснювальна записка ескізного проекта |
| Відомість ескізного проекта |
| Пояснювальна записка до технічного проекта |
| Схема функціональної структури |
| Відомість експлуатаціоних документів |
| Паспорт |
| Загальне призначення системи |
| Технологічна інструкція |
| Інструкція користувача |
| Розробка технічної документації. Адаптація програм | Опис технологічного процесу обробки даних |
| Інструкція по формуванню та введенню бази даних |
| Склад вхідних даних |
| Каталог бази даних |
| Програма та методика випробувань |
| Специфікація |
|  | Опис програм |
| Текст програм |
| Введення в дію | Акт прийому в дослідну експлуатацію |
| Протокол випробувань |
| Акт прийому системи в експлуатацію |
| Акт закінчення робіт |
|  |

# **Додаток Б. Інструкція користувача**

Дана інструкція призначена для ознайомлення з функціями сервісу «Memory.pro».

Запустивши програму користувачеві пропонується увійти до систему (звичайно, якщо він цього ще не авторизований).

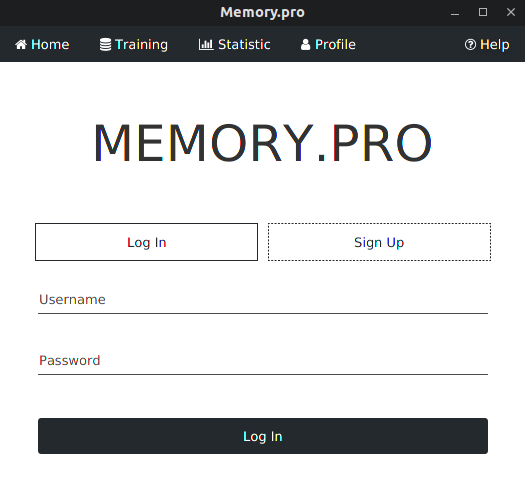


Рис. 1. Форма авторизації

На даний момент користувач не може отримати доступу ні до яких функцій сервісу крім пункту меню «Help». Тому є лише два варіанти:

* Увійти у систему, якщо користувач вже зареєстрований
* Зареєструватися, якщо ще ні.

Вибравши вкладку «Sign Up» користувач переходить на форму реєстрації (рис. 2).

При авторизації користувачу необхідно ввести такі дані як:

* Username
* Пароль

При авторизації:

* Ім’я
* Прізвище
* Username
* Email
* Пароль

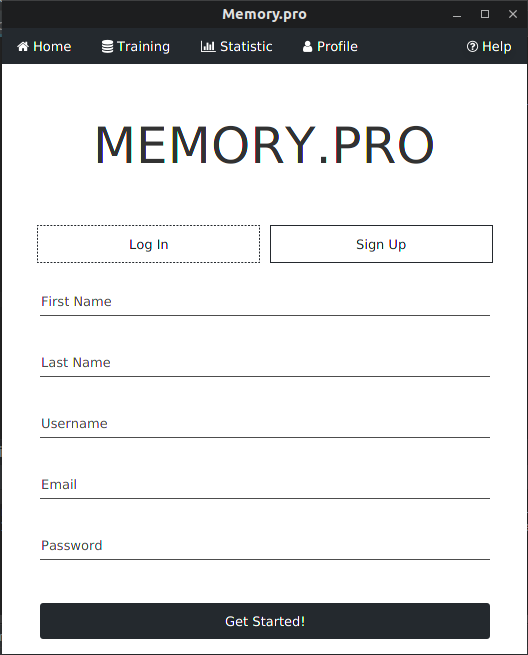


Рис. 2. Форма реєстрації

Якщо введені некоректні дані, відповідні поля будуть підсвічені червоним (рис. 3).

Авторизація та реєстрація сервісу «Memory.pro» реалізована, так, що пароль користувача ніде не зберігається. Це можливо завдяки механізму хешування. Тому пароль знає лише користувач.

Якщо, при авторизації були введені неправильні нікнейм та пароль а також, якщо при реєстрації будуть введені дані вже існуючого користувача то буде показане відповідне попередження (рис. 4).

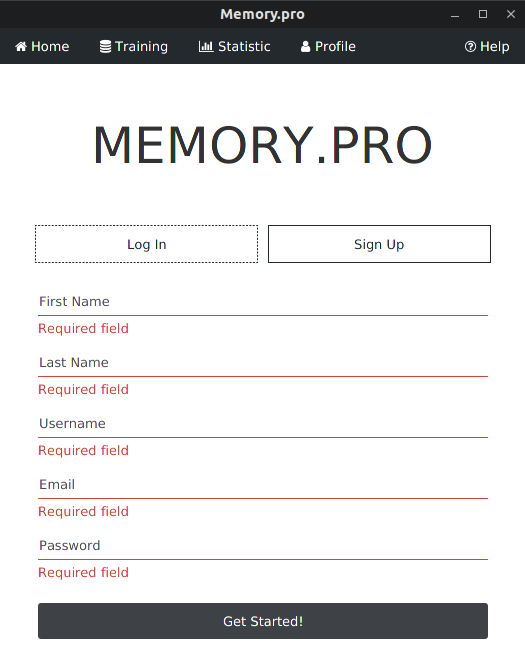
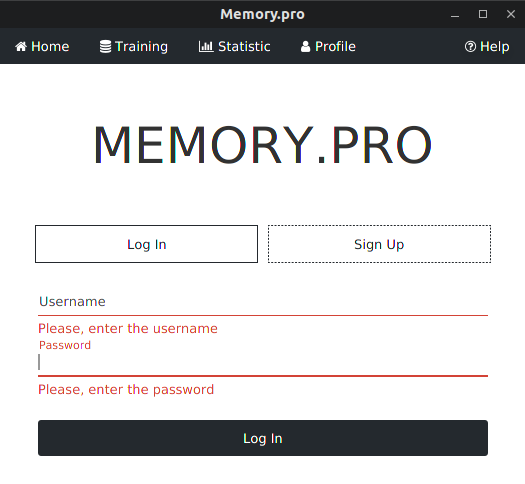


Рис. 3. Валідація форм реєстрації на авторизації

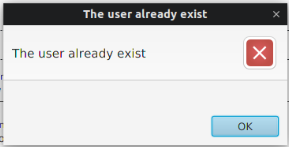
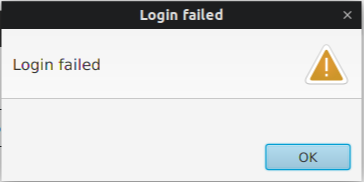


Рис. 4. Повідомлення при неправильних даних

Після авторизації або реєстрації користувач потрапляє на домашню сторінку сервісу. І тепер може перейти, наприклад, у свій кабінет («Profile»).



Рис. 5. Кабінет користувача

Також у своєму кабінеті користувач може передивлятись статистику своїх тренувань по всім доступних типам (та даний момент це числа та слова).

В лівому нижньому кутку кабінету користувача знаходиться кнопка «LOG OUT». Якщо, користувач хоче вийти зі свого профілю то може її натиснути. При цьому, в цілях захисту від випадкового натискання, буде показане спливаюче вікно з підтвердженням

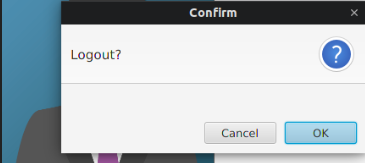


Рис. 6. Підтвердження виходу з системи

При цьому знову відкриється сторінка авторизації, і щоб отримати доступ до сервісу знову доведеться авторизуватись.

Якщо користувач авторизувався а потім закрив програму, то при наступному її запуску йому вже не треба буде авторизовуватися. Програма запам’ятає, що користувач все авторизований.

Якщо користувач авторизований, то від може перейти на сторінку тренування (вкладка меню «Training»).

Щоб почати тренування користувачу потрібно спочатку вибрати параметри тренування:

1. Тип тренування (числа або слова)
2. Кількість елементів даних у тренувані.

Звичайно, якщо дані будуть некоректними (наприклад, кількість елементів від’ємна) відповідне поле буде підсвічене.

Перед початком тренування сторінка виглядає так:

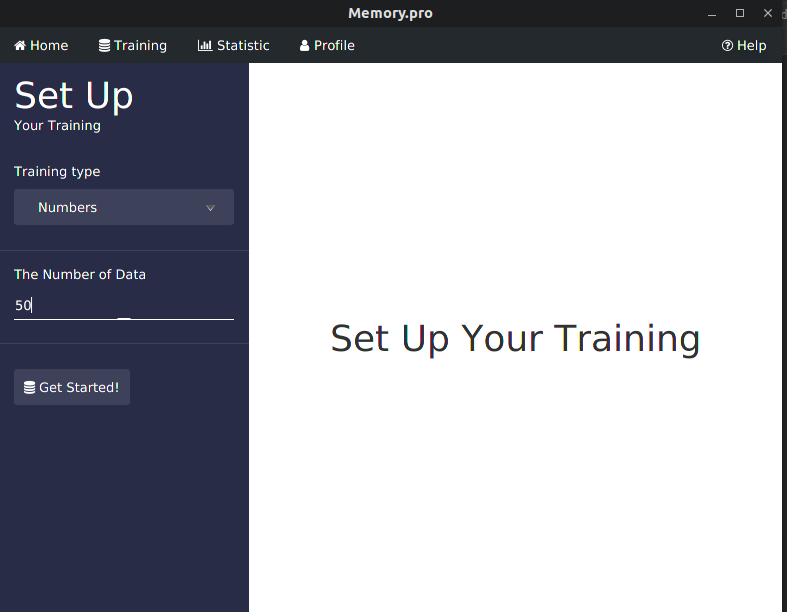


Рис. 7. Сторінка тренування до початку тренування

Коли параметри тренування установленні, користувач може натиснути кнопку «Get Started!» та почати тренування. При цьому надпис посередині сторінки замінюється першим етапом тренування – запам’ятовування.

Під час запам’ятовування програма запам’ятовує, скільки часу користувач запам’ятовував кожен елемент (у мілісекундах).

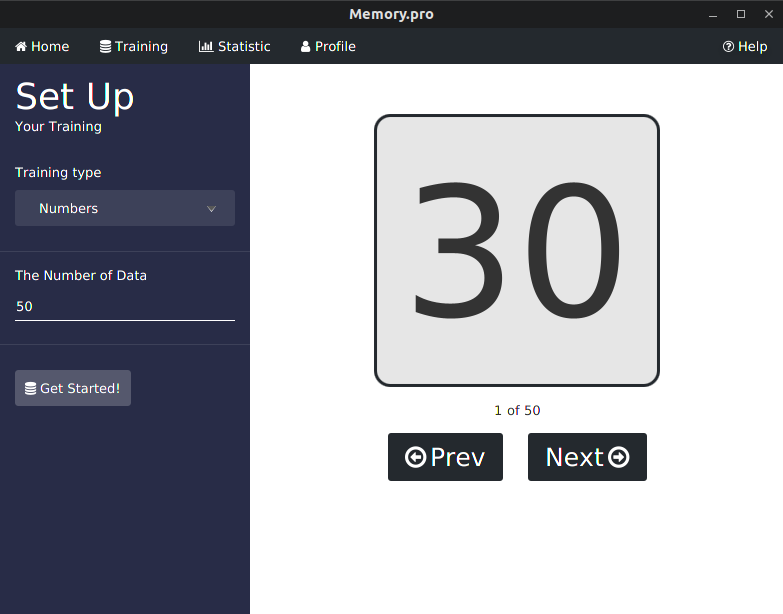


Рис. 8. Етап запам'ятовування у тренуванні "Числа"

Зрозуміло, що при запам’ятовуванні, користувач може натискати кнопки «Next» щоб переходити на наступний елемент та «Prev» - на минулий.

Також варто відмітити, що час запам’ятовування конкретного елемента – це інтервал часу, коли користувач побачив цей елемент уперше и коли переключився на інший.

Коли користувач переключиться на останній елемент, щоб завершити запам’ятовування необхідно ще раз натиснути кнопку «Next», після чого відкривається наступний етап – етап згадування.

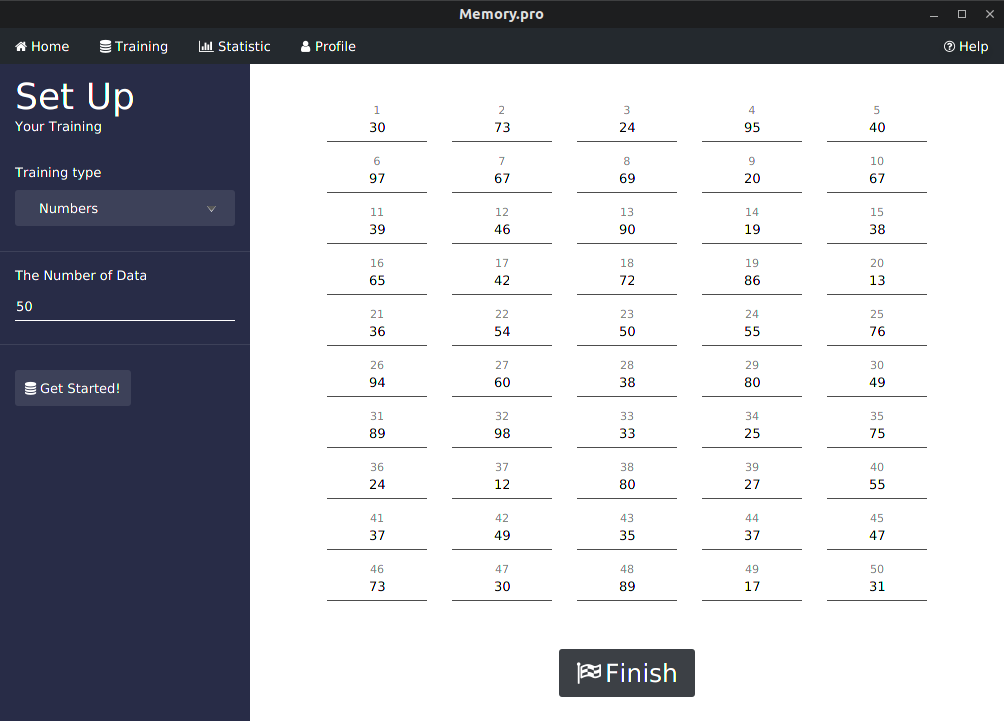


Рис. 9. Третій етап тренування - згадування

Під час згадування користувачу показуються текстові поля – стільки, скільки елементів він вибрав. Спочатку вони пусті, і відображається лише їх порядковий номер. Користувачу треба ввести всі елементи даних, які він зміг запам’ятати у відповідні поля.

Користувач, якщо не пам’ятає певні елементі, може залишити відповідні поля. При цьому, зрозуміло, вони будуть рахуватись за помилку.

Під час згадування програма також слідкує за тим, скільки часу користувач витрачає на згадування. Але тепер час на згадування всіх елементів у секундах.

Після того, як усі поля заповнені достатньо натиснути кнопку «Finish» і тренування буде завершене.

Ось ще зразки етапів запам’ятовування та згадування в тренуванні «Слова»

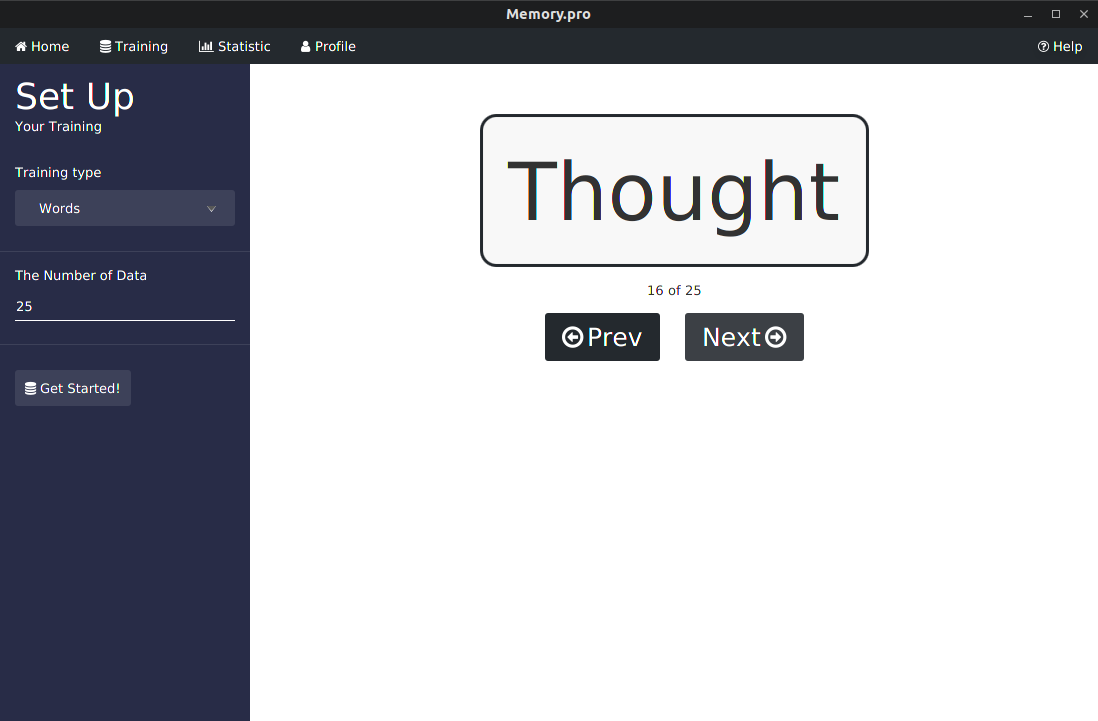


Рис. 10. Тренування "Слова", етап запам'ятовування

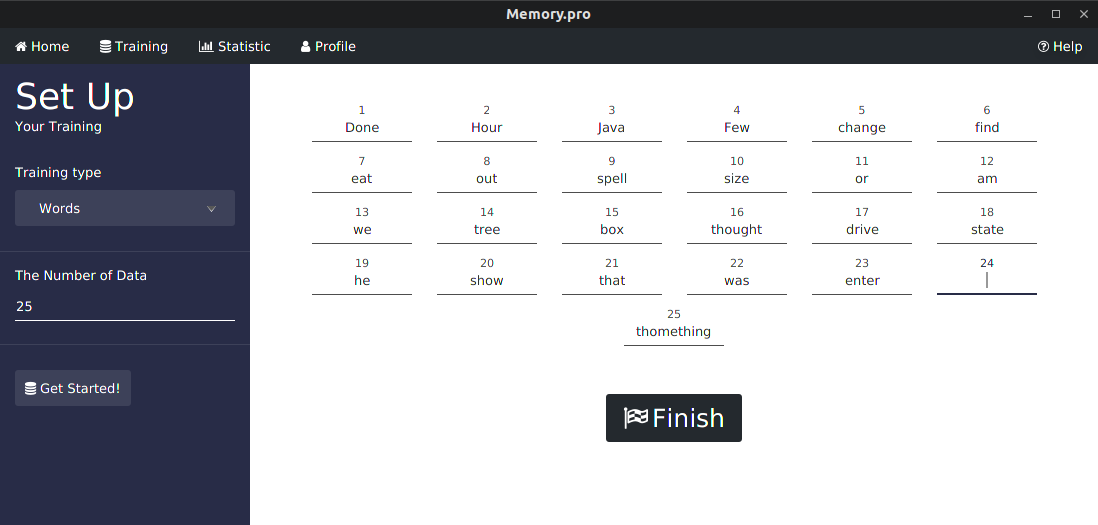


Рис. 11. Тренування «Слова», етап згадування

При тренуванні форма, у якій вказувались параметри тренування нікуди не зникає. Це зроблено для того, що користувач міг почати тренування спочатку, якщо, йому не хочеться завершати теперішнє. Можна почати тренування с тими ж самими параметрами, а можна встановити і нові. Щоб почати тренування спочатку треба просто ще раз натиснути кнопку «Get Started».

Але так як є шанс випадкового натискання цієї кнопки, но перед тим, як почати нове тренування, буде показане спливаюче вікно с підтвердженням

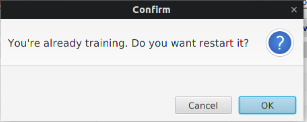


Рис. 12. Підтвердження перезапуску тренування

Після завершення тренування користувачу показується результат тренування. Цей результат зберігається у базі даних, тому пізніше користувач може відкрити результат ще раз, проаналізувати, тощо.

У результаті тренування відображається повна інформація по тренуванню:

* Username
* Тип тренування
* Дата тренування
* Кількість елементів
* Оцінка
* Час на запам’ятовування
* Час на згадування
* Кількість правильно відповідей
* Кількість помилок
* Мінімальний, максимальний та середній час на запам’ятовування одиниці інформації

Також результат містить таблицю, у якій показані усі елементи, що були у тренуванні, час їх запам’ятовування, значення елементу, та відповідь користувача

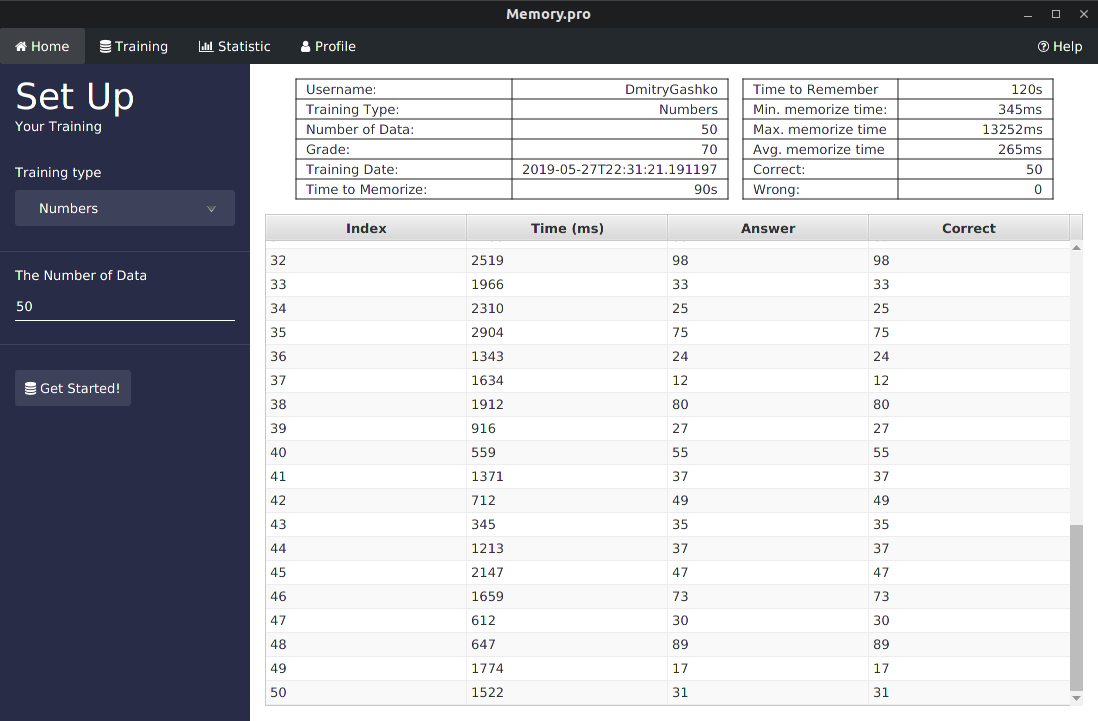
Результат тренування виглядає так:

Рис. 13. Результат (цифри)

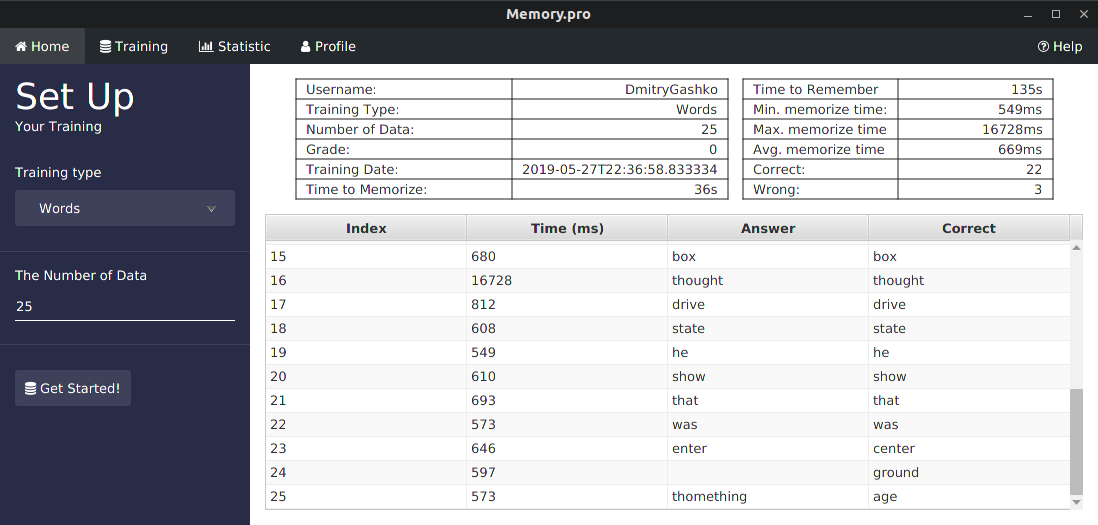


Рис. 14. Результат (Слова)

За результат тренування «Слова» на рис. 14 оцінка 0 балів. Це тому, що було зроблено більше 10% помилок. У цьому випадку тренування відразу отримує оцінку 0. Якщо ви не можете отримати оцінку більше за 0, можливо, ви не знаєте що таке мнемоніка, тому перед тренуваннями необхідно ознайомитись з цим поняттям.

# **Додаток В. Код програмних модулів**

Bootstrap.java

**public class** Bootstrap **extends** Screen {  
 **private** Common **common**;  
 **private** String **currentScreen**;  
  
 @FXML  
 **private** VBox **headerSlot**;  
  
 @FXML  
 **private** StackPane **screenSlot**;  
  
 **public** Bootstrap() {  
 **super**();  
 **common** = **new** Common();  
 }  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initEvents();  
 start();  
 }  
  
 **private void** start() {  
 renderHeader();  
 showScreen(**"index"**);  
 }  
  
 **private void** initEvents() {  
 **common**.getOnScreenChangeCallbacks().add(**this**::showScreen);  
 }  
  
 **private void** showScreen(String alias) {  
 **if** (alias.equals(**currentScreen**)) **return**;  
  
 String path = **common**.getScreens().get(alias);  
  
 **if** (path == **null** || path.isEmpty()) {  
 **alerts**.show(Alerts.*alertErr*, **"Screen Not Found"**);  
 showScreen(**"index"**);  
 **return**;  
 }  
  
 **if** (!alias.equals(**"auth"**) && !checkAuth()) {  
 showScreen(**"auth"**);  
 **return**;  
 }  
  
 Pair<Parent, Screen> componentPair;  
  
 **try** {  
 componentPair = loadScreen(alias);  
 } **catch** (ComponentException e) {  
 **alerts**.showError(e);  
 **return**;  
 }  
  
 Parent root = componentPair.getKey();  
 Screen screen = componentPair.getValue();  
  
 **currentScreen** = alias;  
  
 **screenSlot**.getChildren().clear();  
 **screenSlot**.getChildren().add(root);  
  
 screen.showed();  
 }  
  
 **private void** renderHeader() {  
 Parent header = **null**;  
  
 **try** {  
 header = loadComponent(**"header"**).getKey();  
 } **catch** (ComponentException e) {  
 **alerts**.show(Alerts.*alertErr*, **"Cannot render the Header"**);  
 e.printStackTrace();

return;  
 }  
  
 **headerSlot**.getChildren().clear();  
 **headerSlot**.getChildren().add(header);  
 }  
  
 **private void** showScreen(String alias) {  
 showScreen(alias, **null**);  
 }  
  
 **private boolean** checkAuth() {  
 **return common**.getAuthService().isLoggedIn();  
 }  
  
 **public** Common getCommon() {  
 **return common**;  
 }  
  
 @Override  
 **public void** showed() {  
  
 }  
}

AuthScreen.java

**public class** AuthScreen **extends** Screen {  
 **private** Parent **loginRoot**;  
 **private** Parent **signupRoot**;  
  
 @FXML **private** VBox **fxAuthScreen**;  
 @FXML **private** VBox **fxAuthScreenContent**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initComponents();  
 openLogin();  
 }  
  
  
 **private void** openLogin() {  
 **fxAuthScreenContent**.getChildren().clear();  
 **fxAuthScreenContent**.getChildren().add(**loginRoot**);  
  
 **fxAuthScreen**.getStyleClass().removeIf(className -> className.equals(**"auth-screen--signup"**));  
 **fxAuthScreen**.getStyleClass().add(**"auth-screen--login"**);  
 }  
  
 **private void** openSignup() {  
 **fxAuthScreenContent**.getChildren().clear();  
 **fxAuthScreenContent**.getChildren().add(**signupRoot**);  
  
 **fxAuthScreen**.getStyleClass().removeIf(className -> className.equals(**"auth-screen--login"**));  
 **fxAuthScreen**.getStyleClass().add(**"auth-screen--signup"**);  
 }  
  
 **private void** initComponents() {  
 **try** {  
 **loginRoot** = loadComponent(**"login"**).getKey();  
 **signupRoot** = loadComponent(**"signup"**).getKey();  
  
 } **catch** (ComponentException e) {  
 **alerts**.showError(e);  
 System.*exit*(1);  
 }  
 }  
  
 @FXML **void** onLogIn() {  
 openLogin();  
 }  
 @FXML **void** onSignUp() {  
 openSignup();  
 }  
  
}

LoginComponent.java

**public class** LoginComponent **extends** Component {  
  
 @FXML **private** JFXTextField **fxUsername**;  
 @FXML **private** JFXPasswordField **fxPassword**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initValidation();  
 }  
  
 **private void** initValidation() {  
 Validation.*initValidation*(**fxUsername**);  
 Validation.*initValidation*(**fxPassword**);  
 }  
  
 **private void** login() {  
 **if** (!isValid()) **return**;  
  
 String username = **fxUsername**.getText();  
 String password = **fxPassword**.getText();  
  
 **try** {  
 **common**.getAuthService().login(username, password);  
 } **catch** (AuthServiceException e) {  
 **alerts**.show(Alerts.*alertWarn*, e.getMessage());  
 **return**;  
 }  
  
 **common**.setScreen(**"index"**);  
 }  
  
 **private boolean** isValid() {  
 **boolean** username = **fxUsername**.validate();  
 **boolean** password = **fxPassword**.validate();  
  
 **return** username && password;  
 }  
  
 @FXML **void** onLogin() {  
 login();  
 }  
  
}

SignupComponent.java

**public class** SignupComponent **extends** Component {  
  
 @FXML **private** JFXTextField **fxFirstName**;  
 @FXML **private** JFXTextField **fxLastName**;  
 @FXML **private** JFXTextField **fxUsername**;  
 @FXML **private** JFXTextField **fxEmail**;  
 @FXML **private** JFXPasswordField **fxPassword**;  
 @FXML **private** RegexValidator **fxEmailValidator**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initValidation();  
 }  
  
 **private void** initValidation() {  
 **fxEmailValidator**.setRegexPattern(Common.***EMAIL\_REGEXP***);  
  
 Validation.*initValidation*(**fxFirstName**);  
 Validation.*initValidation*(**fxLastName**);  
 Validation.*initValidation*(**fxUsername**);  
 Validation.*initValidation*(**fxPassword**);  
 Validation.*initValidation*(**fxEmail**);  
 }  
  
 **private void** signup() {  
 **if** (!isValid()) **return**;  
  
 User user = readUserData();  
  
 **try** {  
 **common**.getAuthService().signupAndLogin(user);  
 } **catch** (AuthServiceException e) {  
 **alerts**.show(Alerts.*alertErr*, e.getMessage());  
 **return**;  
 }  
  
 **common**.setScreen(**"index"**);  
 }  
  
 **private** User readUserData() {  
 User user = **new** User();  
  
 user.setFirstName(**fxFirstName**.getText());  
 user.setLastName(**fxLastName**.getText());  
 user.setUsername(**fxUsername**.getText());  
 user.setPassword(**fxPassword**.getText());  
 user.setEmail(**fxEmail**.getText());  
  
 **return** user;  
 }  
  
 **private boolean** isValid() {  
 **boolean** firstName = **fxFirstName**.validate();  
 **boolean** lastName = **fxLastName**.validate();  
 **boolean** username = **fxUsername**.validate();  
 **boolean** password = **fxPassword**.validate();  
 **boolean** email = **fxEmail**.validate();  
  
 **return** firstName && lastName && username && password && email;  
 }  
  
 @FXML **void** onSignup() {  
 signup();  
 }  
  
}

AuthService.java

**public class** AuthService {  
 **private** String **loginDataPath** = **"data/config"**;  
  
 **private** UserDao **userDao**;  
 **private** File **loginDataFile**;  
 **private** Optional<LoginData> **loginDataOpt**;  
  
 **public** AuthService(UserDao userDao) {  
 **this**.**userDao** = userDao;  
  
 initLoginDataFile();  
 loadLoginData();  
 }  
  
 **public void** signup(User user) **throws** AuthServiceException {  
 **if** (**userDao**.checkExist(user)) {  
 **throw new** AuthServiceException(**"The user already exist"**);  
 }  
  
 String salt = PasswordUtils.*generateSalt*(512);  
 String key = PasswordUtils.*hashPassword*(user.getPassword(), salt)  
 .orElseThrow(() -> **new** AuthServiceException(**"Internal error"**));  
  
 user.setPasswordSalt(salt);  
 user.setPasswordKey(key);  
 user.setRegisterDate(LocalDate.*now*());  
  
 **try** {  
 **userDao**.add(user);  
 } **catch** (RollbackException e) {  
 **throw new** AuthServiceException(**"Can't save the user"**);  
 }  
 }  
  
 **public void** login(String username, String password) **throws** AuthServiceException {  
 String salt = **userDao**.getSaltByUsername(username)  
 .orElseThrow(() -> **new** AuthServiceException(**"Login failed"**));  
  
 String key = PasswordUtils.*hashPassword*(password, salt)  
 .orElseThrow(() -> **new** AuthServiceException(**"Login failed"**));  
  
 **if** (!checkKey(username, key)) {  
 **throw new** AuthServiceException(**"Login failed"**);  
 }  
  
 setLoginData(username, key);  
 }  
  
 **public void** signupAndLogin(User user) **throws** AuthServiceException {  
 signup(user);  
 login(user.getUsername(), user.getPassword());  
  
 }  
  
 **public boolean** isLoggedIn() {  
 **if** (**loginDataOpt**.isEmpty()) {  
 **return false**;  
 }  
  
 LoginData loginData = **loginDataOpt**.get();  
  
 **boolean** res = checkKey(loginData.getUsername(), loginData.getKey());  
  
 **if** (!res) {  
 removeLoginData();  
 }  
  
 **return** res;  
 }  
  
 **public boolean** logout() {  
 **return this**.removeLoginData();  
 }  
  
 **private boolean** checkKey(String username, String key) {  
 **return userDao**.checkKey(username, key);  
 }  
  
 **private void** initLoginDataFile() {  
 **loginDataFile** = **new** File(**loginDataPath**);  
 **loginDataFile**.getParentFile().mkdirs();  
 }  
  
 **private void** saveLoginData() {  
 **if** (**loginDataOpt**.isEmpty()) {  
 **return**;  
 }  
  
 **try**(FileOutputStream fos = **new** FileOutputStream(**loginDataFile**);  
 ObjectOutputStream out = **new** ObjectOutputStream(fos)  
 ) {  
  
 out.writeObject(**loginDataOpt**.get());  
  
 } **catch** (IOException e) {  
 **return**;  
 }  
 }  
  
 **private void** setLoginData(String login, String key) {  
 **loginDataOpt** = Optional.*of*(**new** LoginData(login, key));  
 saveLoginData();  
 }  
  
 **private boolean** removeLoginData() {  
 **loginDataOpt** = Optional.*empty*();  
 **return loginDataFile**.delete();  
 }  
  
 **private void** loadLoginData() {  
 **try**(FileInputStream fis = **new** FileInputStream(**loginDataFile**);  
 ObjectInputStream ois = **new** ObjectInputStream(fis)  
 ) {  
  
 **loginDataOpt** = Optional.*of*((LoginData) ois.readObject());  
  
 } **catch** (IOException | ClassNotFoundException e) {  
 **loginDataOpt** = Optional.*empty*();  
 }  
 }  
  
 **public** String getLoggedInUsername() {  
 **if** (**loginDataOpt**.isEmpty() || !isLoggedIn()) {  
 **return ""**;  
 }  
  
 **return loginDataOpt**.get().getUsername();  
 }  
  
}

LoginData.java

**class** LoginData **implements** Serializable {  
  
 **private** String **username**;  
 **private** String **key**;  
  
 **public** LoginData(String username, String key) {  
 **this**.**username** = username;  
 **this**.**key** = key;  
 }  
  
 **public** String getUsername() {  
 **return username**;  
 }  
  
 **public void** setUsername(String username) {  
 **this**.**username** = username;  
 }  
  
 **public** String getKey() {  
 **return key**;  
 }  
  
 **public void** setKey(String key) {  
 **this**.**key** = key;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "LoginData{"** +  
 **"username='"** + **username** + **'\''** +  
 **", key='"** + **key** + **'\''** +  
 **'}'**;  
 }  
}

PaswordUtils.java

**public class** PasswordUtils {  
 **private static final** SecureRandom ***RAND*** = **new** SecureRandom();  
  
 **private static final int *ITERATIONS*** = 65536;  
 **private static final int *KEY\_LENGTH*** = 512;  
 **private static final** String ***ALGORITHM*** = **"PBKDF2WithHmacSHA512"**;  
  
 **static public** String generateSalt(**int** len) {  
 **if** (len < 1) len = 1;  
  
 **byte**[] salt = **new byte**[len];  
 ***RAND***.nextBytes(salt);  
  
 **return** Base64.*getEncoder*().encodeToString(salt);  
 }  
  
 **static public** Optional<String> hashPassword(String password, String salt) {  
 **if** (salt == **null** || salt.isEmpty()) {  
 **return** Optional.*empty*();  
 }  
  
 **char**[] passwordChars = password.toCharArray();  
 **byte**[] saltBytes = salt.getBytes();  
  
 PBEKeySpec spec = **new** PBEKeySpec(passwordChars, saltBytes, ***ITERATIONS***, ***KEY\_LENGTH***);  
  
 Arrays.*fill*(passwordChars, Character.***MIN\_VALUE***);  
  
 **try** {  
 SecretKeyFactory fac = SecretKeyFactory.*getInstance*(***ALGORITHM***);  
 **byte**[] securePassword = fac.generateSecret(spec).getEncoded();  
 **return** Optional.*of*(Base64.*getEncoder*().encodeToString(securePassword));  
  
 } **catch** (NoSuchAlgorithmException | InvalidKeySpecException ex) {  
 System.***err***.println(**"Exception encountered in hashPassword()"**);  
 **return** Optional.*empty*();  
  
 } **finally** {  
 spec.clearPassword();  
 }  
 }  
  
}

TrainingScreen.java

**public class** TrainingScreen **extends** Screen {  
  
 @FXML **private** BorderPane **fxLeftContainer**;  
 @FXML **private** BorderPane **fxCenterContainer**;  
  
 **private** TrainingSetupComponent **setupComponent**;  
 **private** TrainingMemorizeComponent **memorizeComponent**;  
 **private** TrainingRememberComponent **rememberComponent**;  
 **private** ResultComponent **resultComponent**;  
  
 **private** Parent **setupComponentRoot**;  
 **private** Parent **memorizeComponentRoot**;  
 **private** Parent **rememberComponentRoot**;  
 **private** Parent **resultComponentRoot**;  
  
 **private** TrainingService **trainingService**;  
 **private** ArrayList<String> **trainingData**;  
  
 **private boolean isTrainingInit** = **false**;  
  
 **private** ArrayList<Integer> **timesToMemorize**;  
 **private** ArrayList<String> **answers**;  
 **private int timeToRemember**;  
 **private** String **trainingType**;  
 **private int dataCount**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initChildComponents();  
 }  
  
 @Override  
 **public void** showed() {  
 runSetup();  
 }  
  
 **private void** runSetup() {  
 **fxLeftContainer**.getChildren().clear();  
 **fxLeftContainer**.setCenter(**setupComponentRoot**);  
  
 **setupComponent**.run((trainingType, dataCount) -> {  
 **this**.**trainingType** = trainingType;  
 **this**.**dataCount** = dataCount;  
  
 **if** (**isTrainingInit**) {  
 **var** ans = **alerts**.ask(**"You're already training. Do you want restart it?"**);  
 **if** (!ans) **return**;  
 }  
  
 **isTrainingInit** = **true**;  
  
 runMemorize();  
 });  
 }  
  
 **private void** runMemorize() {  
 initTrainingService();  
  
 **trainingService**.setUp(**dataCount**);  
 **trainingData** = **trainingService**.start();  
  
 **fxCenterContainer**.getChildren().clear();  
 **fxCenterContainer**.setCenter(**memorizeComponentRoot**);  
  
 System.***out***.println(**trainingData**);  
  
 **memorizeComponent**.run(**trainingData**, **trainingType**, (timesToMemorize) -> {  
 **this**.**timesToMemorize** = timesToMemorize;  
  
 runRemember();  
 });  
 }  
  
 **private void** runRemember() {  
 **fxCenterContainer**.getChildren().clear();  
 **fxCenterContainer**.setCenter(**rememberComponentRoot**);  
  
 **rememberComponent**.run(**dataCount**, (timeToRemember, answers) -> {  
 **this**.**timeToRemember** = timeToRemember;  
 **this**.**answers** = answers;  
  
 finishTraining();  
 });  
 }  
  
 **private void** finishTraining() {  
 TrainingResult result = getResult();  
 **int** resultId = **trainingService**.finish(result);  
  
 **fxCenterContainer**.getChildren().clear();  
 **fxCenterContainer**.setCenter(**resultComponentRoot**);  
  
 **resultComponent**.run(**trainingType**, resultId);  
 }  
  
 **private** TrainingResult getResult() {  
 TrainingResult result = **new** TrainingResult();  
  
 result.setTrainingType(**trainingType**);  
 result.setDataCount(**dataCount**);  
 result.setTimesToMemorize(**timesToMemorize**);  
 result.setTimeToRemember(**timeToRemember**);  
 result.setAnswers(**answers**);  
  
 **return** result;  
 }  
  
 **private void** initTrainingService() {  
 User user = getUser();  
  
 **if** (user == **null**) {  
 **alerts**.show(Alerts.*alertErr*, **"Internal Error"**);  
 **return**;  
 }  
  
 **var** em = **common**.getEm();  
  
 **if** (**trainingType**.equals(**"Words"**)) {  
 **trainingService** = **new** WordsTrainingService(user, em);  
  
 } **else if** (**trainingType**.equals(**"Numbers"**)) {  
 **trainingService** = **new** NumberTrainingService(user, em);  
  
 } **else** {  
 **alerts**.show(Alerts.*alertErr*, **"Wrong Training Type"**);  
 System.*exit*(1);  
 }  
  
 }  
  
 **private void** initChildComponents() {  
 **try** {  
 **var** setupPair = loadComponent(**"setup"**);  
 **var** memorizePair = loadComponent(**"memorize"**);  
 **var** rememberPair = loadComponent(**"remember"**);  
 **var** resultPair = loadComponent(**"result"**);  
  
 **setupComponent** = (TrainingSetupComponent) setupPair.getValue();  
 **memorizeComponent** = (TrainingMemorizeComponent) memorizePair.getValue();  
 **rememberComponent** = (TrainingRememberComponent) rememberPair.getValue();  
 **resultComponent** = (ResultComponent) resultPair.getValue();  
  
 **setupComponentRoot** = setupPair.getKey();  
 **memorizeComponentRoot** = memorizePair.getKey();  
 **rememberComponentRoot** = rememberPair.getKey();  
 **resultComponentRoot** = resultPair.getKey();  
  
 } **catch** (ComponentException e) {  
 **alerts**.show(Alerts.*alertErr*, **"Cant'l load Training Components"**);  
 e.printStackTrace();  
 }  
 }

ITrainingService.java

**public interface** ITrainingService {  
  
 */\*\*  
 \** ***@param dataCount*** *the number of data  
 \*/* **void** setUp(**int** dataCount);  
  
 */\*\*  
 \** ***@return*** *training data as string array  
 \*/* ArrayList<String> start();  
  
 */\*\*  
 \** ***@return*** *result id  
 \*/* **int** finish(TrainingResult result);  
  
}

IResultService.java

**public interface** IResultService {  
  
 Optional<Result> loadResult(**int** resultId);  
  
}

ResultService.Result.java

**public class** Result {  
  
 **private int id**;  
 **private** String **username**;  
  
 **private int grade**;  
 **private int dataCount**;  
  
 **private** LocalDateTime **startTime**;  
 **private int memorizeTime**;  
 **private int rememberTime**;  
  
 **private int minMemorizeTime**;  
 **private int maxMemorizeTime**;  
 **private int avgMemorizeTime**;  
  
 **private int correctAns**;  
  
 **private** ArrayList<ResultData> **data**;  
  
 **public int** getErrors() {  
 **return dataCount** - **correctAns**;  
 }  
  
}

ResultService.ResultData.java

**public class** ResultData {  
  
 **private** String **value**;  
 **private** String **answer**;  
 **private int time**;  
 **private int index**;

}

TrainingSetupComponent.java

**public class** TrainingSetupComponent **extends** Component {  
  
 @FXML **private** JFXComboBox **fxSelectTrainingType**;  
 @FXML **private** JFXTextField **fxNumberOfData**;  
  
 **private** OnDoneCallback **onDoneCallback**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initValidation();  
 }  
  
 **public void** run(OnDoneCallback onDoneCallback) {  
 **this**.**onDoneCallback** = onDoneCallback;  
 }  
  
 **private void** done() {  
 **if** (!isValid()) **return**;  
  
 String trainingType = ((Label)**fxSelectTrainingType**.getValue()).getText();  
 **int** dataCount = Integer.*parseInt*(**fxNumberOfData**.getText());  
  
 **if** (dataCount < 1 || dataCount > 2000) {  
 **alerts**.show(Alerts.*alertErr*, **"Check your data"**);  
 **return**;  
 }  
  
 **onDoneCallback**.call(trainingType, dataCount);  
 }  
  
 **private void** initValidation() {  
 Validation.*initValidation*(**fxSelectTrainingType**);  
 Validation.*initValidation*(**fxNumberOfData**);  
 }  
  
 **private boolean** isValid() {  
 **boolean** type = **fxSelectTrainingType**.validate();  
 **boolean** dataCount = **fxNumberOfData**.validate();  
  
 **return** type && dataCount;  
 }  
  
 @FXML **public void** onDone() {  
 done();  
 }  
  
 @FunctionalInterface  
 **public interface** OnDoneCallback {  
 **void** call(String trainingType, **int** dataCount);  
 }  
}

TrainingRemember.java

**public class** TrainingRememberComponent **extends** Component {  
  
 @FXML **private** FlowPane **fxInputsContainer**;  
  
 **private** OnDoneCallback **onDoneCallback**;  
  
 **private** LocalDateTime **startTime**;  
  
 **private int dataCount**;  
  
 **private boolean isDone**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initValidation();  
 }  
  
 **public void** run(**int** dataCount, OnDoneCallback onDoneCallback) {  
 **this**.**onDoneCallback** = onDoneCallback;  
 **this**.**dataCount** = dataCount;  
  
 **this**.**isDone** = **false**;  
 **this**.**startTime** = LocalDateTime.*now*();  
  
 renderInputs();  
 }  
  
 **private void** renderInputs() {  
 ArrayList<JFXTextField> inputs = **new** ArrayList<>(**dataCount**);  
  
 **for** (**int** i = 0; i < **dataCount**; i++) {  
 JFXTextField input = **new** JFXTextField();  
 input.setPromptText(Integer.*toString*(i + 1));  
  
 inputs.add(input);  
 }  
  
 **fxInputsContainer**.getChildren().clear();  
 **fxInputsContainer**.getChildren().addAll(inputs);  
 }  
  
 **private** ArrayList<String> getAnswers() {  
 **return this**.**fxInputsContainer**.getChildren().stream()  
 .map(input -> ((JFXTextField)input).getText())  
 .collect(Collectors.*toCollection*(ArrayList::**new**));  
 }  
  
 **private void** done() {  
 **if** (**isDone**) **return**;  
 **isDone** = **true**;  
  
 **var** answers = getAnswers();  
 **int** time = getTimeToRemember();  
  
 **onDoneCallback**.call(time, answers);  
 }  
  
 **private int** getTimeToRemember() {  
 LocalDateTime endTime = LocalDateTime.*now*();  
  
 **return** (**int**)***SECONDS***.between(**startTime**, endTime);  
 }  
  
 **private void** initValidation() {  
  
 }  
  
 @FXML **public void** onDone() {  
 done();  
 }  
  
 @FunctionalInterface  
 **public interface** OnDoneCallback {  
 **void** call(**int** timeToRemember, ArrayList<String> answers);  
 }

TrainingRememberComponent.java

**public class** TrainingRememberComponent **extends** Component {  
  
 @FXML **private** FlowPane **fxInputsContainer**;  
  
 **private** OnDoneCallback **onDoneCallback**;  
  
 **private** LocalDateTime **startTime**;  
  
 **private int dataCount**;  
  
 **private boolean isDone**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
 initValidation();  
 }  
  
 **public void** run(**int** dataCount, OnDoneCallback onDoneCallback) {  
 **this**.**onDoneCallback** = onDoneCallback;  
 **this**.**dataCount** = dataCount;  
  
 **this**.**isDone** = **false**;  
 **this**.**startTime** = LocalDateTime.*now*();  
  
 renderInputs();  
 }  
  
 **private void** renderInputs() {  
 ArrayList<JFXTextField> inputs = **new** ArrayList<>(**dataCount**);  
  
 **for** (**int** i = 0; i < **dataCount**; i++) {  
 JFXTextField input = **new** JFXTextField();  
 input.setPromptText(Integer.*toString*(i + 1));  
  
 inputs.add(input);  
 }  
  
 **fxInputsContainer**.getChildren().clear();  
 **fxInputsContainer**.getChildren().addAll(inputs);  
 }  
  
 **private** ArrayList<String> getAnswers() {  
 **return this**.**fxInputsContainer**.getChildren().stream()  
 .map(input -> ((JFXTextField)input).getText())  
 .collect(Collectors.*toCollection*(ArrayList::**new**));  
 }  
  
 **private void** done() {  
 **if** (**isDone**) **return**;  
 **isDone** = **true**;  
  
 **var** answers = getAnswers();  
 **int** time = getTimeToRemember();  
  
 **onDoneCallback**.call(time, answers);  
 }  
  
 **private int** getTimeToRemember() {  
 LocalDateTime endTime = LocalDateTime.*now*();  
  
 **return** (**int**)***SECONDS***.between(**startTime**, endTime);  
 }  
  
 **private void** initValidation() {  
  
 }  
  
 @FXML **public void** onDone() {  
 done();  
 }  
  
 @FunctionalInterface  
 **public interface** OnDoneCallback {  
 **void** call(**int** timeToRemember, ArrayList<String> answers);  
 }

ResultComponent.java

**public class** ResultComponent **extends** Component {  
  
 @FXML **private** TableView **fxDataTable**;  
 @FXML **private** TableColumn **fxDataTableIndex**;  
 @FXML **private** TableColumn **fxDataTableTime**;  
 @FXML **private** TableColumn **fxDataTableAnswer**;  
 @FXML **private** TableColumn **fxDataTableCorrect**;  
  
 **private** SimpleIntegerProperty **resultId** = **new** SimpleIntegerProperty();  
 **private** SimpleStringProperty **username** = **new** SimpleStringProperty();  
 **private** SimpleIntegerProperty **grade** = **new** SimpleIntegerProperty();  
 **private** SimpleIntegerProperty **dataCount** = **new** SimpleIntegerProperty();  
 **private** SimpleStringProperty **trainingType** = **new** SimpleStringProperty();  
 **private** SimpleStringProperty **startTime** = **new** SimpleStringProperty();  
 **private** SimpleIntegerProperty **memorizeTime** = **new** SimpleIntegerProperty();  
 **private** SimpleIntegerProperty **rememberTime** = **new** SimpleIntegerProperty();  
 **private** SimpleIntegerProperty **minMemorizeTime** = **new** SimpleIntegerProperty();  
 **private** SimpleIntegerProperty **maxMemorizeTime** = **new** SimpleIntegerProperty();  
 **private** SimpleIntegerProperty **avgMemorizeTime** = **new** SimpleIntegerProperty();  
 **private** SimpleIntegerProperty **correct** = **new** SimpleIntegerProperty();  
 **private** SimpleIntegerProperty **errors** = **new** SimpleIntegerProperty();  
  
 **private** ResultService **resultService**;  
 **private** Result **result**;  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
  
 }  
  
 **public void** run(String trainingType, **int** resultId) {  
 **this**.**trainingType**.set(trainingType);  
 **this**.**resultId**.set(resultId);  
  
 initResultService();  
  
 Optional<Result> resultOption = **resultService**.loadResult(resultId);  
  
 **if** (resultOption.isEmpty()) {  
 **alerts**.show(Alerts.*alertErr*, **"The result not found"**);  
 }  
  
 **result** = resultOption.get();  
 initProperties();  
  
 initDataTable();  
 }  
  
 **private void** initProperties() {  
 **username**.set(**result**.getUsername());  
 **grade**.set(**result**.getGrade());  
 **dataCount**.set(**result**.getDataCount());  
 **startTime**.set(**result**.getStartTime().toString());  
 **memorizeTime**.set(**result**.getMemorizeTime());  
 **rememberTime**.set(**result**.getRememberTime());  
 **minMemorizeTime**.set(**result**.getMinMemorizeTime());  
 **maxMemorizeTime**.set(**result**.getMaxMemorizeTime());  
 **avgMemorizeTime**.set(**result**.getAvgMemorizeTime());  
 **correct**.set(**result**.getCorrectAns());  
 **errors**.set(**result**.getErrors());  
 }  
  
 **private void** initResultService() {  
 String type = **this**.**trainingType**.get();  
  
 **if** (type.equals(**"Numbers"**)) {  
 **resultService** = **new** NumberResultService(**common**.getEm());  
 } **else if** (type.equals(**"Words"**)) {  
 **resultService** = **new** WordsResultService(**common**.getEm());  
 } **else** {  
 **alerts**.show(Alerts.*alertErr*, **"Unknown training type"**);  
 }  
 }  
  
 **private void** initDataTable() {  
 **fxDataTableIndex**.setCellValueFactory(**new** PropertyValueFactory<>(**"index"**));  
 **fxDataTableTime**.setCellValueFactory(**new** PropertyValueFactory<>(**"time"**));  
 **fxDataTableAnswer**.setCellValueFactory(**new** PropertyValueFactory<>(**"answer"**));  
 **fxDataTableCorrect**.setCellValueFactory(**new** PropertyValueFactory<>(**"value"**));  
  
 **fxDataTable**.setItems(getDataTableItems());  
 }  
  
 **private** ObservableList<ResultData> getDataTableItems() {  
 ObservableList<ResultData> collection = FXCollections.observableArrayList();  
 collection.addAll(**result**.getData());  
  
 **return** collection;  
 }

}

Header.java

**public class** HeaderComponent **extends** Component {  
  
 @Override  
 **public void** initialize(URL url, ResourceBundle resourceBundle) {  
  
 }  
  
 **private void** goHome() {  
 **common**.setScreen(**"index"**);  
 }  
  
 **private void** goToTraining() {  
 **common**.setScreen(**"training"**);  
 }  
  
 **private void** goToStatistic() {  
 **common**.setScreen(**"statistics"**);  
 }  
  
 **private void** goToProfile() {  
 **common**.setScreen(**"profile"**);  
 }  
  
 **private void** openHelp() {  
 String title = **"Memory.pro - you can memorize everything!"**;  
  
 String content = **"Contacts: dimagashko@gmail.com\n"** +  
 **"\n"** +  
 **"You can fork this project:\n"** +  
 **"https://github.com/DimaGashko/memory-pro\n"** +  
 **"\n"** +  
 **"Copyright \u00a9 2019 Dmitry Gashko\n"**;  
  
 **alerts**.show(Alerts.*alertInfo*, **"Memory.pro"**, title, content);  
 }  
  
 @FXML **public void** onHome() { goHome(); }  
 @FXML **public void** onStatistic() { goToStatistic(); }  
 @FXML **public void** onTraining() { goToTraining(); }  
 @FXML **public void** onProfile() { goToProfile(); }  
 @FXML **public void** onHelp() { openHelp(); }  
}