РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: мобільний застосунок для написання та перевірки диктантів українською мовою.

Мета кваліфікаційної роботи: створення мобільного застосунку для написання та перевірки словникових диктантів, що сприятиме засвоєнню правил оновленого Українського правопису.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в тому, що використання даного мобільного застосунку значно спростить адаптацію людей до правил оновленого Українського правопису.

Актуальність теми кваліфікаційної роботи визначається тим, що наразі для операційної системи Android не існує застосунків, з подібним функціоналом: диктування за допомогою синтезатора голосу, повна перевірка знання орфографії, незалежність від наявності підключення до інтернету.

Список ключових слів: МОБІЛЬНИЙ ЗАСТОСУНОК, УКРАЇНСЬКИЙ ПРАВОПИС, СИНТЕЗАТОР МОВЛЕННЯ, ДИКТАНТ ДИНАМІЧНЕ ПРОГРАМУВАННЯ, ANDROID.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: mobile application for writing and checking dictations in Ukrainian.

The purpose of the qualification work: to create a mobile application for writing and checking vocabulary dictations, which will help master the rules of the updated Ukrainian spelling.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and its scope, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of the work is that the use of this mobile application will greatly simplify the adaptation of people to the rules of the updated Ukrainian spelling.

The relevance of the topic of qualifying work is determined by the fact that currently there are no applications for the Android operating system with such functionality: dictation using a voice synthesizer, full spell check, regardless of the availability of an Internet connection.

List of keywords: MOBILE APPLICATION, UKRAINIAN SPELLING, SPEECH SYNTHESIZER, DICTIONARY DYNAMIC PROGRAMMING, ANDROID.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЗНО – Зовнішнє незалежне оцінювання;

ПК – персональний комп'ютер;

СМ – синтез мовлення;

Google TTS – Google Text-to-Speech, Ґуґл текст-у-мовлення;

IDE – інтегроване середовище розробки;

LCS – longest common subsequence, пошук найдовшої спільної підпослідовності.

ЗМІСТ

ВСТУП

22 травня 2019 р. Постановою №437 Кабінету Міністрів України введено в дію оновлений Український правопис. Таким чином виникає необхідність адаптації людей до введених змін. Для спрощення цього процесу видається доцільним використовувати можливості сучасних мобільних пристроїв, які є у розпорядженні більшості користувачів.

Метою кваліфікаційної роботи є створення мобільного застосунку для написання та перевірки словникових диктантів, що сприятиме засвоєнню правил оновленого Українського правопису.

Даний мобільний додаток призначений для адаптація людей до правил оновленого Українського правопису.

Для досягнення мети слід виконати такі завдання:

– проаналізувати зміни в оновленому Українського правопису;

– порівняти наявне програмне забезпечення аналогічного призначення;

– сформулювати функціональні вимоги до власного програмного забезпечення;

– обґрунтувати вибір інструментів розробки;

– спроєктувати алгоритми і структури даних;

– створити мобільний застосунок;

– здійснити тестування та, за необхідності, вдосконалення.

В результаті виконання роботи створено мобільний застосунок для написання словникових диктантів українською мовою в умовах набуття чинності оновленого Українського правопису.

Наразі для операційної системи Android не існує застосунків, з подібним функціоналом: диктування за допомогою синтезатора голосу, повна перевірка знання орфографії, незалежність від наявності підключення до інтернету. Ключовою особливістю даної роботи є саме використання випадково згенерованих на основі інформації про користувача диктантів. Це дозволяє виявляти теми, що потребують повторення, та спонукає користувача до

навчання. Кожен новий диктант є унікальним, і генерується персонально для користувача. В програмі наявна система збереження результатів, за допомогою якої можна переглянути свій прогрес в навчанні.

Практичне значення роботи полягає в тому, що створене в її рамках програмне забезпечення може використовуватися для спрощення адаптації людей до правил оновленого Українського правопису.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Особливості введення в дію нового Українського правопису

Постановою №437 Кабінету Міністрів України [8] було введено в дію новий Український правопис [13].

Розглянемо стислий перелік змін, розроблений Інститутом мовознавства Національної академії наук України [10]. Їх можна умовно поділити на дві групи: власне зміни у правилах, та доповнення до чинної норми. Розглянемо основні з них.

Зміни у правилах:

1. Передавання іншомовних коренів (проєкт, проєкція), звуків (плеєр, конвеєр, феєрверк, Соєр, Хаям, Феєрбах) та буквосполучень (Дікенс, Текерей, Бекі).

2. Правопис префіксів разом (мінісукня, віцепрезидент, ексміністр) та окремо (пів хвилини, пів яблука, пів Києва).

3. Правопис російських прізвищ (Донський, Трубецький).

4. Зміни у правилах, що впливають на правопис окремих слів (священник, Святвечір).

5. Правопис великої літери у назвах товарних знаків, марок виробів («Автомобілі марки «Жигулі» вироблялися з 1970 по 2014 рік», «Він приїхав на старих обшарпаних «жигулях»»).

Доповнення до чинної норми:

1. Використання літер г або ґ під час передавання звуку [g] (Верґілій – Вергілій, Ґеорг – Георг, Ґете – Гете).

2. Використання буквосполучень ау або ав під час передавання давньогрецького й латинського au (аудієнція – авдієнція, аудиторія – авдиторія, лауреат – лавреат).

3. Використання літер т або ф під час передавання буквосполучення th у словах грецького походження (ефір – етер, кафедра – катедра, логарифм – логаритм).

4. На початку слова пишемо літеру і. Проте дозволяється варіативне написання у словах ірій – ирий та ірод – ирод.

5. Варіативні форми родового відмінка іменників на -ть та слів кров, любов, осінь, сіль, Русь, Білорусь (радості – радости, любові – лобови, Білорусі – Білоруси).

Аналізуючи викладені новації, наведемо кілька основних найбільших правил Українського правопису 2019 року:

1. Тепер пишемо архимандрит, архиєрей, архиєпископ, архиєрей тощо через И – поряд із попередніми архімандрит, архієрей, архієпископ, архієрей.

2. Тепер пишемо також гідности, незалежности, радости, смерти, чести, хоробрости; крови, любови, осени, соли, Руси, Білоруси – як варіант, із -И в кінці в родовому відмінку однини; також лишається дотеперішня норма на -І.

3. Тепер «топ-100», «топ-десять» – «поза законом».

4. Тепер назви сайтів та інших інтернет-сервісів пишемо тільки українською, з обов’язковим відмінюванням: твітер, ґуґл; мережа «Фейсбук», енциклопедія «Вікіпедія»; фейсбука, ютуба, імейла (із закінченням -А, -Я в родовому відмінку однини).

5. Слова «ви», «ваш» тощо завжди пишемо з малої літери, якщо тільки це не лист із персональним зверненням до однієї особи з виявом особливої ввічливості.

6. Тепер пишемо тільки разом: віцепрем’єр, віцеконсул, ексчемпіонка, ексміністр, експрезидент, контрадмірал, вебсайт, вебсторінка, преміумклас, максісукня, мідімода, мініспідниця, топменеджер, топмодель, лейбгвардієць, лейбмедик, оберофіцер, оберлейтенант, оберпрокурор, штабскапітан, унтерофіцер тощо.

7. Тепер пишемо тільки окремо пів у значенні половина: пів аркуша, пів відра, пів години, пів літра, пів міста, пів огірка, пів острова, пів яблука, пів

ящика, пів ями; пів Європи, пів Києва, пів України.

8. Тепер пишемо тільки двоактний, двоокис, двооксид, двоопуклий, двоосьовий, триатомний, триокисень, чотириосьовий – а не «двох-», «трьох-», «чотирьох-», як раніше, перед наступним А чи О).

Зробивши огляд змін в Українському правописі приходимо до висновку, що існує необхідність полегшити адаптацію носіїв української мови до її нових правил.

1.1.2. Огляд наявного програмного забезпечення аналогічного спрямування

Під час огляду програм, наявних для завантаження на платформі Google Play, спостерігаємо, що повних аналогів для операційної системи Android не існує.

Наявні лише застосунки, що частково реалізують один із видів запропонованого функціоналу. Наприклад, в застосунку для вивчення іноземних мов «Doulingo» [15] (рис. 1.1) наявна можливість написання невеличкого диктанту.

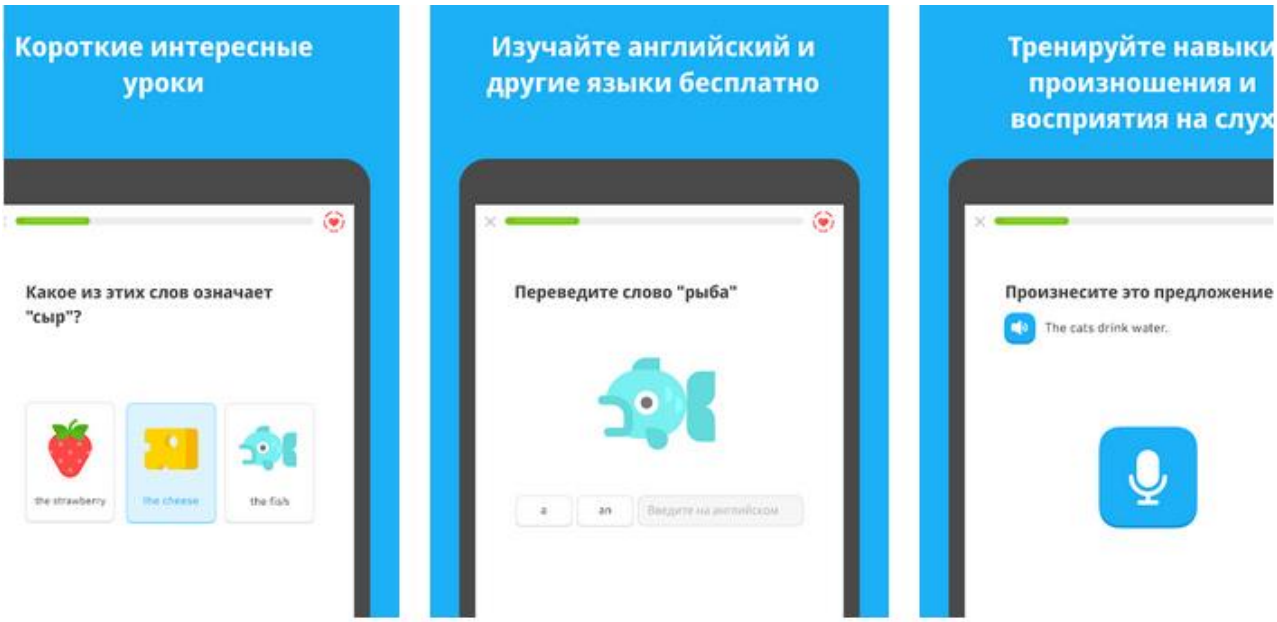

Рис. 1.1. Застосунок для вивчення іноземних мов «Doulingo»

Але при цьому він перевіряє знання саме правил синтаксису, а не орфографії. До того ж застосунок не оптимізований для покращення знань з української мови, адже позиціонує себе в першу чергу як помічник у вивченні іноземних мов. Додаток не вміє працювати в роздільному вікні, немає підтримки планшетів.

Розглянемо інший приклад. Застосунок «ЗНО 2020 тести: Українська мова» [5] (рис. 1.2) дозволяє користувачам скласти тест ЗНО, і таким чином перевірити свої знання орфографії.

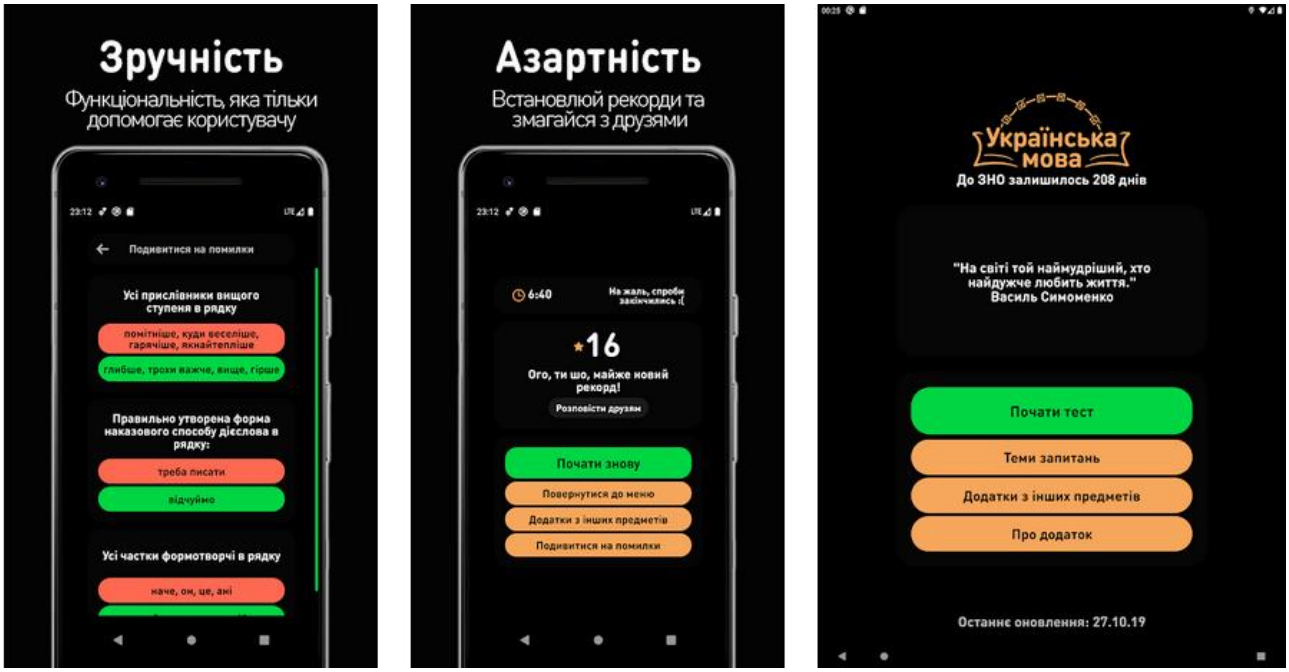

Рис. 1.2. Застосунок «ЗНО 2020 тести: Українська мова»

Але при цьому застосунок орієнтований саме на підготовку до ЗНО 2021 року, тож він перевіряє знання правил старої редакції Українського правопису, і не є аналогом.

Таким чином зроблено висновок, що у крамниці застосунків Google Play відсутнє програмне забезпечення, спрямоване на ознайомлення користувачів з правилами нового правопису, що підтверджує необхідність у створенні такого застосунку.

1.2. Призначення розробки та галузь застосування

Наразі для операційної системи Android не існує застосунків, з подібним функціоналом: диктування за допомогою синтезатора голосу, повна перевірка знання орфографії, незалежність від наявності підключення до інтернету. Ключовою особливістю даної роботи є саме використання випадково згенерованих на основі інформації про користувача диктантів. Це дозволяє виявляти теми, що потребують повторення, та спонукає користувача до навчання. Кожен новий диктант є унікальним, і генерується персонально для користувача. В програмі наявна система збереження результатів, за допомогою якої можна переглянути свій прогрес в навчанні.

Створене в рамках даної кваліфікаційної роботи програмне забезпечення може використовуватися для спрощення адаптації людей до правил оновленого Українського правопису.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка мобільного застосунку для написання словникових диктантів за правилами оновленого Українського правопису» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою кваліфікаційної роботи є створення мобільного застосунку для написання та перевірки словникових диктантів, що сприятиме засвоєнню правил оновленого Українського правопису.

Для досягнення мети слід виконати такі завдання:

– проаналізувати зміни в оновленому Українського правопису;

– порівняти наявне програмне забезпечення аналогічного призначення;

– сформулювати функціональні вимоги до власного програмного забезпечення;

– обґрунтувати вибір інструментів розробки;

– спроєктувати алгоритми і структури даних;

– створити мобільний застосунок;

– здійснити тестування та, за необхідності, вдосконалення.

В результаті виконання роботи повинен бути створений мобільний застосунок для написання словникових диктантів українською мовою в умовах набуття чинності оновленого Українського правопису.

Після аналізу особливості введення в дію нового Українського правопису можемо сформулювати перелік задач, які повинен виконувати створюваний застосунок:

1. Видобути із покажчика Українського правопису слова та номери правил, які необхідно знати для написання кожного слова.

2. Виправити помилки у словах покажчика та відсіяти слова, непридатні для диктування.

3. Розробити структуру створюваного застосунку.

4. Розробити алгоритм виділення неправильно написаних літер у слові при наявності еталону.

5. Розробити структуру даних для збереження даних про користувача та алгоритм для генерації диктантів з урахуванням знань користувача.

6. Розробити структуру даних для збереження результату диктанту.

7. Створити головне меню застосунку.

8. Створити екран для написання диктанту.

9. Створити екран для відображення результату диктанту та для виведення загального списку написаних диктантів.

10. Створити екран для пошуку за словами покажчика Українського правопису.

11. Створити екран для налаштування параметрів програми.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблений додаток повинен мати наступний функціонал:

– диктування за допомогою синтезатора голосу;
– повна перевірка знання орфографії;
– незалежність від наявності підключення до інтернету.

Ключовою особливістю даної роботи повинне бути саме використання випадково згенерованих на основі інформації про користувача диктантів. Це дозволятиме виявляти теми, що потребують повторення, та спонукати користувача до навчання. Кожен новий диктант повинен бути унікальним, і генеруватися персонально для користувача.

В програмі повинна бути наявна система збереження результатів, за допомогою якої можна переглянути свій прогрес в навчанні.

1.5.2. Вимоги до інформаційної безпеки

Основні вимоги до інформаційної безпеки мобільного застосунку:

– цілісність даних;
– доступність інформації для користувачів.

Для надійної роботи даного програмного забезпечення необхідно дотримуватися таких факторів:

− використання ліцензійного ПЗ;
− захист від зловмисних програм;
− архівація даних на сервері;
− безперервний доступ до мережі.

1.5.3. Вимоги до складу та параметрів технічних засобів

Даний додаток призначений для використання на мобільному пристрої (наприклад, смартфон), що відповідає таким вимогам:

- операційна система Android версії 5.1 і вище;
- оптимальна роздільна здатність дисплею – 1280*720;
- доступ до мережі Інтернет;
- встановлений синтезатор голосу Google

1.5.4. Вимоги до інформаційної та програмної сумісності

Мобільний додаток має бути розроблений на кросплатформовій мові програмування, завдяки чому йому можна буде використовувати під різними платформами (Android , Windows, IOS тощо) та мати засоби для доступу до мережі Інтернет.

Програма повинна являти собою самостійний виконуваний модуль, бути структурована і за коментована

Додаток має стабільно працювати на усіх останніх версіях ANDROID.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Даний мобільний додаток призначений для адаптація людей до правил оновленого Українського правопису.

Розроблений додаток має наступний функціонал:

– диктування за допомогою синтезатора голосу;
– повна перевірка знання орфографії;
– незалежність від наявності підключення до інтернету.

Ключовою особливістю даної роботи є саме використання випадково згенерованих на основі інформації про користувача диктантів. Це дозволяє виявляти теми, що потребують повторення, та спонукає користувача до навчання. Кожен новий диктант є унікальним, і генеруватися персонально для користувача.

В програмі наявна система збереження результатів, за допомогою якої можна переглянути свій прогрес в навчанні.

2.2. Опис застосованих математичних методів

В даній програмі вирішується задача порівняння слів в різних файлах. Для її вирішення використовується завдання пошуку послідовності, яка є підпослідовністю кількох послідовностей

Пошук найдовшої спільної підпослідовності (англ. longest common subsequence, LCS) - це завдання пошуку послідовності, яка є підпослідовністю кількох послідовностей (зазвичай - двох).

Реалізація даного методу базується на основі динамічного програмування.

Динамічне програмування є водночас і методом математичної оптимізації і методом комп'ютерного програмування. В обох контекстах воно використовує

підхід спрощення пошуку розв'язку складної задачі, розбиттям її на простіші підзадачі, часто методом рекурсії. Хоча деякі задачі не можуть бути розв'язані таким чином, рішення, які охоплюють кілька точок у часі дійсно часто розбиваються рекурсивно на підзадачі. Белман називав це принципом оптимальності. Подібно до цього, в комп'ютерних науках про проблему, яка може бути розбита на підзадачі рекурсивно, говорять що вона має оптимальну підструктуру.

Якщо підзадачі можуть бути вкладеними рекурсивно всередині більших задач, так що методи динамічного програмування можуть бути застосовні, то існує залежність між розв'язком загальної задачі, і розв'язком підзадач .В методах оптимізації це відношення виражається рівнянням Беллмана.

У динамічному програмуванні для керованого процесу серед множини усіх допустимих управлінь шукають оптимальне у сенсі деякого критерію тобто таке яке призводить до екстремального (найбільшого або найменшого) значення цільової функції - деякої числової характеристики процесу. Під багатоступеневістю розуміють або багатоступеневу структуру процесу, або розподілення управління на ряд послідовних етапів (ступенів, кроків), що відповідають, як правило, різним моментам часу. Таким чином, в назві «Динамічне програмування» під «програмуванням» розуміють «ухвалення рішень», «планування», а слово «динамічне» вказує на суттєве значення часу та порядку виконання операцій в процесах і методах, що розглядаються.

З точки зору математичної оптимізації, динамічне програмування полягає в спрощенні знаходження загального оптимального розв'язку, шляхом пошуку розв'язків в підзадачах, отриманих розбиттям задачі на послідовні проміжки часу. Це виражається у визначенні послідовності значень функцій V_1, V_2, …, V_n, з аргументом y, котрий позначає стан[en] системи в моменти часу і від 1 до n.

Визначенням $V_n(y)$ є значення, отримане в стані y в кінцевий момент часу n.

Значення V_i в попередні моменти часу i = n −1, n − 2, …, 2, 1 можуть бути знайдені рухаючись назад, використовуючи рекурсивну залежність, названу рівняння Беллмана.

Для i = 2, …, n, значення V_{i-1} для будь-якого стану у визначається з V_i через максимізацію значення простої функції (зазвичай, суми) виграшу від рішення в момент часу i − 1 і функції V_i у новому стані системи, якби це рішення було втілено.

Оскільки V_i вже було розраховано для потрібних станів, то вище наведена операція забезпечує необхідне оптимальне значення V_{i-1} для цих станів. Нарешті, V_1 як початковий стан системи є значенням оптимального рішення.

Оптимальне значення змінних рішення може бути відновлене одне за одним виконанням обернених у часі обчислень.

2.3. Опис використаних технологій та мов програмування

2.3.1. Вибір платформи для створення застосунку

Розробляти застосунок було вирішено під операційну систему Android, що розроблена компанією Google. Вона була вибрана не просто так. По-перше вона є однією із найпопулярніших [18], збільшуючи кількість користувачів, які зможуть скористатися цим додатком. А по-друге, пристрої на базі цієї операційної системи є портативними (телефони, планшети), що дозволяє в будь-який момент скористатися додатком, та перевірити свій рівень знань. Це збільшить ефективність додатку.

Характеристики платформи Android:

1. Інструментарій. Платформа легко пристосовується для використання VGA, бібліотек двовимірної і тривимірної графіки, розроблених на основі OpenGL ES 1.0-3.1 специфікації, традиційних інструментаріїв для смартфонів.
2. Бази даних. SQLite для структурованих даних.
3. Технології зв'язку. Android підтримує багато технологій, що забезпечують зв'язок, у тому числі: GSM, Bluetooth, EDGE, 3G та WiFi.

4. Обмін повідомленнями. Для обміну повідомленнями доступні як SMS, так і MMS сервіси, у тому числі й потокові повідомлення.

5. Браузери. На Android доступний браузер, розроблений на основі SCRUM framework.

6. Java Virtual Machine. Програми, написані на Java, можна скомпілювати в Dalvik байткод і виконувати на Dalvik virtual machine, яка являє собою розроблену спеціально для використання на мобільних пристроях віртуальну машину, незважаючи на те, що не є стандартною Java Virtual Machine.

7. Підтримка медіа. Android підтримує такі формати для аудіо/відео даних та зображень: MPEG-4, H.264, MP3, та AAC, AMR, JPG, PNG, GIF.

8. Підтримка нестандартного обладнання. Android підтримує відеокамери, фотоапарати, дотикові екрани, GPS, компаси, акселерометри, та прискорювачі 3D графіки.

9. Середовище розробки. Офіційним середовищем розробки є Android Studio, створене на базі IntelliJ IDEA. Містить емулятор, засоби відлагодження, профілювання пам'яті та швидкодії. Також доступні плагіни для IntelliJ IDEA, Eclipse та NetBeans.

Деякі користувачі відзначають, що Android проявляє себе краще одного зі своїх конкурентів, Apple iOS, в ряді особливостей, таких як веб-серфінг, інтеграція з сервісами Google і інших. Також Android, на відміну від iOS, є відкритою платформою, що дозволяє реалізувати на ній більше функцій.

Незважаючи на початкову заборону на установку програм з «неперевірених джерел» (наприклад, з карти пам'яті), це обмеження відключається штатними засобами в налаштуваннях пристрою, що дозволяє встановлювати програми на телефони та планшети без інтернет-підключення (наприклад, користувачам, які не мають Wi-Fi-точки доступу і не бажають витрачати гроші на мобільний інтернет, який зазвичай коштує дорого), а також дозволяє будь-кому безкоштовно писати програми для Android і тестувати на своєму пристрої.

Android доступний для різних апаратних платформ, таких як ARM, MIPS, x86. У версії 4.3 з'явилась підтримка багатокористувацького режиму.

2.3.2. Вибір середовища для розробки застосунку

Android Studio - інтегроване середовище розробки (IDE) для платформи Android. Середовище побудоване на базі вихідного коду продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains [14]. Воно розробляється компанією Google і адаптоване під розв’язування типових завдань під час розробки на обраній платформі. Через це воно найкраще підходить для розробки.

Android Studio - інтегроване середовище розробки виробництва Google, за допомогою якої розробникам стають доступні інструменти для створення додатків на платформі Android OS. Android Studio можна встановити на Windows, Mac і Linux.

IDE можна завантажити і користуватися безкоштовно. У ній присутні макети для створення UI, з чого зазвичай починається робота над додатком. В Studio містяться інструменти для розробки рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

Середа Android Studio призначена як для невеликих команд розробників мобільних додатків (навіть в кількості однієї людини), або ж великих міжнародних організацій з GIT або іншими подібними системами управління версіями. Досвідчені розробники зможуть вибрати інструменти, які більше підходять для масштабних проектів. Рішення для Android розробляються в Android Studio з використанням Java або C ++. В основі робочого процесу Android Studio закладений концепт безперервної інтеграції, що дозволяє відразу ж виявляти наявні проблеми. Тривала перевірка коду забезпечує можливість ефективного зворотного зв'язку з розробниками. Така опція дозволяє швидше

опублікувати версію мобільного застосування в Google Play App Store. Для цього є також підтримка інструментів LINT, Pro-Guard і App Signing.

За допомогою засобів оцінки продуктивності визначається стан файлу з пакетом прикладних програм. Візуалізація графіки дає можливість дізнатися, чи відповідає додаток орієнтиру Google в 16 мілісекунд. За допомогою інструменту для візуалізації пам'яті розробник дізнається, коли його застосування буде використовувати занадто багато оперативної пам'яті і коли відбудеться «прибирання сміття». Інструменти для аналізу батареї показують, яке навантаження припадає на пристрій.

Android Studio сумісна з платформою Google App Engine для швидкої інтеграції в хмарі нових API і функцій. У середовищі розробки ви знайдете різні API, такі як Google Play, Android Pay і Health. Є підтримка всіх платформ Android, починаючи з Android 1.6. Є варіанти Android, які істотно відрізняються від версії Google Android. Найпопулярніша з них це Amazon Fire OS. В Android Studio можна створювати APK для цієї ОС. Підтримка Android Studio обмежується онлайн-форумами.

2.3.3. Вибір джерела слів для диктування

Джерелом слів для диктантів було обрано сам Український правопис, який має предметний покажчик [13] (рис. 2.1). У ньому зібрані слова, що містять основні орфограми із посиланнями на конкретні правила, які потрібно знати щоб правильно написати ці слова. Таким чином отримуємо задачу видобути ці слова, а також правила, для того щоб можна було обґрунтувати користувачеві його помилки.

БУКВЕНІ ПОЗНАЧЕННЯ ДЕЯКИХ ПРИГОЛОСНИХ ЗВУКІВ

§ 5. Буква Г

Буква **г** передає на письмі глотковий щілинний приголосний звук як в українських словах (*берегти́, вого́нь, га́дка, гука́ти, дорогі́й, жага́, згин, кри́га, могу́тній, пагіне́ць*), так і в загальних та власних назвах іншомовного походження (на місці **h, g**) (*абориге́н, агіта́ція, агре́сія, бага́ж, болга́рин, брига́да, газе́та, генера́л, геоло́гія, горизо́нт, гра́мота, делега́т, кілогра́м, логопе́д, магази́н, педаго́г, фотогра́фія; Єва́нгеліє, Гоме́р, А́нглія, Га́ага*) (див. ще § 122).

§ 6. Буква Ґ

Буква **ґ** передає на письмі задньоязиковий зімкнений приголосний:

1. в українських та в давно запозичених і зукраїнізованих словах: *а́ґрус, та́ва, та́здра, тандж, та́нок, татуно́к, твалт, те́тати, тедзь, телтота́ти, телтоті́ти, тертелі, тертота́ти, тертоті́ти, ти́тнути, тирли́та, тлей, тніт* (у лампі), *то́тель-мо́тель, тонт(а), трасува́ти, тра́ти* (іменник), *тре́чний, тринджо́ли, трунт, ту́дзик, ту́ля, тура́льня, джиту́н, дзи́та, дзи́тлик, дрита́ти* і *дри́тати, ремита́ти* тощо та в похідних від них: *а́ґрусовий, газдува́ти, твалтува́ти, те́ртіт, тратча́стий, трунтови́й, трунтува́ти(ся), ту́дзиковий, ту́лька, прота́вити* і под.

2. у власних назвах — топонімах України: *Ґорта́ни* (гірський масив), *Ґоро́нда, У́тля* (села на Закарпатті), у прізвищах українців: *Ґалата́н, Ґалято́вський, Ґе́ник, Ґерза́нич, Ґерда́н, Ґжи́цький, Ґи́та, Ґо́та, Ґо́йдич, Ґо́нта, Ґри́та, Ґудзь, Ґу́ла, Лома́та.*

§ 7. Апостроф

Роздільність вимови **я, ю, є, ї** та попереднього твердого приголосного на письмі позначаємо апострофом.

Апостроф пишемо перед **я, ю, є, ї:**

1. Після букв на позначення губних приголосних **б, п, в, м, ф**: *б'ю, п'ять, п'є, в'я́зи, солов'ї́, м'я́со, рум'я́ний,* (на) *тім'ї́, жира́ф'ячий, мереф'я́нський; П'я́ста, В'ячесла́в, Дем'ян, Максим'ю́к, Стеф'ю́к.*

Примітка. Апостроф не пишемо, коли перед буквою на позначення губного звука є інша буква (крім **р**), що належить до кореня (основи): *дзвя́кнути, духмя́ний, мавпячий,*

7

Рис. 2.1. Офіційний документ Українського правопису

2.3.4. Вибір мови програмування для обробки даних

Для видобування інформації із тексту Українського правопису треба створити спеціальну програму, яка автоматизувала б цей процес. Для її написання в якості мови програмування було обрано Python, адже він якнайкраще підходить для виконання поставленої цілі – отримання набору даних.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах.

В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

– чистий синтаксис (для виділення блоків слід використовувати відступи);

– переносність програм (що властиве більшості інтерпретованих мов);

– стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);

– можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

– стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане мовою Python;

– зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини,

у діалоговому режимі може використовуватися як потужний калькулятор);

– відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата Стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C + + (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Код програми, написаний на цій мові, максимізує відношення корисної роботи, що він виконує, до його кількості. Сама програма на цій мові складається із модулів, кожен з яких виконує певну функцію [19]. Таким чином розробнику залишається лише об’єднати функціонал декількох модулів для отримання бажаного результату.

Python портованна і працює майже на всіх відомих платформах - від КПК до мейнфреймів. Існують порти під Microsoft Windows, всі варіанти UNIX (включаючи FreeBSD та GNU/Linux), Plan 9, Mac OS та Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 та навіть OS/390, Symbian та Android[17].

У міру старіння платформи її підтримка в основній гілці мови припиняється. Наприклад, з версії 2.6 припинена підтримка Windows 95, Windows 98 та Windows ME. Однак на цих платформах можна використовувати

попередні версії Python - спільнота активно підтримує версії Python починаючи від 2.3 (для них виходять виправлення).

При цьому, на відміну від багатьох портованих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Навіть більше, існує спеціальна версія Python для віртуальної машини Java - Jython, що дозволяє інтерпретатору виконуватися на будь-якій системі, яка підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на ньому. Також кілька проєктів забезпечують інтеграцію з платформою Microsoft.NET, основні з яких - IronPython та Python.Net.

Багата стандартна бібліотека є однією з привабливостей мови Python. Тут є засоби для роботи з багатьма мережевими протоколами та форматами Інтернету, наприклад, модулі для написання HTTP-серверів та клієнтів, для розбору та створення поштових повідомлень, для роботи з XML, тощо. Набір модулів для роботи з операційною системою дозволяє писати крос-платформні застосунки. Існують модулі для роботи з регулярними виразами, текстовими кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, серіалізацією даних, юніт-тестуванням та ін.

2.3.5. Технологія синтезу мовлення

Синтез мовлення (СМ) - перетворення друкарського тексту на мовний сигнал (в широкому сенсі - відновлення форми мовного сигналу за його параметрами.

Комп'ютерна система, що здійснює синтез мовлення, називається синтезатором мовлення, (СМ) та може бути побудованою на основі програмного чи апаратного рішення.

Система, що спроможна виконувати конвертування нормального друкованого тексту на аудіо в реальному часі називається текст-у-мовлення (ТУМ, англ. text-to-speech, TTS).

Всі способи синтезу мовлення можна підрозділити на три групи:

– параметричний синтез;
– конкатенативний, або синтез компіляції (компілятивний);
– синтез за правилами.

Параметричний синтез мовлення є кінцевою операцією в вокодерних системах, де мовний сигнал представляється набором невеликого числа параметрів, що безперервно змінюються. Параметричний синтез доцільно застосовувати в тих випадках, коли набір повідомлень обмежений і змінюється не дуже часто. Перевагою такого способу є можливість записати мовлення для будь-якої мови і будь-якого диктора. Якість параметричного синтезу може бути дуже високою (залежно від ступеня стиснення інформації в параметричному уявленні). Проте параметричний синтез не може застосовуватися для довільних, заздалегідь не заданих повідомлень.

Компіляційний синтез зводиться до складання повідомлення із заздалегідь записаного словника початкових елементів синтезу. Розмір елементів синтезу не менше слова. Очевидно, що зміст повідомлень, що синтезуються, фіксується обсягом словника. Як правило, число одиниць словника не перевищує декількох сотень слів. Основна проблема в компілятивному синтезі — обсяги пам'яті для зберігання словника. У зв'язку з цим використовуються різноманітні методи стиснення/кодування мовного сигналу. Компілятивний синтез має широке практичне застосування. За кордоном різноманітні пристрої (від військових літаків до побутових пристроїв) оснащуються системами мовної відповіді. У нашій країні системи мовної відповіді до недавнього часу використовувалися в основному в області військової техніки, зараз вони знаходять все більше застосування в повсякденному житті, наприклад, в довідкових службах операторів стільниковому зв'язку при отриманні інформації про стан рахунку абонента.

Повний синтез мовлення за правилами (або синтез за друкарським текстом) забезпечує управління всіма параметрами мовного сигналу і, таким чином, може генерувати мовлення за заздалегідь невідомим текстом. В цьому

разі параметри, отримані при аналізі мовного сигналу, зберігаються в пам'яті так само, як і правила з'єднання звуків у слова і фрази. Синтез реалізується шляхом моделювання мовного тракту, застосування аналогової або цифрової техніки. Причому в процесі синтезування значення параметрів і правила з'єднання фонем вводять послідовно через певний часовий інтервал, наприклад 5-10 мс. Метод синтезу мовлення за друкарським текстом (синтез за правилами) ґрунтується на запрограмованому знанні акустичних і лінгвістичних обмежень і не використовує безпосередньо елементів людської мови. У системах, заснованих на цьому способі синтезу, виділяється два підходи. Перший підхід направлений на побудову моделі мовотворчої системи людини, він відомий під назвою артикуляторного синтеза. Другий підхід - формантний синтез за правилами. Розбірливість і натуральність таких синтезаторів може бути доведена до величин, порівнянних з характеристиками природної мови.

Синтез мовлення за правилами з використанням попередньо запам'ятованих відрізків природної мови, - це різновид синтезу мовлення за правилами, яка набула поширення у зв'язку з появою можливостей маніпулювання мовним сигналом в оцифрованій формі. Залежно від розміру початкових елементів синтезу виділяються такі види синтезу:

– мікросегментний (мікрохвильовий);
– алофонічний;
– дифонний;
– напівскладовий;
– складовий;
– синтез з одиниць довільного розміру.

Зазвичай як такі елементи використовуються напівсклади - сегменти, що містять половину приголосного і половину суміжного з ним голосного. При цьому можна синтезувати мову за заздалегідь не заданим текстом, але важко управляти інтонаційними характеристиками. Якість такого синтезу не відповідає якості природної мови, оскільки на границях зшивання дифонів часто виникають спотворення. Компіляція мовлення із заздалегідь записаних

словоформ також не розв'язує проблеми високоякісного синтезу довільних повідомлень, оскільки акустичні і просодичні (тривалість і інтонація) характеристики слів змінюються залежно від типу фрази і місця слова у фразі. Це положення не міняється навіть при використанні великих обсягів пам'яті для зберігання словоформ.

2.3.6. Вибір бібліотеки для диктування слів

Для диктування слів користувачеві вирішено застосовувати TextToSpeech API [20], вбудований в операційну систему Android. Це дозволяє застосовувати сторонні бібліотеки, зменшуючи розмір додатка, використовувати систему синтезатора голосу на всіх версіях Android, починаючи з версії 1.6.

Google Text-to-Speech (скорочено Google TTS) також Google speech synthesis (укр. Синтез мовлення від Google) - застосунок-читач екрану створений Android, Inc. для операційної системи Android у 2013 році. Цей застосунок використовує технологію «синтезу мовлення» (також відому як текст-у-мовлення) аби читати вголос текст з екрану.

Станом на 2017 рік, підтримуються такі мови: бенгальська (Банґладеш), бенгальська (Індія), кантонська (Гонконг), чеська, данська, нідерландська, англійська (Австралія), англійська (Велика Британія), англійська (Індія), англійська (США), фінська, французька, німецька, гінді, угорська, індонезійська, італійська, японська, кхмерська, корейська, мандаринська (Тайвань), мандаринська (Китай), непальська, норвезька, польська, португальська (Бразилія), російська, сінхала, іспанська (Іспанія), іспанська (США), шведська, тайська, турецька, українська та в'єтнамська.

Завдяки функції синтезу мовлення від Google, Android застосунки можуть озвучувати текст на екрані. Зокрема синтез мовлення можуть використовувати: Google Play Книги (щоб читати вголос книги), Google перекладач (щоб озвучувати переклади та вимову слів), TalkBack та застосунки зі спеціальними можливостями - щоб надавати вам голосові підказки на пристрої. Для кожної

мови користувач мусить встановити голосовий пакет-даними аби застосунок почав повноцінно функціонувати.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Алгоритм перевірки введених користувачем слів

Для того, щоб перевірити чи правильно користувач ввів слово, недостатньо просто порівняти два рядка, так як, наприклад, користувач міг випадково поставити два пробіли натомість одного. Це не є помилкою, але порівняння не зарахує це слово. Тож, в даній роботі створені правила, за якими програма буде «виправляти» введене користувачем слово:

1. Видалення пробілів на початку та в кінці слова, до першого символу.
2. Заміна подвоєних пробілів на одинарні.
3. Видалення всіх символів, що не є літерами української мови, числами, дефісом або лапками.
4. Заміна «неправильно» написаних лапок.

У випадку, якщо користувач зробив помилку, програмі потрібно сказати йому, в якому місці слова він її зробив. Просто проходячись по обох рядках та порівнюючи їх, це зробити неможливо. Таким чином, потрібно знайти кращий алгоритм. Насправді подібна задача є класичною, і носить назву «Пошук найбільшої спільної підпослідовності» [11].

Пошук найдовшої спільної це класична задача інформатики, яка має застосування, зокрема, в задачі порівняння текстових файлів (утиліта diff), а також у біоінформатиці (рис. 2.2).

		A	B	C	B
	0	0	0	0	0
D	0	← 0	← 0	← 0	← 0
C	0	← 0	← 0	↖ 1	← 1
B	0	← 0	↖ 1	← 1	↖ 2
A	0	↖ 1	← 1	← 1	↑ 2

Рис. 2.2. Пошук найдовшої спільної підпослідовності

Оригінальна умова цієї задачі є такою:

– підпослідовність можна отримати з деякої послідовності, якщо видалити з неї деяку множину елементів (можливо, порожню). Наприклад, BCDB є підпослідовністю послідовності ABCDBAB. Також вона буде підпослідовністю послідовності XBXCDXBX;

– послідовність Z є спільною підпослідовність послідовностей X і Y, якщо Z є підпослідовністю як X, так і Y;

– потрібно для двох послідовностей X і Y знайти спільну підпослідовність найбільшої довжини.

Подібні задачі вирішуються за допомогою алгоритмів динамічного програмування [2] – методів оптимізації, пристосованих до операцій, в яких процес прийняття рішень може бути розділений на окремі кроки.

Таким чином, для того щоб отримати рішення початкового завдання, треба спочатку вирішити менші. Для того щоб отримати із менших рішень рішення початкового завдання, використовують функцію переходу.

Для нашого випадку вона виглядає так :

$$f(n_1, n_2) = \begin{cases} 0, & n_1 = 0 \vee n_2 = 0 \\ f(n_1 - 1, n_2 - 1) + 1, & s[n_1] = s[n_2] \\ max(f(n_1 - 1, n_2), f(n_1, n_2 - 1)), & s[n_1] \neq s[n_2] \end{cases}, \quad (2.1)$$

де f(x,y) - це найбільша спільна підпослідовність першого рядку, обрізаного до x-того символу, та другого рядку, обрізаного відповідно до y-того символу.

Для побудови підпослідовності в наявний алгоритм для кожної задачі додають запам'ятовування тих підзадач, через які вона вирішується. Наступною дією, починаючи з останнього елемента, піднімаємося до початку за напрямками, заданим першим алгоритмом, і записуємо символи в кожній позиції. Це і буде відповіддю в цій задачі.

Розглянемо два рядки (або числові послідовності) – A і B. Нехай перший рядок складається з n символів $a_0..a_{n-1}$, другий рядок складається з m символів $b_0…b_{m-1}$. Підпослідовність цього рядка (послідовності) називається деякий підмножина символів початкового рядка, наступних у тому ж порядку, в якому вони йдуть в заданій стрічці, але не обов'язково підряд. Якщо в рядку n символів, то у неї 2^n різних підпослідовностей: кожен із n символів рядка може або входити, або не входити в будь-яку обрану підпослідовність. Порожня підпослідовність не містить жодного елемента і також є підпослідовність будь-якого рядка.

Тож, щоб виконати такий алгоритм, достатньо створити двовимірний масив, в кожній комірці якого буде міститись значення f(x, y), і, проходячись по всім коміркам масиву, заповнювати їх, доки не дійдемо до комірки із відповіддю на початкове запитання.

Тепер, коли наявна спільна для двох слів частина,можна виділити їх різницю, та візуально відобразити. Таким чином, можна не тільки перевіряти правильність написання слова, а й показати користувачеві, де саме він зробив помилку. Приклад реалізації даного алгоритму наведено в лістингу 1.

Лістинг 1. Реалізація алгоритму перевірки введених користувачем слів

```
def Check(args: list, argst: str):
    if args.__len__() == 1:
        print("Checking parsed words...")
        wordscheck.WordsCheck(WordsDictionary, SaveVariables)
        print("Checking ended.")
    elif args.__len__() == 2:
        if args[1] == "show":
            print("Computing...")
            __unchecked_num__ = 0
            for word in WordsDictionary.keys():
                if word not in wordscheck.CheckedWords:
                    __unchecked_num__ += 1
            print(f"Number of checked words =
{min(wordscheck.CheckedWords.__len__(), WordsDictionary.__len__())}.\n"
                  f"Number of unchecked words = {__unchecked_num__}.")
    else:
        print("Invalid number of arguments!")
```

2.4.2. Видобування слів із додатку Українського правопису

Для вирішення цієї задачі було вирішено написати окрему програму на мові Python, яка заздалегідь видобудує всі слова та збереже в окремий файл. Під час видобудування програма повинна зчитувати слова і параграфи правил, що їх стосуються, з предметного покажчика Українського правопису. Для правильного зчитування подібного тексту, потрібно правильно розуміти його місцезнаходження на сторінці та шрифт.

Можна сформулювати алгоритм, за яким буде працювати наш видобудувач :

1. Пропускаємо текст жирний текст, та текст зі шрифтом «Calibri».

2. Текст написаний курсивом є новим словом.

3. Все інше є частиною правила, що відповідає поточному слову.

4. Зчитування рядків відбувається зверху донизу спочатку в першій половині сторінці, а потім у другій, зберігаючи порядок слів у рядку.

Реалізація даного алгоритму наведена в лістингу 2.

Лістинг 2. Алгоритм видобудування слів та збереження в окремий файл

```
def Parse(args: list, argstr: str):
    """Function, that starts parsing process"""
    global IsLocationChanged, PdfFileLocation, PdfFile, WordsDictionary
    if args.__len__() > 1:
        if args[1] == "show":
            print(f"Parsed pages : {ParsedPages}.\n"
                  f"Total number of parsed words = {WordsDictionary.keys().__len__()}")
            #print(WordsDictionary)
        elif args[1] == "save":
            print("Saving words...")
            file = open(ResultFileLocation, "w")
            for (word, rules) in WordsDictionary.items():
                file.write(word + utils.__RulesSplitter__)
                file.write(utils.__RulesSplitter__.join(rules) +
                           utils.__WordBlockSplitter__)
            file.close()
            print("Words saved!")
        elif args[1] == "load":
            location = utils.GetPath(args[2], argstr)
            print(f"Loading words from \"{location}\"...")
            file = open(location, "r")
            WordsDictionary = dict()
            for word_info in file.read().split(utils.__WordBlockSplitter__):
```

```
            spl = word_info.split(utils.__RulesSplitter__)
            WordsDictionary[spl[0]] = spl[1:]
        print("Words loaded!")
    elif PdfFileLocation is None:
        print("Error : invalid pdf file location!")
        '''elif ResultFileLocation is None:
            print("Error : invalid output file location!")'''
    else:
        if IsLocationChanged:
            PdfFile = pdfparse.PdfFile(PdfFileLocation)
            IsLocationChanged = False
        # noinspection PyUnboundLocalVariable
        len = PdfFile.__len__()
        if args.__len__() == 2:
            start = int(args[1])
            if start < len:
                if start not in ParsedPages:
                    wordsparse.ParseWordsFromStr(
                        PdfFile.GetWordStr(start),
                        WordsDictionary)
                    ParsedPages.append(start)
            else:
                print(f"Invalid page number ({start}, when max value is {len - 1}).")
                return
        elif args.__len__() == 3:
            start = int(args[1])
            stop = int(args[2])
            if start <= stop < len:
                for i in range(start, stop + 1):
                    if i not in ParsedPages:
```

```
                SaveVariables()
                wordsparse.ParseWordsFromStr(
                    PdfFile.GetWordStr(i),
                    WordsDictionary)
                ParsedPages.append(i)
            else:
                print(f"Invalid page range [{start}; {stop}].")
                return
        print("Parsing finished.")
    else:
        print("Incorrect number of arguments!")
        return
```

Текст видобутий із PDF файлу в такий спосіб, на жаль, містить у собі різні артефакти, що роблять неможливим його використання без його попередньої обробки. Тож, для того щоб це виправити, вирішено створити іще один алгоритм.

Його завданням буде виявлення та виправлення ось таких артефактів :

1. Рядок містить помилки що не дозволяють зчитати з нього слово;
2. Слово містить невідомі символи що не дозволяють його використовувати для перевірки користувача;
3. Слово є скороченням або абревіатурою, що не можна диктувати користувачу, адже тоді буде не зрозуміло, що саме потрібно написати.

Так як ці артефакти є досить складними у виправленні, цю задачу було вирішено перекласти на людину, проте програма при цьому буде виконувати не менш важливу задачу: виявлення цих артефактів. Адже справді, якщо людина вручну буде переглядати весь той великий набір тексту, то в результаті з великою вірогідністю спрацює людський фактор і з’являться інші помилки.

Розглянемо перший артефакт зі списку. Для того, щоб його попередити потрібно перевіряти, чи правильно спрацював алгоритм видобування і

отримане слово містить всю необхідну інформацію (наприклад правила, які його стосуються). У випадку, якщо збіг не відбувається, потрібно повідомити про це користувачу і дати йому вирішити, що потрібно робити з цим словом.

Другим артефактом є невідомі символи, що в тексті Українського правопису могли позначати букву з наголосом. Знову ж таки, без допомоги користувача неможливо з'ясувати, на що схожий цей символ, і як потрібно його інтерпретувати, тож просто робимо перевірку для кожного символу, який зустрічаємо, чи входить він до списку дозволених символів, і в іншому випадку повідомляємо про це користувача.

Третім артефактом є абревіатури та скорочення, приклади яких містить Український правопис. Їх не можна використовувати для диктанту, тож потрібно їх або відредагувати, або видалити зі списку слів. Насправді, виявити такі слова достатньо легко, адже завжди разом із ними йдуть якісь символи на позначення абревіатури або скорочення: це або дві великі літери, що йдуть одна після одної, або крапка, або дужки і так далі. Таким чином, зіставивши список всіх символів що можуть означати артефакт, можна виявити всі підозрілі слова. Далі знову ж таки потрібно запитати користувача про те, потрібно робити з артефактом.

Приклад реалізації алгоритму виявлення деяких артефактів наведено в лістингу 3.

Лістинг 3. Виявлення та видалення артефактів з файлу

```
__WarningMess__ = "<! Warning>"
def GetWarning() -> str:
    if SupportsColors:
        return colorit.background(__WarningMess__, colorit.Colors.yellow)
    else:
        return "\n" + __WarningMess__
def GetPath(arg, arg_str):
    return (
```

```
        arg if arg[0] != '\"'
        else re.search(r'\"(.*)\"', arg_str).group(1)
    ).replace("\\", "/")
__RulesSplitter__ = "~"
__WordBlockSplitter__ = "^"
res = ""
while res not in ["т", "н"]:
    res = input("Enable terminal? (т|н) : ")
SupportsColors = res == "т"
```

Кожне слово, разом із його правилами, що пройшло всі три етапи перевірки, додається до списку результатів. Таким чином, отримуємо список слів, які можна використовувати для укладання диктантів. Інформацію про правила, яких стосується слово, буде використовуватися для спеціалізації диктанту відповідно до проблемного правила користувача (рис. 2.2).

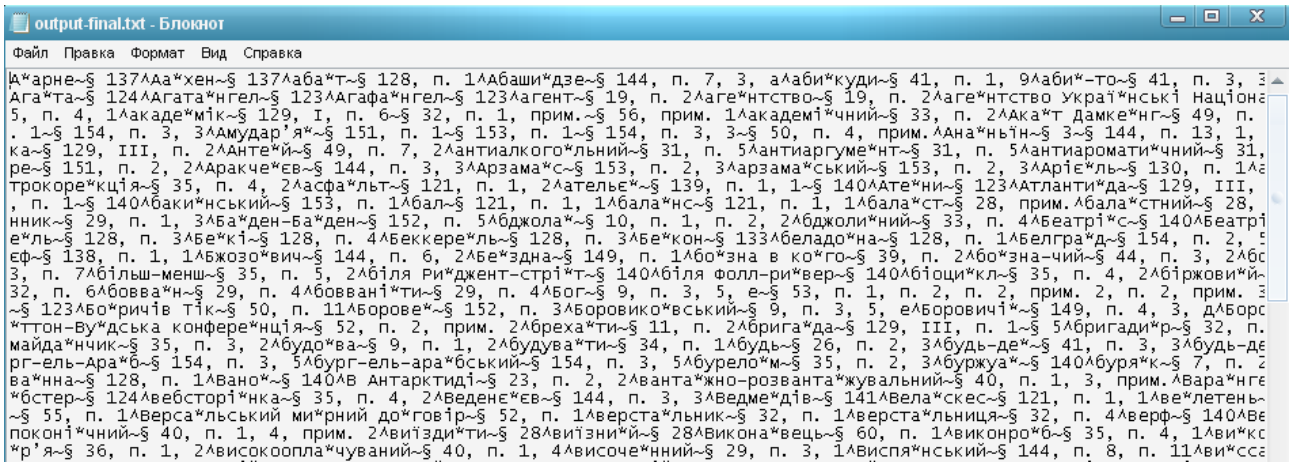

Рис. 2.2. Текст, видобутий із PDF файлу

2.4.3. Створення персоналізованих диктантів

Для того, щоб спонукати користувача вивчати невідомі йому теми, диктанти треба складати з слів, які відповідають цим невідомим темам. Наприклад, якщо користувач погано знає, коли потрібно ставити знак м’якшення, то йому треба диктувати саме слова із цією літерою. Таку процедуру потрібно продовжувати, допоки користувач не покаже, що тему він знає.

Якщо ж користувач вперше користується даною системою, то інформація про нього відсутня. Тому доцільно при першому диктанті вибирати слова для диктування випадково, допоки не буде зроблена помилка. Так системою буде виявлено проблемну тему.

Для збереження інформації про користувача вирішено використовувати вбудовану в Android структуру даних – геш-таблицю [1], що дозволяє записувати та знаходити значення пари ключ-значення.

Геш-таблиця - структура даних, що реалізує інтерфейс асоціативного масиву, а саме, вона дозволяє зберігати пари (ключ, значення) і здійснювати три операції: операцію додавання нової пари, операцію пошуку і операцію видалення за ключем.

Важлива властивість геш-таблиць полягає в тому, що, при деяких розумних припущеннях, всі три операції (пошук, вставлення і видалення елементів) зазвичай виконується за час O(1). Але при цьому не гарантується, що час виконання окремої операції малий, з певною імовірністю час може бути сумірним із пошуком у списку. З ростом коефіцієнта заповнення таблиці ця імовірність, і, відповідно, середній час виконання операцій, ростуть. Тому при досягненні деякого значення коефіцієнта заповнення необхідно здійснювати перебудову індексу геш-таблиці: збільшити розміри масиву H і заново додати в порожню геш-таблицю всі пари.

Реалізація доступу до геш-таблиць методом ланцюжків:

1. Кожна комірка масиву H є вказівником на зв'язаний список

(ланцюжок) пар ключ-значення, відповідних одному і тому самому геш-значенню ключа. Колізії просто призводять до того, що з'являються ланцюжки довжиною більше одного елемента.

2. Операції пошуку або видалення елемента вимагають перегляду всіх елементів відповідного ланцюжка, щоб знайти в ньому елемент з заданим ключем. Для додавання нового елемента необхідно додати елемент в кінець або початок відповідного списку, і, у випадку якщо коефіцієнт заповнення стане занадто великим, збільшити розмір масиву H і перебудувати таблицю.

3. При припущенні, що кожний елемент може потрапити в будь-яку позицію таблиці H з однаковою ймовірністю і незалежно від того, куди потрапив будь-який елемент, пересічний час роботи операції пошуку елемента складає $\Theta(1 + \alpha)$, де α - коефіцієнт заповнення таблиці.

У нашому випадку ключем буде номер правила в Українському правописі, а значенням – коефіцієнт незнання, що прямо пропорційний кількості слів, написаних користувачем неправильно.

Якщо робиться помилка, то значення коефіцієнту відповідного правила ми збільшуємо на константу. Якщо ж користувач пише записує продиктоване слово правильно, то значення коефіцієнту ділиться на константу. Таким чином засвоєння нових правил з точки зору застосунку має такий вигляд :

1. Виявлення проблемної теми.
2. Примус користувача її вивчати.
3. Вивчення користувачем теми.
4. Перевірка, наскільки гарно користувач вивчив тему.

При цьому важливо, щоб перевірка тривала менше ніж виявлення, щоб не дати користувачеві заскучати.

В результаті отримуємо алгоритм та структуру даних, реалізуючи які в застосунку отримано диктанти, персоналізовані під конкретного користувача. Також, інформацію з цієї ж геш-таблиці можна використовувати для створення рекомендацій користувачеві: обрати три теми із найбільшим коефіцієнтом незнання, та вивести їх на екран. Саме ці теми йому потрібно краще вивчити.

2.4.4. Створення інтерфейсу програми

Для того, щоб покращити взаємодію користувача з додатком, а також щоб зробити інтерфейс сучасним, було вирішено використати Material Design (Матеріальний дизайн) [16] [17].

Матеріальний дизайн - принципи дизайну сайтів, програмного забезпечення і застосунків, а також правила дизайну інтерфейсів для операційної системи Android від компанії Google. Вперше представлений на конференції Google I/O 25 червня 2014 року. Ідея дизайну полягає в інтерфейсі, поведінка і вигляд якого наслідують правила поведінки і вигляду паперових карток в реальному житті. Інтерфейс перетворюється на живу, приємну людині, взаємодію.

Особливості:

– матеріальний дизайн за допомогою системи управління тінями створює візуальну ілюзію простору між застосунком та екраном пристрою;

– була збільшена реалістичність анімації за допомогою прискорення та пригальмовування рухів, пружного стрибання об'єктів;

– з'явилася об'єктна хореографія, при якій елементи здатні впливати на сусідні елементи.

В основі Material Design лежать чотири принципи:

1. Тактильні поверхні. Всі елементи інтерфейсу - це шари цифрового паперу. Вони розташовуються на різній висоті і відкидають тіні. Це допомагає користувачам відрізнити головні елементи від другорядних і робить інтерфейс інтуїтивно зрозумілим.

2. Поліграфічний дизайн. Логічно, що на цифровому папері потрібно писати цифровим чорнилом. Все, що зображено і написано на шарах-елементах, підпорядковується законам друкованого дизайну. Так можна акцентувати увагу користувача на потрібному елементі і позначити ієрархію інтерфейсу.

3. Усвідомлена анімація. Всі елементи, які є на екрані, не можуть просто

так з'являтися і зникати, - адже в реальному житті так не буває. Об'єкти плавно переходять один в інший і підказують користувачеві, як працює інтерфейс.

4. Адаптивний дизайн. Все це повинно працювати на будь-яких пристроях.

Таким чином, було вирішено використовувати матеріальний дизайн. Це досить сильно вплинуло на розмір та складність коду, до якого потрібно було додати опис анімацій та власних елементів управління.

В результаті отримали дійсно зручний інтерфейс, в якому використана мінімальна кількість тексту, а функції елементів управління інтуїтивно зрозумілі.

Об’єднавши інтерфейс та програмні реалізації описаних вище алгоритмів, отримано готовий до використання мобільний застосунок.

Розроблений додаток має наступний функціонал:

– диктування за допомогою синтезатора голосу;
– повна перевірка знання орфографії;
– незалежність від наявності підключення до інтернету.

Ключовою особливістю даної роботи є саме використання випадково згенерованих на основі інформації про користувача диктантів. Це дозволяє виявляти теми, що потребують повторення, та спонукає користувача до навчання. Кожен новий диктант є унікальним, і генеруватися персонально для користувача.

В програмі наявна система збереження результатів, за допомогою якої можна переглянути свій прогрес в навчанні.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Джерелом слів для диктантів було обрано Український правопис, що розміщений в файлі pravo.pdf, який має предметний покажчик. У ньому зібрані слова, що містять основні орфограми із посиланнями на конкретні правила, які

потрібно знати щоб правильно написати ці слова.

Кожне слово, разом із його правилами, що пройшло всі три етапи перевірки, додається до списку результатів у відповідний файл output-final.txt. Таким чином, отримуємо список слів, які можна використовувати для укладання диктантів. Інформацію про правила, яких стосується слово, буде використовуватися для спеціалізації диктанту відповідно до проблемного правила користувача.

Для того, щоб перевірити чи правильно користувач ввів слово, створені правила, за якими програма буде «виправляти» введене користувачем слово.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для розробки програмного додатку використано комп’ютер з наступними параметрами:

– підтримка 64-розрядного процесору та операційної системи;
– операційна система Windows 10 64bit;
– процесор Intel Core i5 3470 3.2 GHz/AMD X8 FX-8350 4 GHz;
– оперативна пам'ять 8GB ОЗУ;
– відеокарта nVidia GTX 650 з 1 Гб відеопам’яті, AMD HD7860 з 1 Гб відеопам’яті;
– DirectX версії 12;
– жорсткий диск мінімум в 72 GB.

2.6.2. Використані програмні засоби

Розроблений застосунок під операційну систему Android, з використанням інтегрованого середовища розробки (IDE) Android Studio . Для видобування інформації із тексту Українського правопису створена програма на мові програмування Python.. Для диктування слів користувачеві застосовано

TextToSpeech API, вбудований в операційну систему Android. Це дозволяє застосовувати сторонні бібліотеки, зменшуючи розмір додатка, використовувати систему синтезатора голосу на всіх версіях Android, починаючи з версії 1.6.

2.6.3. Виклик та завантаження програми

Для встановлення мобільного додатку на користувальницький пристрій потрібно завантажити інсталяційний файл Word_Dictant.apk (на пристрій або на карту пам’яті). Його треба відкрити за допомогою файлового менеджеру та діяти за інструкціям по встановленню на екрані пристрою (рис. 2.3, 2.4).

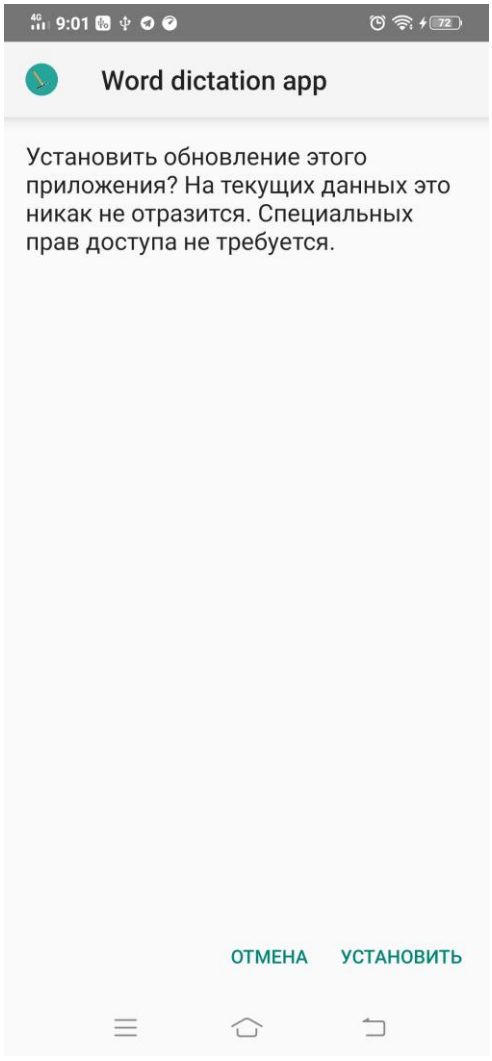

Рис. 2.3. Встановлення додатку на смартфон

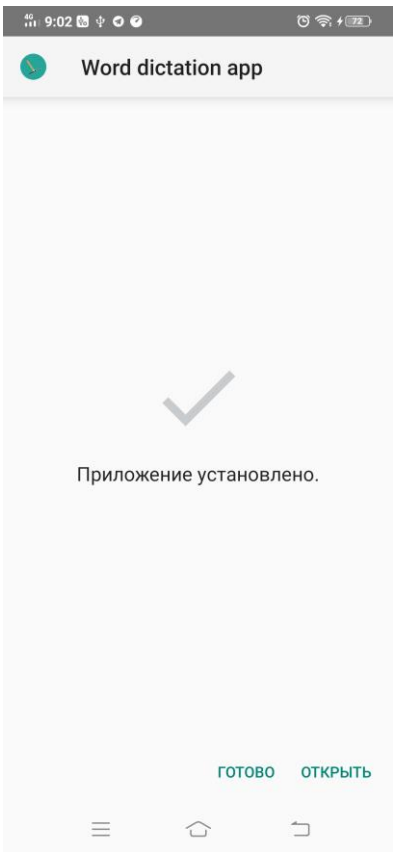

Рис. 2.4. Повідомлення про успішне завантаження

Після завантаження системи, його можна відкрити із головного меню встановлених програм, клацнувши на відповідну іконку з зображенням додатку.

Для запуску мобільного додатку, треба перейти в меню на телефоні, де буде розташована піктограма (рис. 2.5).

Рис. 2.5. Піктограма додатку на екрані телефона

2.6.4. Опис інтерфейсу користувача

Меню додатку (рис. 2.6) надає доступ до основних компонентів, що забезпечує функціонал програми: Налаштування, Результати диктантів, Пошук по правопису та Головне меню.

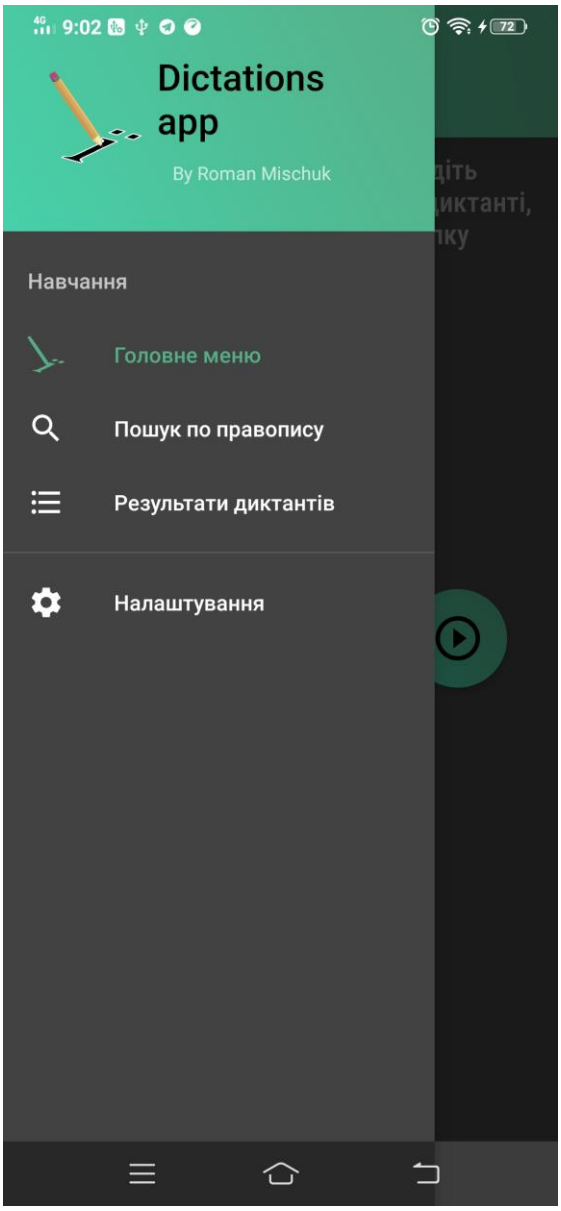

Рис. 2.6. Меню додатку

Пункт меню «Головне меню» (рис. 2.7) забезпечує основний функціонал додатку, а саме:

– диктування тексту із файлу за допомогою синтезатора голосу;
– ввід тексту користувачем;
– повна перевірка знання орфографії.

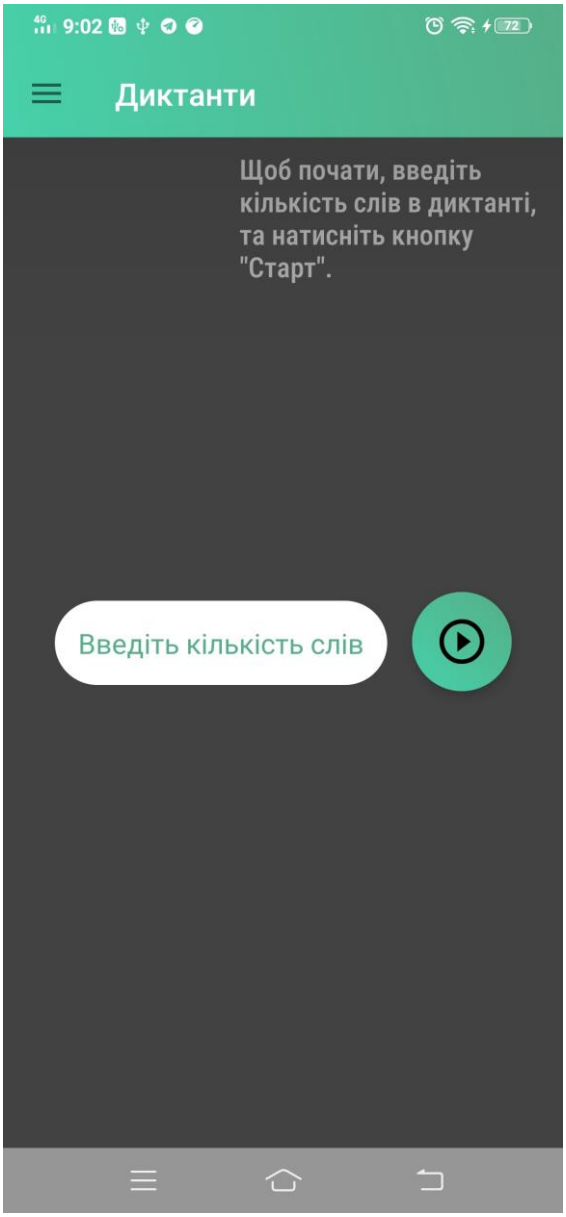

Рис. 2.7. Головне меню додатку «Диктанти»

Для того, щоб почати роботу з диктантом, необхідно внести кількість слів для перевірки (рис. 2.8).

Рис. 2.8. Ввід кількості слів в тексті для перевірки

Джерелом слів для диктантів було обрано Український правопис, який має предметний покажчик (рис. 2.9). У ньому зібрані слова, що містять основні орфограми із посиланнями на конкретні правила, які потрібно знати щоб правильно написати ці слова.

УКРАЇНСЬКИЙ ПРАВОПИС

СХВАЛЕНО

Кабінетом Міністрів України

(Постанова № 437 від 22 травня 2019 р.)

спільним рішенням

Президії Національної академії наук України

(протокол № 22/10 від 24 жовтня 2018 р.)

і Колегії Міністерства освіти і науки України

(протокол № 10/4-13 від 24 жовтня 2018 р.)

ЗАТВЕРДЖЕНО

Українською національною комісією з питань правопису

(протокол № 5 від 22 жовтня 2018 р.)

2019

Рис. 2.10. Зображення файлу Українського правопису

Текст видобутий із даного PDF файлу, містить у собі різні артефакти, що роблять неможливим його використання без його попередньої обробки.

Розроблена програма виявляє та виправляє такі артефакти :

1. Рядок містить помилки що не дозволяють зчитати з нього слово;
2. Слово містить невідомі символи що не дозволяють його використовувати для перевірки користувача;
3. Слово є скороченням або абревіатурою, що не можна диктувати користувачу, адже тоді буде не зрозуміло, що саме потрібно написати.

Кожне слово, разом із його правилами, що пройшло всі три етапи перевірки, додається до списку результатів. Таким чином, отримуємо список слів, які можна використовувати для укладання диктантів. Інформацію про правила, яких стосується слово, буде використовуватися для спеціалізації диктанту відповідно до проблемного правила користувача (рис. 2.11).

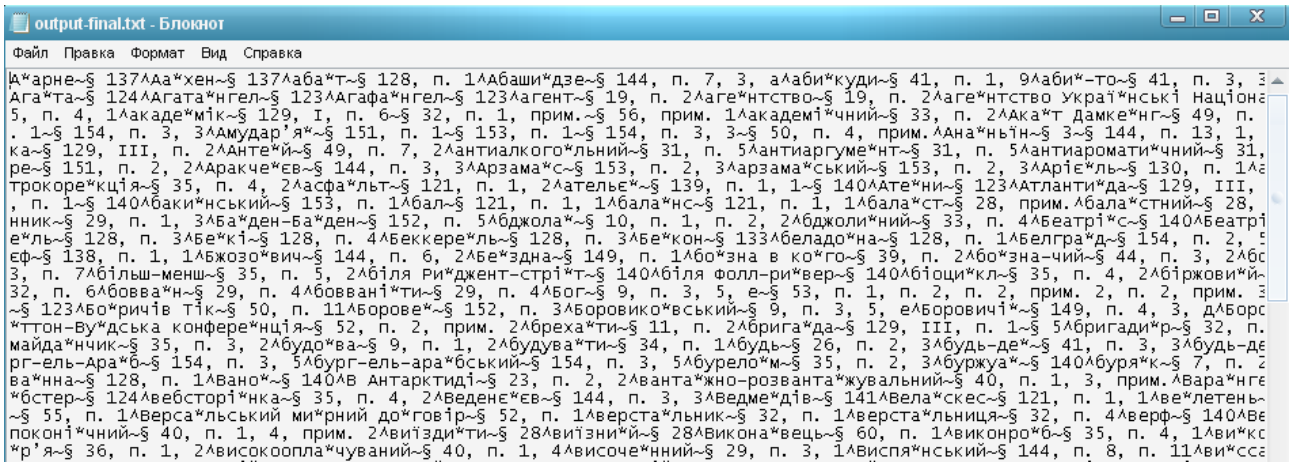

Рис. 2.11. Текст – еталон, що містить слова і правила українською мовою

Для того, щоб спонукати користувача вивчати невідомі йому теми, диктант складається зі слів, які відповідають цим невідомим темам. Наприклад, якщо користувач погано знає, коли потрібно ставити знак м’якшення, то йому треба диктувати саме слова із цією літерою. Таку процедуру потрібно продовжувати, допоки користувач не покаже, що тему він знає.

Якщо ж користувач вперше користується даною системою, то інформація про нього відсутня. Тому при першому диктанті вибираються слова для диктування випадково, допоки не буде зроблена помилка. Так системою буде виявлено проблемну тему.

Для диктування слів користувачеві застосований застосунок TextToSpeech API - застосунок-читач. Цей застосунок використовує технологію «синтезу мовлення» (також відому як текст-у-мовлення) аби читати вголос текст з екрану.

На рис. 2.12. наведено зображення екрану під час диктування слова. У відповідне поле користувач вводить слово, яке він щойно прослухав. Також він може прослухати його ще раз, прослухати повільніше, або запитати пояснення невідомого слова (рис. 2.13).

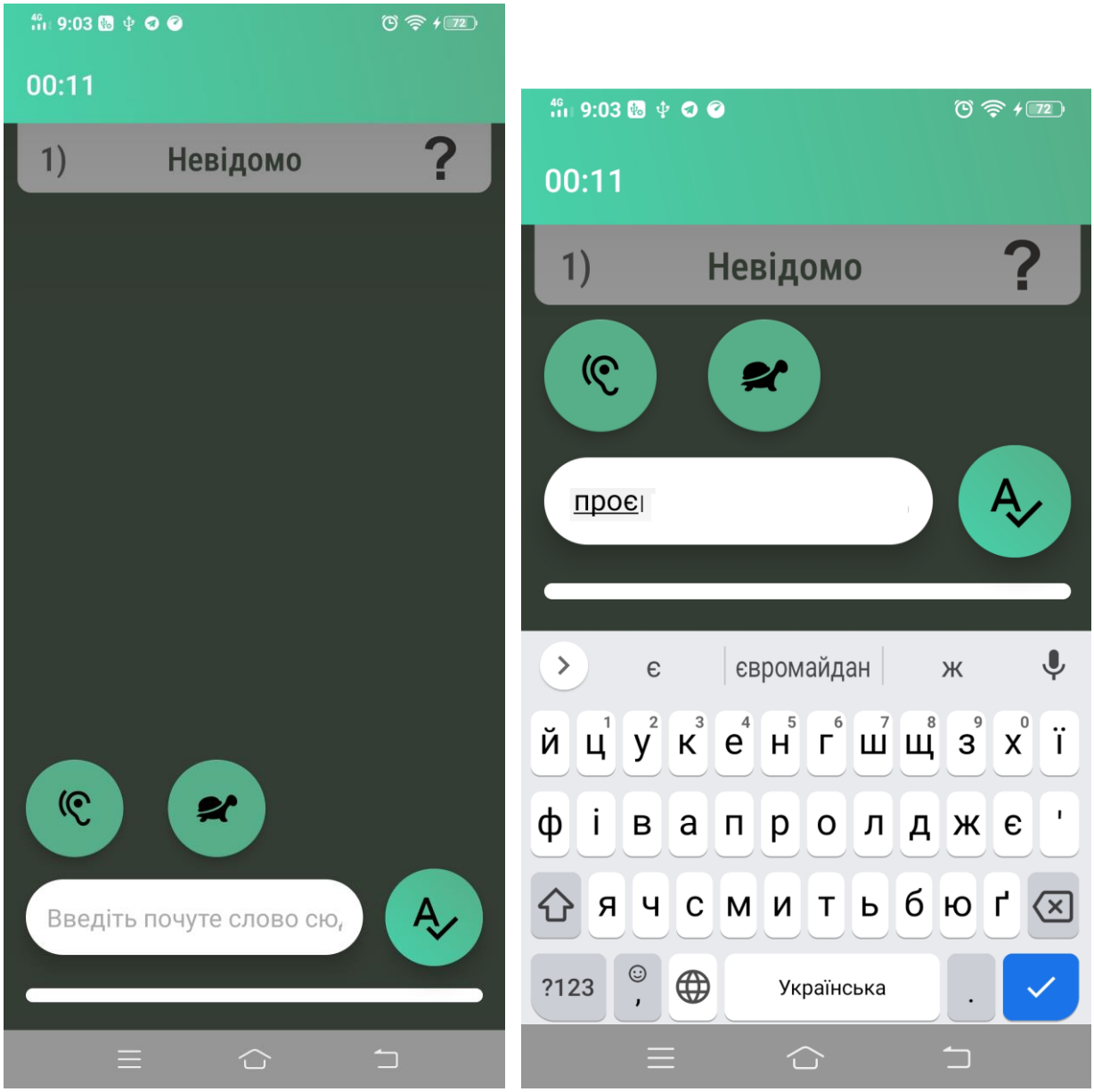

Рис. 2.12. Ввід слова диктанту

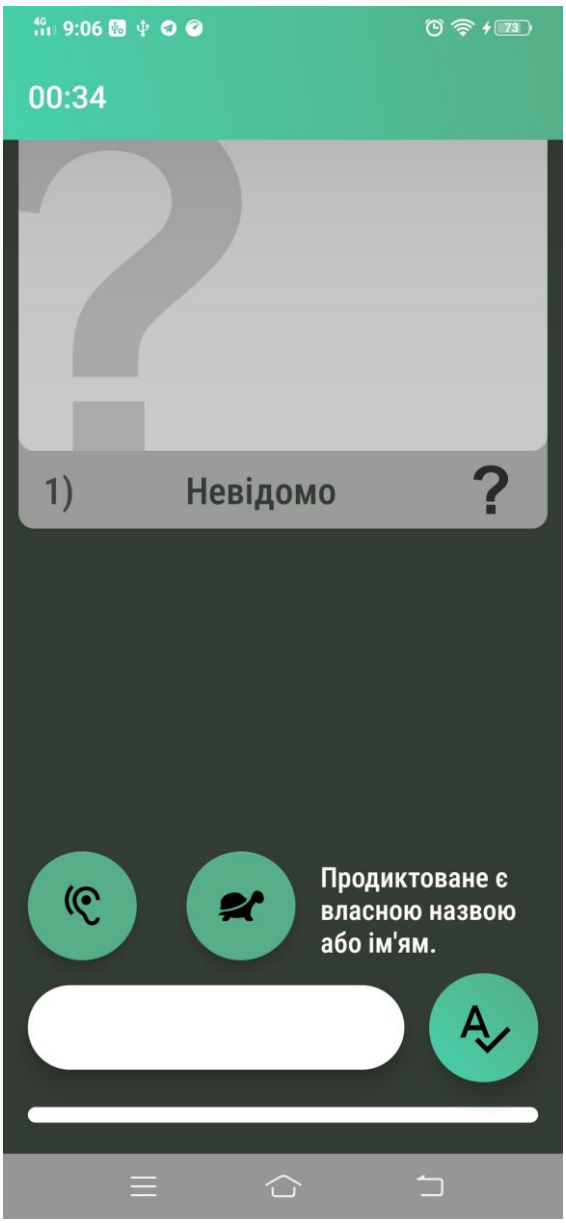

Рис. 2.13. Пояснення значення слова з помічника програми

В програмі передбачені правила, за якими програма буде «виправляти» випадково введене користувачем слово:

1. Видалення пробілів на початку та в кінці слова, до першого символу.
2. Заміна подвоєних пробілів на одинарні.
3. Видалення всіх символів, що не є літерами української мови, числами, дефісом або лапками.
4. Заміна «неправильно» написаних лапок.

Для збереження інформації про користувача використовується вбудована в Android структура даних – геш-таблиця, що дозволяє записувати та знаходити значення пари ключ-значення.

У даному випадку ключем є номер правила в Українському правописі, а значенням – коефіцієнт незнання, що прямо пропорційний кількості слів, написаних користувачем неправильно.

Якщо робиться помилка, то значення коефіцієнту відповідного правила збільшується на константу. Якщо ж користувач записує продиктоване слово правильно, то значення коефіцієнту ділиться на константу.

Таким чином засвоєння нових правил з точки зору застосунку має такий вигляд :

1. Виявлення проблемної теми.
2. Примус користувача її вивчати.
3. Вивчення користувачем теми.
4. Перевірка, наскільки гарно користувач вивчив тему.

В результаті отримано диктанти, персоналізовані під конкретного користувача. Також, інформацію з цієї ж геш-таблиці можна використовувати для створення рекомендацій користувачеві: обрати три теми із найбільшим коефіцієнтом незнання, та вивести їх на екран. Саме ці теми йому потрібно краще вивчити (рис. 2.14).

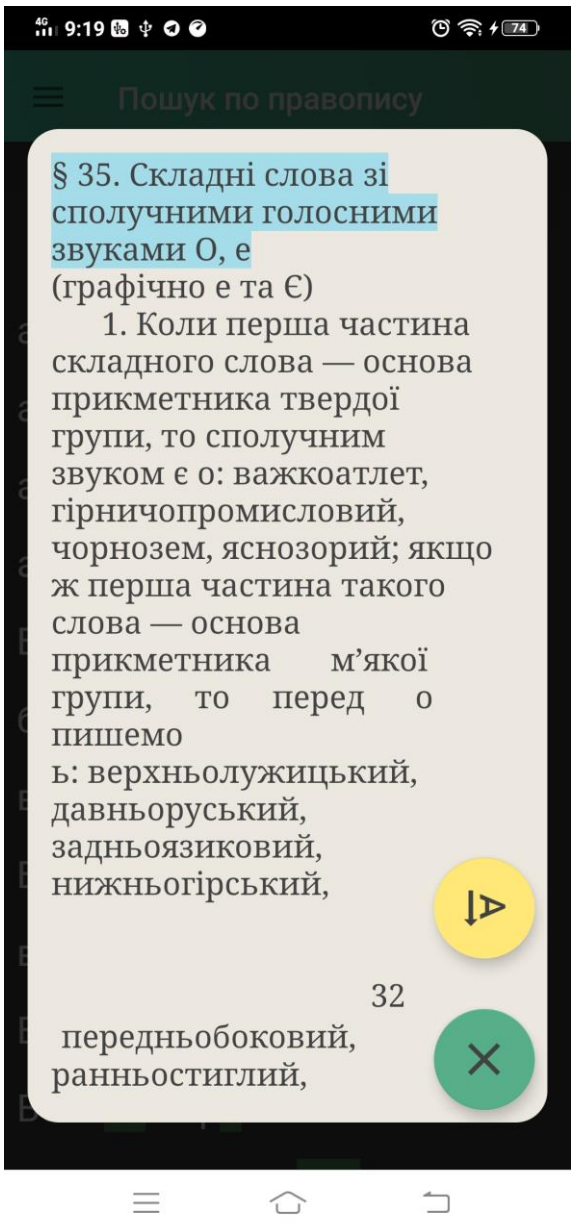

Рис. 2.14. Зображення «проблемної теми» користувача

Пункт меню «Налаштування» призначений для корегування та налаштування синтезатора мови для своїх вподобань (рис. 2.15, 2.16) та дозволяє змінити швидкість диктовки та висоту голосу для більш сприятливого звучання.

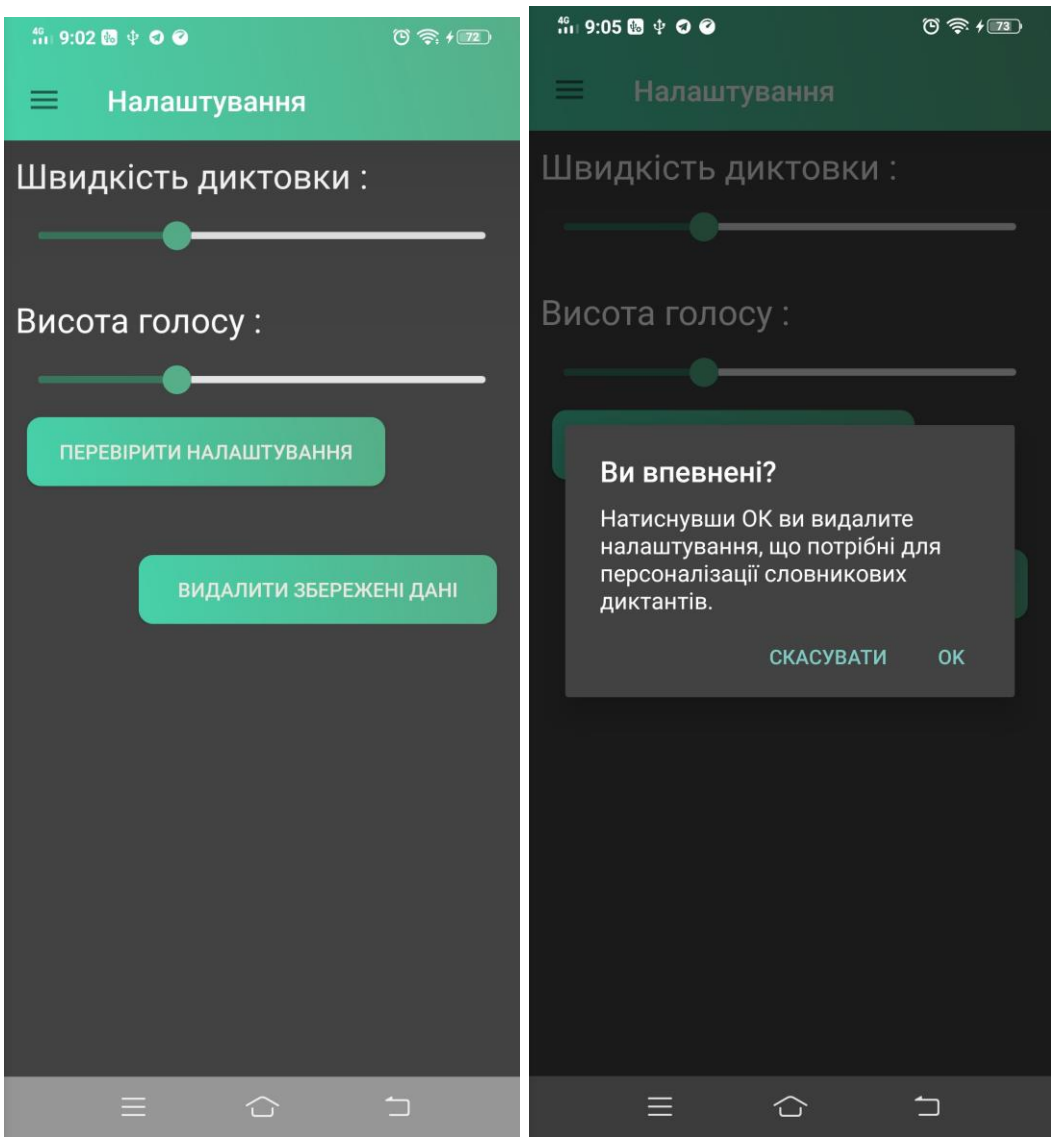

Рис. 2.15. Меню «Налаштування»

Меню «Пошук по правопису» дозволяє знайти в тексті Українського правопису необхідні йому слова та правила їх правопису (рис. 2.17-2.19).

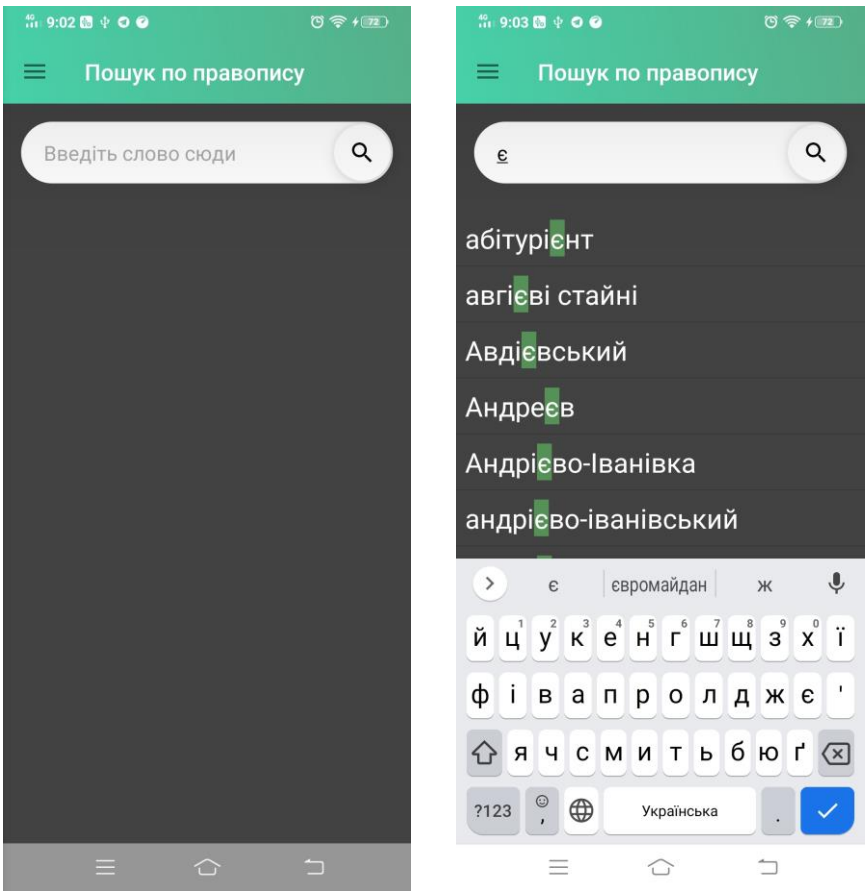

Рис. 2.16. Пошук слів за критеріями

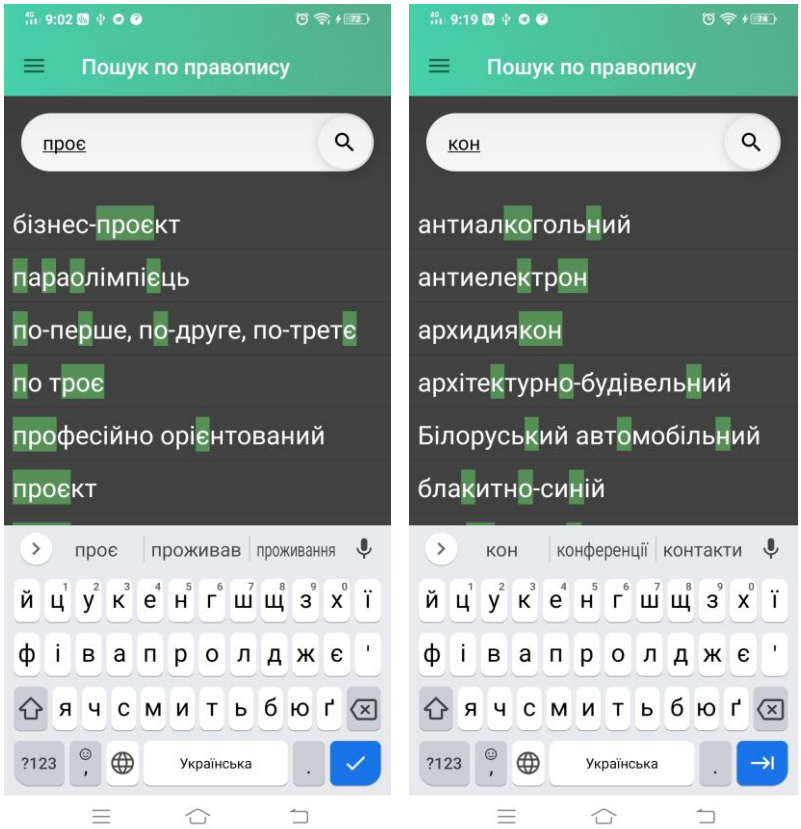

Рис. 2.17. Пошук слів за наявністю букв

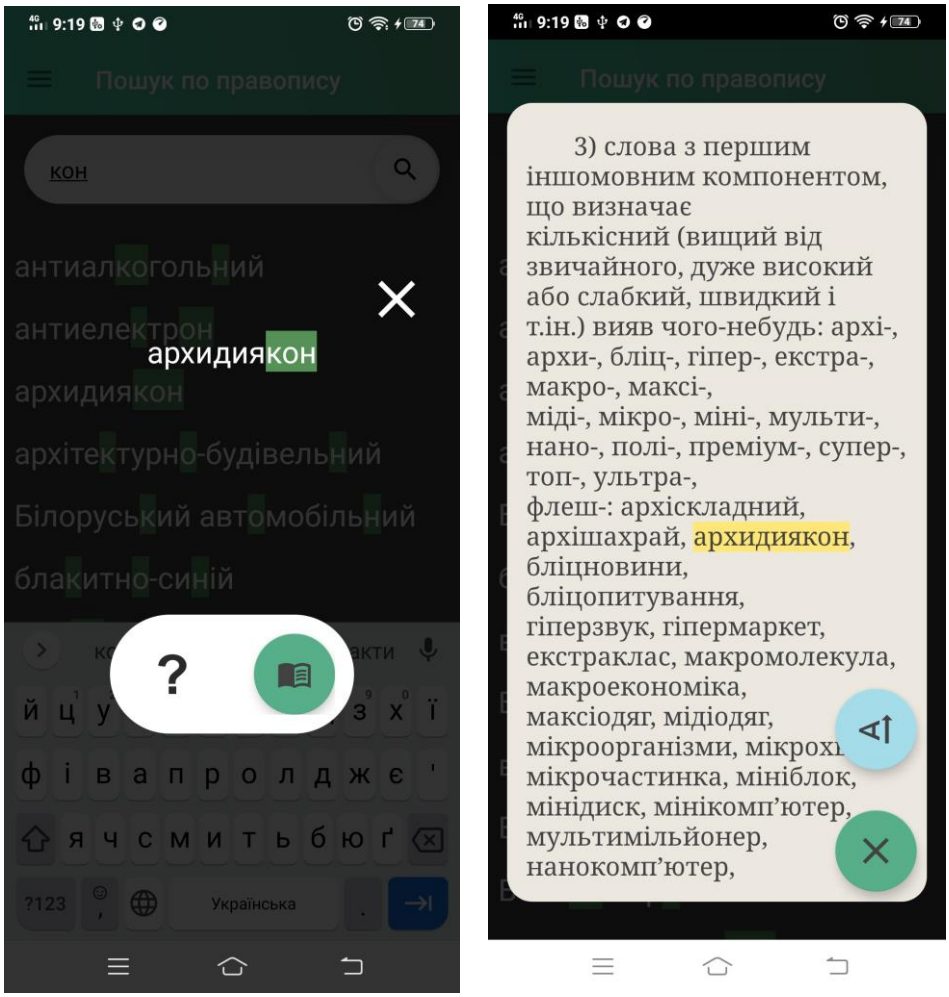

Рис. 2.18. Перегляд слова та правила правопису, до якого воно відноситься

Для завершення роботи з додатком необхідно скористатися стандартними клавішами виходу з додатку, при цьому на екрані з'являться попередження про вихід (рис. 2.19).

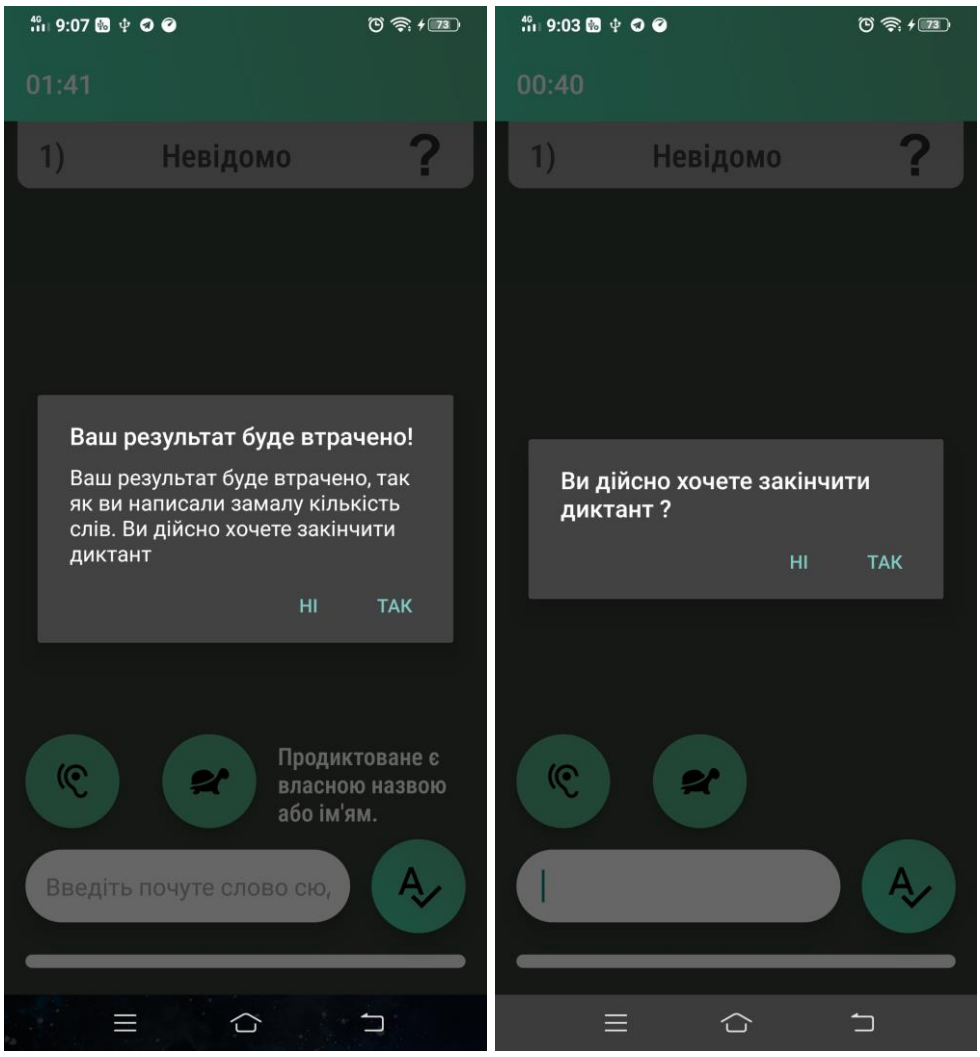

Рис. 2.19. Попередження про вихід з додатку

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані розробки програмного забезпечення:

- передбачуване число операторів – 860;
- коефіцієнт складності програми – 1,25;
- коефіцієнт кореляції програми в ході її розробки - 0,1;
- середня годинна заробітна плата програміста, грн/год – 100;
- вартість машино-години ЕОМ, грн/год – 7.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_и + t_a + t_п + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

$t_п$ – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \tag{3.2}$$

де q – передбачуване число операторів,

С– коефіцієнт складності програми,

р – коефіцієнт кореляції програми в ході її розробки.

$$Q = 860 \cdot 1{,}25 \cdot (1 + 0{,}1) = 1182;$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин}, \tag{3.3}$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1.2 ... 1.5,

К – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. до 2 – 0,8.

$$t_u = \frac{1182 \cdot 1{,}2}{85 \cdot 0{,}8} = 22, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)K}; \tag{3.4}$$

$$t_a = \frac{1182}{20 \cdot 0{,}8} = 73, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K}; \quad (3.5)$$

$$t_n = \frac{1182}{25 \cdot 0{,}8} = 59, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K}; \quad (3.6)$$

$$t_{отл} = \frac{1182}{4 \cdot 0{,}8} = 369, \text{ людино-годин,}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \quad (3.7)$$

$$t_{отл}^{к} = 1{,}2 \cdot 369 = 443, \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}; \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15...20)K}; \quad (3.9)$$

$$t_{др} = \frac{1182}{15 \cdot 0{,}8} = 98, \text{ людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}; \quad (3.10)$$

$$t_{до} = 0{,}75 \cdot 98 = 73$$, людино-годин.

$$t_{д} = 98 + 73 = 172$$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 +22 + 73 +59 + 443 +172 = 821, людино-годин.

В результаті ми розрахували, що в загальній складності необхідно 821 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин,

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$С_{ЗП} = 821 \cdot 100 = 82100, \text{грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 443 \cdot 7 = 3104, \text{ грн.}$$

$$\hat{E}_{ïî} = 82100 + 3104 = 85204, \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.}, \qquad (3.14)$$

де B_k - число виконавців;

$F_р$ - місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{821}{1 \cdot 176} = 4{,}6 \text{ міс.}$$

Таким чином, очікувана тривалість розробки складе 4,6 місяця, а витрати на створення програмного забезпечення 85204 грн.

ВИСНОВКИ

Аналіз оновленого Українського правопису показує, що значна кількість норм зазнала суттєвих змін, через що з’являється необхідність адаптації носіїв мови до нових правил.

Порівняння наявного програмного забезпечення дає підстави стверджувати, що прямих аналогів для української мови не існує. Хоча окремі застосунки дозволяють вивчати правила правопису за допомогою тестових завдань, створення тестів вимагає значних зусиль, і ці застосунки не встигають охопити всі зміни правопису.

Виходячи з аналізу змін правопису та враховуючи недоліки програм-аналогів, було сформульовано функціональні вимоги до власного програмного забезпечення, серед яких можливість роботи оффлайн, робота в середовищі Android, забезпечення індивідуальної освітньої траєкторії користувача.

Для створення програмного забезпечення було обрано середовище розробки Android Studio як офіційний інструмент компанії Google для цієї платформи.

Спроєктовано формат зберігання завдань, інформації про рівень засвоєння користувачем окремих норм правопису, інтерфейс користувача; створено алгоритми визначення положення помилок в словах та генерації словникового диктанту згідно індивідуальної освітньої траєкторії користувача.

Основним результатом роботи є мобільний застосунок для написання та перевірки словникових диктантів, що сприятиме засвоєнню правил оновленого Українського правопису.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 4,6 місяця, трудомісткість розробки ПЗ – 821 людино-годин, а витрати на її створення програмного забезпечення – 85204 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Геш-таблиця / Матеріал з Вікіпедії - вільної енциклопедії URL : https://uk.wikipedia.org/wiki/Геш-таблиця. дата звернення: 22.01.2021.

2. Динамічне програмування / Матеріал з Вікіпедії - вільної енциклопедії URL: https://uk.wikipedia.org/wiki_Динамічне_програмування. дата звернення: 22.01.2021.

3. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

4. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

5. ЗНО 2020 тести: Українська мова // Категорія «Освіта» / Крамниця застосунків Google Play URL: https://play.google.com/store/apps/ details?id=com.truedevels.zno_ukrainian_language. дата звернення: 22.01.2021.

6. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

7. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

8. Питання українського правопису : Постанова Кабінету Міністрів України від 22.05.2019 №437 // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/437-2019-%D0%BF#Text дата звернення: 22.01.2021

9. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К.: Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

10. Стислий огляд основних змін у новій редакції «Українського правопису» (2019) // Розділ «Новини» / Інститут мовознавства ім. О.О. Потебні Національної академії наук України URL: http://www.inmo.org.ua/assets/files/2019/ukr.-pravopys-2019.-osnovni-zminy.pdf дата звернення: 22.01.2021.

11. Пошук найдовшої спільної підпослідовності / Матеріал з Вікіпедії - вільної енциклопедії URL: https://uk.wikipedia.org/wiki/ Найдовша_спільна_підпослідовність. дата звернення: 22.01.2021.

12. Український правопис 2019 року // Розділ «Освіта» /Міністерство освіти та науки України. URL : https://mon.gov.ua/storage/app/media/ zagalna%20serednya/05062019-onovl-pravo.pdf . дата звернення: 22.01.2021.

13. Український правопис 2019 року : Покажчик, сторінка 206 // Розділ «Освіта» /Міністерство освіти та науки України. URL: https://mon.gov.ua/storage/app/media/zagalna%20serednya/05062019-onovl-pravo.pdf . дата звернення: 22.01.2021.

14. Android Studio/ Матеріал з Вікіпедії - вільної енциклопедії URL: https://uk.wikipedia.org/wiki/Android_Studio. дата звернення: 22.01.2021.

15. Duolingo: вивчайте англійську // Категорія «Освіта» / Крамниця застосунків Google Play URL: https://play.google.com/store /apps/details?id=com.duolingo. дата звернення: 22.01.2021.

16. Material.io / Офіційний сайт розробників дизайну, на якому зібрана інформація та рекомендації по його використанню URL: https://material.io/ . дата звернення: 22.01.2021.

17. Material design / Матеріал з Вікіпедії - вільної енциклопедії URL: https://uk.wikipedia.org/wiki/Material_design. дата звернення: 22.01.2021.

18. Mobile operating systems' market share worldwide from January 2012 to October 2020 // Розділ «Telecommunications» / Статистична компанія Statista

URL : https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ дата звернення: 22.01.2021.

19. Python/ Матеріал з Вікіпедії - вільної енциклопедії URL: https://uk.wikipedia.org/wiki/Python . дата звернення: 22.01.2021.

20. TextToSpeech// Android Docs/Android Developers URL: https://developer.android.com/reference/android/speech/tts/TextToSpeech. дата звернення: 22.01.2021.

ДОДАТОК А

КОД ПРОГРАМИ

main.py

```
import os, pickle, sys, re
from parsing import pdfparse, wordsparse, wordscheck
import utils

# Private functions
VariablesSaveLocation = "Savings/vars"
def SaveVariables() -> None:
    """Saves global variables like pdf file location"""
    pickle.dump([
        PdfFileLocation,
        ResultFileLocation,
        wordsparse.WordsWithNagolos,
        wordsparse.SkippedWords,
        wordsparse.SplintedWords,
        ParsedPages,
        WordsDictionary,
        wordscheck.CheckedWords],
        open(VariablesSaveLocation, "wb"))
def LoadVariables() -> None:
    """Loads global variables from save file"""
    global PdfFileLocation, ResultFileLocation, IsLocationChanged,\
        ParsedPages, WordsDictionary
    if os.path.isfile(VariablesSaveLocation) and\
            os.stat(VariablesSaveLocation).st_size > 0:
        PdfFileLocation, \
        ResultFileLocation, \
        wordsparse.WordsWithNagolos, \
        wordsparse.SkippedWords, \
        wordsparse.SplintedWords, \
        ParsedPages, \
        WordsDictionary, \
        wordscheck.CheckedWords \
            = pickle.load(open(VariablesSaveLocation, "rb"))
        IsLocationChanged = True

# Global functions
def Help(args: list, argstr: str = None) -> None:
    """Function, that prints help about some function"""
    if args.__len__() == 1:
        for func_name in AvailableCommands:
            Help(["help", func_name])
    elif args.__len__() == 2:
        if args[1] in AvailableCommands.keys():
            print(f"[{args[1]}] - {AvailableCommands[args[1]][1]}\n")
        else: print(f"\'{args[1]}\' - there is no such command.")
```

```
        def Exit(args: list = None, argstr: str = None):
            print("Do you want to save session ?")
            inp = ""
            while inp not in ["т", "н"]:
                inp = input("(т|н) : ")
            if inp == "т":
                SaveVariables()
            if PdfFile:
                PdfFile.close()
            print("Exiting...")
            sys.exit(0)

        def Location(args: list, argstr: str):
            """Function that allows you to change such variables as PdfFileLocation or
ResultFileLocation"""
            global PdfFileLocation, ResultFileLocation, IsLocationChanged
            if args.__len__() == 1:
                if PdfFileLocation is None and ResultFileLocation is None:
                    print("There is no locations specified.")
                else:
                    if PdfFileLocation is not None:
                        print(f"Parsing PDF file location : \"{PdfFileLocation}\";")
                    if ResultFileLocation is not None:
                        print(f"Output text file location : \"{ResultFileLocation}\";")
            elif args.__len__() > 1:
                if args[1] == "show":
                    for loc in args[1:]:
                        if loc == "pdf":
                            print(f"Parsing PDF file location : \"{PdfFileLocation}\";")
                        elif loc == "output":
                            print(f"Output text file location : \"{ResultFileLocation}\";")
                elif args[1] == "set" and args.__len__() >= 4:
                    location = utils.GetPath(args[3], argstr)
                    if args[2] == "pdf":
                        if os.path.exists(location):
                            PdfFileLocation = location
                            IsLocationChanged = True
                            print(f"New PDF file location : \"{PdfFileLocation}\";")
                        else:
                            print(f"Invalid file location : \"{location}\"!")
                    elif args[2] == "output":
                        ResultFileLocation = location
                        print(f"New output text file location : \"{ResultFileLocation}\";")
                elif args[1] == "save":
                    SaveVariables()
                    print("Variables saved.")
                elif args[1] == "load":
                    LoadVariables()
                    print("Variables loaded.")
                elif args[1] == "clear" and args.__len__() >= 3:
                    if args[2] == "words":
```

```
            global WordsDictionary, ParsedPages
            WordsDictionary = dict()
            ParsedPages = list()
            print("Parsed words were cleared.")
        elif args[2] == "splinted":
            wordsparse.SplintedWords = dict()
            wordsparse.SkippedWords = list()
            print("Splinted words were cleared.")
        elif args[2] == "checked":
            wordscheck.CheckedWords = dict()
            print("Checked words were cleared.")
        elif args[2] == "all":
            WordsDictionary = dict()
            ParsedPages = list()
            wordsparse.SplintedWords = dict()
            wordsparse.SkippedWords = list()
            wordscheck.CheckedWords = dict()
            wordsparse.WordsWithNagolos = dict()
            WordsDictionary = dict()
        else:
            print("Invalid variable name.")
    else: print("Invalid command")

def Parse(args: list, argstr: str):
    """Function, that starts parsing process"""
    global IsLocationChanged, PdfFileLocation, PdfFile, WordsDictionary
    if args.__len__() > 1:
        if args[1] == "show":
            print(f"Parsed pages : {ParsedPages}.\n"
                  f"Total number of parsed words = {WordsDictionary.keys().__len__()}")
            #print(WordsDictionary)
        elif args[1] == "save":
            print("Saving words...")
            file = open(ResultFileLocation, "w")
            for (word, rules) in WordsDictionary.items():
                file.write(word + utils.__RulesSplitter__)
                file.write(utils.__RulesSplitter__.join(rules) +
                           utils.__WordBlockSplitter__)
            file.close()
            print("Words saved!")
        elif args[1] == "load":
            location = utils.GetPath(args[2], argstr)
            print(f"Loading words from \"{location}\"...")
            file = open(location, "r")
            WordsDictionary = dict()
            for word_info in file.read().split(utils.__WordBlockSplitter__):
                spl = word_info.split(utils.__RulesSplitter__)
                WordsDictionary[spl[0]] = spl[1:]
            print("Words loaded!")
        elif PdfFileLocation is None:
            print("Error : invalid pdf file location!")
            '''elif ResultFileLocation is None:
```

```
                print("Error : invalid output file location!")'"
            else:
                if IsLocationChanged:
                    PdfFile = pdfparse.PdfFile(PdfFileLocation)
                    IsLocationChanged = False
                # noinspection PyUnboundLocalVariable
                len = PdfFile.__len__()
                if args.__len__() == 2:
                    start = int(args[1])
                    if start < len:
                        if start not in ParsedPages:
                            wordsparse.ParseWordsFromStr(
                                PdfFile.GetWordStr(start),
                                WordsDictionary)
                            ParsedPages.append(start)
                    else:
                        print(f"Invalid page number ({start}, when max value is {len - 1}).")
                        return
                elif args.__len__() == 3:
                    start = int(args[1])
                    stop = int(args[2])
                    if start <= stop < len:
                        for i in range(start, stop + 1):
                            if i not in ParsedPages:
                                SaveVariables()
                                wordsparse.ParseWordsFromStr(
                                    PdfFile.GetWordStr(i),
                                    WordsDictionary)
                                ParsedPages.append(i)
                    else:
                        print(f"Invalid page range [{start}; {stop}].")
                        return
                print("Parsing finished.")
            else:
                print("Incorrect number of arguments!")
                return

        def Check(args: list, argst: str):
            if args.__len__() == 1:
                print("Checking parsed words...")
                wordscheck.WordsCheck(WordsDictionary, SaveVariables)
                print("Checking ended.")
            elif args.__len__() == 2:
                if args[1] == "show":
                    print("Computing...")
                    __unchecked_num__ = 0
                    for word in WordsDictionary.keys():
                        if word not in wordscheck.CheckedWords:
                            __unchecked_num__ += 1
                    print(f"Number of checked words = {min(wordscheck.CheckedWords.__len__(),
WordsDictionary.__len__())}.\n"
                          f"Number of unchecked words = {__unchecked_num__}.")
```

```
            else:
                print("Invalid number of arguments!")


        # Global variables
        PdfFileLocation = None
        ResultFileLocation = None
        IsLocationChanged = True

        PdfFile = None
        '"Structure : {word: [rule1, rule2, ...], ...}"'
        WordsDictionary = dict()
        ParsedPages = list()

        AvailableCommands = {
            "help": (Help, "Prints help (this message).\n"
                           "    Usages :\n"
                           "   >\"help\"\n"
                           "    Prints help for all available functions.\n"
                           "   >\"help [func_name]\",\n"
                           "    where func_name - name of required function.\n"
                           "    Prints help for required function."),
            "exit": (Exit, "Closes the application.\n"
                           "    Usage : \"exit\""),
            "location": (Location, "Works with files locations.\n"
                                   "    Usages :\n"
                                   "   >\"location\"\n"
                                   "    Prints all specified locations.\n"
                                   "   >\"location show [file_names...]\","
                                   "    where file_names - names of files, location of which you want to check
(pdf, output).\n"
                                   "    Prints locations of specified files.\n"
                                   "   >\"location set (pdf|output) [file_location]\",\n"
                                   "    where file_location - new file location.\n"
                                   "    Sets location of pdf or output files.\n"
                                   "   >\"location save\"\n"
                                   "    Dumps all variables to restore in next session.\n"
                                   "   >\"location load\"\n"
                                   "    Loads variables from the dump.\n"
                                   "   >\"location clear (words|checked|splinted|all)\"\n"
                                   "    Deletes value of specified variable (words)."),
            "parse": (Parse, "Parses words from the specified PDF file.\n"
                             "    Usages :\n"
                             "   >\"parse [page_num]\",\n"
                             "    where page_num - number of the parsing page.\n"
                             "    Parses words from the specified page of the PDF.\n"
                             "   >\"parse [start_page_num] [end_page_num]\",\n"
                             "    where start_page_num and end_page_num specify range of parsing
pages.\n"
                             "    Parses words from the specified pages range.\n"
                             "   >\"parse show\"\n"
                             "    Shows parsed words.\n"
```

```
                "   >\"parse save\"\n"
                "    Saves parsed words to the selected location.\n"
                "   >\"parse load [file_location]\"\n"
                "    Loads saved words from specified file_location."
            ),
    "check": (Check, "Checks words parsed from the PDF file.\n"
                "  Usages :\n"
                "  >\"check\"\n"
                "  Starts words checking process.\n"
                "  You can stop it and save the result at any moment.\n"
                "  >\"check show\"\n"
                "  Prints number of checked words."
            )
}

# Application start :
LoadVariables()
print("PRAVOPYS PARSER APPLICATION:")
print("Enter your command or run \'help\' to show list of available commands.")
try:
    while True:
        Input = input(">>>")
        Inputs = Input.split()
        if Inputs[0] in AvailableCommands.keys():
            AvailableCommands[Inputs[0]][0](Inputs, Input)
        else: print(f"\'{Inputs[0]}\' - no such command.")
except KeyboardInterrupt:
    Exit()
```

utils.py

```
import colorit, re

__ErrorMess__ = "<! Error>"
def GetError() -> str:
    if SupportsColors:
        return colorit.background(__ErrorMess__, colorit.Colors.red)
    else:
        return "\n" + __ErrorMess__

__WarningMess__ = "<! Warning>"
def GetWarning() -> str:
    if SupportsColors:
        return colorit.background(__WarningMess__, colorit.Colors.yellow)
    else:
        return "\n" + __WarningMess__

def GetPath(arg, arg_str):
    return (
        arg if arg[0] != '\"'
        else re.search(r"\"(.*)\"", arg_str).group(1)
    ).replace("\\", "/")
```

```
__RulesSplitter__ = "~"
__WordBlockSplitter__ = "^"

res = ""
while res not in ["т", "н"]:
    res = input("Enable terminal coloring? (т|н) : ")
SupportsColors = res == "т"
```

Main.java

```
package com.company.r;
import javafx.util.Pair;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {
    public static int CONTAINS_CAPITAL_LETTER = 1;
    public static int CONTAINS_NUMER = 2;
    public static String WORDS_NAME = "words";
    public static String RULES_NAME = "rules";
    public static String WORDS_SPLITTER = "^";
    public static String WORDSRULES_SPLITTER = "~";
    public static String PARAGRAF_SYMBOL = "\u00A7";
    public static String notWordsRegex = "[^A-Яа-яґєїіҐЄЇІ0-9\'"\\-\\s]+";
    public static String wordsRegex = "A-Яа-яґєїіҐЄЇІ";
    public static String smallWordsRegex = "[а-яґєїі]";
    public static String capitalWordsRegex = "[А-ЯҐЄЇІ]";

    public static String replaceAll(String pattern, String replaceTo,  String s){
        while(s.contains(pattern)) s = s.replace(pattern, replaceTo);
        return s;
    }
    public static String formatWord(String word){
        int ind = word.indexOf(WORDSRULES_SPLITTER);
        if(ind == -1) ind = word.length();
        return replaceAll("*", "", word.substring(0, ind));
    }

    static String[] words;
    static int root = -1;
    static int delta = 4;
    static HashMap<Integer, HashMap<Integer, Integer>> tree = new HashMap<>();
    static int min(int a, int b, int c){
        return Math.min(a, Math.min(b, c));
    }
    static int dist(int test_word, String user_input){
```

```
    int w = words[test_word].length();
    int u = user_input.length();
    int[][] dp = new int[w+1][u+1];

    for(int i = 0; i <= w; i++)
        dp[i][0] = i;
    for(int i = 0; i <= u; i++)
        dp[0][i] = i;

    for(int W = 1; W <= w; W++){
        for(int U = 1; U <= u; U++){
            if(words[test_word].charAt(W-1) !=
                    user_input.charAt(U-1)){
                dp[W][U] = min(
                        dp[W-1][U-1] + 1,
                        dp[W-1][U] + 1,
                        dp[W][U-1] + 1
                );
                if(U == u && W > u)
                    dp[W][U] = Math.min(
                            dp[W][u],
                            dp[u][u]
                    );
            }
            else dp[W][U] = dp[W-1][U-1];
        }
    }
    return dp[w][u];
}
static void goodAdd(int word_num){
    if(root == -1){
        root = word_num;
    }
    else {
        int min = -1;
        int mini = -1;
        int mind = -1;
        for(int i = 0; i < words.length; i++){
            int dist = dist(i, words[word_num]);
            if(dist < min || min == -1){
                if(!tree.get(i).containsKey(dist)){
                    min = dist;
                    mini = i;
                    mind = dist;
                }
            }
        }
        tree.get(mini).put(mind, word_num);
    }
}
static void add(int word_num, int start_from){
    if(root == -1){
```

```
            tree.put(word_num, new HashMap<>());
            root = word_num;
        }
        else{
            int dist = dist(start_from, words[word_num]);
            if(tree.get(start_from).containsKey(dist)){
                add(word_num, tree.get(start_from).get(dist));
            }
            else{
                tree.put(word_num, new HashMap<>());
                tree.get(start_from).put(dist, word_num);
            }
        }
    }
    static ArrayList<Pair<Integer, String>> similar(String input, int start_from){
        ArrayList<Pair<Integer, String>> ret = new ArrayList<>();
        int dist = dist(start_from, input);
        if(dist <= delta) ret.add(new Pair<>(dist, words[start_from]));
        int end = dist + delta;
        for(int i = Math.max(1, dist - delta); i <= end; i++)
            if(tree.get(start_from).containsKey(i))
                ret.addAll(similar(input,
                        tree.get(start_from).get(i)));
        ret.sort(Comparator.comparing(x -> x.getKey()));
        return ret;
    }
    static ArrayList<Pair<Integer, String>> badSimilar(String input){
        ArrayList<Pair<Integer, String>> ret = new ArrayList<>();
        for(int i = 0; i < words.length; i++){
            int dist = dist(i, input);
            if(dist <= delta)
                ret.add(new Pair<>(dist, words[i]));
        }
        ret.sort(Comparator.comparing(x -> x.getKey()));
        return ret;
    }

    public static void main(String[] args){
        try {
            words = Files.readAllLines(Path.of("files/words")).get(0).split("\\^");
            for(int i = 0; i < words.length; i++){
                words[i] = formatWord(words[i]);
                tree.put(i, new HashMap<>());
            }
            for(int i = 0; i < words.length; i++){
                //goodAdd(i);
                add(i, root);
            }
            System.out.println("Now write :");
            Scanner scanner = new Scanner(System.in);
            while(true){
                String inp = scanner.next();
```

```
            long st = System.currentTimeMillis();
            System.out.println(badSimilar(inp));
            System.out.println(System.currentTimeMillis() - st);
            st = System.currentTimeMillis();
            System.out.println(similar(inp, root));
            System.out.println(System.currentTimeMillis() - st);
         }
      }
      catch (Exception e){
         e.printStackTrace();
      }
   }
}
```

Tue.java

```
package com.company.r;

import javafx.util.Pair;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.lang.reflect.Array;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {
   public static int CONTAINS_CAPITAL_LETTER = 1;
   public static int CONTAINS_NUMER = 2;
   public static String WORDS_NAME = "words";
   public static String RULES_NAME = "rules";
   public static String WORDS_SPLITTER = "^";
   public static String WORDSRULES_SPLITTER = "~";
   public static String PARAGRAF_SYMBOL = "\u00A7";
   public static String notWordsRegex = "[^A-Яa-яґєїіҐЄЇІ0-9\"'\\-\\s]+";
   public static String wordsRegex = "A-Яa-яґєїіҐЄЇІ";
   public static String smallWordsRegex = "[a-яґєїі]";
   public static String capitalWordsRegex = "[A-ЯҐЄЇІ]";

   public static String replaceAll(String pattern, String replaceTo,  String s){
      while(s.contains(pattern)) s = s.replace(pattern, replaceTo);
      return s;
   }
   public static String formatWord(String word){
      int ind = word.indexOf(WORDSRULES_SPLITTER);
      if(ind == -1) ind = word.length();
      return replaceAll("*", "", word.substring(0, ind));
```

```
}
static int min(int a, int b, int c){
   return Math.min(a, Math.min(b, c));
}
static int dist(String test_word, String user_input){
   int w = test_word.length();
   int u = user_input.length();
   int[][] dp = new int[w+1][u+1];

   for(int i = 0; i <= w; i++)
      dp[i][0] = i;
   for(int i = 0; i <= u; i++)
      dp[0][i] = i;

   for(int W = 1; W <= w; W++){
      for(int U = 1; U <= u; U++){
         if(test_word.charAt(W-1) !=
               user_input.charAt(U-1)){
            dp[W][U] = min(
                  dp[W-1][U-1] + 1,
                  dp[W-1][U] + 1,
                  dp[W][U-1] + 1
            );
            if(W > U && U == u)
               dp[W][U] = Math.min(
                     dp[W][U],
                     dp[U][U]
               );
         }
         else dp[W][U] = dp[W-1][U-1];
      }
   }
   return dp[w][u];
}

static int delta = 4;
static HashMap<String, List<Pair<Integer, String>>> tree = new HashMap<>();
static String[] words;
static void add(String word){
   tree.put(word, Arrays.asList(new Pair(0, word)));
   for(int d = 1; d <= delta; d++){
      String prev = word.substring(d);
      for(int i = 0; i < word.length() - d; i++){
         String buff = word.substring(0, i) + prev;
         if(tree.containsKey(buff))
            tree.get(buff).add(new Pair<>(
                  dist(word, buff),
                  word
            ));
         else
            tree.put(buff, Arrays.asList(
               new Pair<>(
```

```
                        dist(word, buff),
                        word
                    )
                ));
            prev = prev.substring(1);
        }
    }
}

public static void main(String[] args){
    try {
        words = Files.readAllLines(Path.of("files/words")).get(0).split("\\^");
        for(int i = 0; i < words.length; i++){
            words[i] = formatWord(words[i]);
            //add(0, i);
        }
        Scanner scanner = new Scanner(System.in);
        while(true){
            //System.out.println(similar(scanner.next(), root));
        }
    }
    catch (Exception e){
        e.printStackTrace();
    }
    /*words = new String[]{"авторитет", "hello"};
    System.out.println(dist(0, words[1]));
    System.out.println(dist(1, words[0]));*/
}
}
```

MainMenuActivity.java

```
package ua.roman.dictations;

import androidx.appcompat.app.AlertDialog;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.ObjectAnimator;
import android.animation.TimeInterpolator;
import android.content.ComponentName;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.speech.tts.Voice;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
```

```
import androidx.appcompat.widget.*;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.interpolator.view.animation.FastOutSlowInInterpolator;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;

import java.util.Locale;
import java.util.Set;

import ua.roman.dictations.algo.Funcs;
import ua.roman.dictations.dictants.Dictant;
import ua.roman.dictations.dictants.DictantActivity;
import ua.roman.dictations.dictants.TimedDictantActivity;
import ua.roman.dictations.helper.CommonActivity;

import static ua.roman.dictations.algo.Funcs.showError;

public class MainMenuActivity extends CommonActivity {
    //Змінна, що міститить в собі кнопку початку диктанту
    private static final int START_MENU_SIZE = 2;
    private boolean isStartMenuOpened = false;
    private boolean isStartMenuClicked = false;
    private FloatingActionButton startButton;
    private DictantStartMenuItem[] dictantStartMenu;
    private View dictantStartMenuBack;
    private ObjectAnimator dictantStartMenuBackAnimator;

    //Змінна, що містить в собі поле для введення кількості слів в диктанті
    private TextInputEditText wordsCount;
    private TextInputLayout wordsCountLayout;

    //Код помилки, коли користувач неправильно ввів кількість слів в диктанті
    private static final int INCRORRECT_WORDS_COUNT_FORMAT = 1;
    //Код помилки, коли користувач перевищив максимальну кількість слів в диктанті
    private static final int MAX_WORDS_COUNT_THROWED = 2;
    private static final int MIN_WORDS_COUNT_THROWED = 3;

    //Змінна, що містить в собі синтезатор голосу
    private TextToSpeech tts;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //Виконуємо дії системи при страрті
        super.onCreate(savedInstanceState);
        //Вимикаємо стандартну анімацію переходів для активностей
        overridePendingTransition(R.anim.fade_in, R.anim.dont_move);
        //Встановлюємо на екрані потрібну активність
        setContentView(R.layout.main_menu_layout);
```

```
        //Задаємо значення змінній
        wordsCountLayout = findViewById(R.id.main_menu_words_count_layout);
        wordsCount = findViewById(R.id.main_menu_words_count);
        wordsCount.setOnKeyListener(new View.OnKeyListener() {
            @Override
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                wordsCountLayout.setError(null);
                return false;
            }
        });

        //Задаємо значення змінної
        startButton = findViewById(R.id.main_menu_start_button);

        Toolbar toolbar = findViewById(R.id.main_menu_toolbar);
        DrawerLayout drawerLayout = findViewById(R.id.main_menu_drawer_layout);
        super.initToolbarDrawer("Диктанти", toolbar, drawerLayout, this);

        findViewById(R.id.main_menu_background).setOnTouchListener(new
View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                Funcs.hideKeyBoard(MainMenuActivity.this);
                wordsCount.clearFocus();
                return false;
            }
        });

        //Перевіряємо, як працює синтезатор голосу
        checkTTS();
        initButton();
    }

    @Override
    public void onBackPressed() {
        if(isStartMenuOpened) {
            changeStartDictantMenuOpened(false);
            return;
        }
        new AlertDialog.Builder(this, R.style.AlertDialog)
                .setTitle("Ви дійсно хочете вийти ?")
                .setPositiveButton("Так", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        tts.stop();
                        finish();
                    }
                }).setCancelable(false)
                .setNegativeButton("Ні", null).create().show();
    }

    @Override
```

```
        protected void onStop() {
            tts.stop();
            super.onStop();
            finish();
        }

        //Коли користувач повертається в додаток
        @Override
        protected void onResume() {
            //Якщо він намагався виправити помилку
            wordsCount.setText("");
            Funcs.hideKeyBoard(this);
            super.onResume();
        }

        //Функція, що перевіряє, як користувач ввів кількість слів
        private int checkNumber(){
            String text = wordsCount.getText().toString();
            if(text.equals("") || Integer.parseInt(text) <= 0) return
INCRORRECT_WORDS_COUNT_FORMAT;
            else if(Integer.parseInt(text) > Dictant.MAX_WORDS_COUNT) return
MAX_WORDS_COUNT_THROWED;
            else if(Integer.parseInt(text) < Dictant.MIN_WORDS_COUNT) return
MIN_WORDS_COUNT_THROWED;
            return 0;
        }

        //Функція, що запускає перевірку синтезатору голосу
        private void checkTTS(){
            tts = new TextToSpeech(this, new TextToSpeech.OnInitListener() {
                @Override
                public void onInit(int status) {
                    if(status == TextToSpeech.SUCCESS && tts.getEngines().size() > 0 &&
                            Funcs.isPackageInstalled("com.google.android.tts",
MainMenuActivity.this.getPackageManager())){
                        if(!isLanguageAvailable(Locale.forLanguageTag("uk-UA")))
                            showError(TTS_UKRAINIAN_IS_NOT_INSTALLED,
MainMenuActivity.this);
                    } else showError(TTS_IS_NOT_INSTALLED, MainMenuActivity.this);
                }
            });
        }
        //Функція, що перевіряє наявність паку для деякої мови
        private boolean isLanguageAvailable(Locale language) {
            if(language == null) return false;
            boolean available = false;
            switch (tts.isLanguageAvailable(language))
            {
                case TextToSpeech.LANG_AVAILABLE:
                case TextToSpeech.LANG_COUNTRY_AVAILABLE:
                case TextToSpeech.LANG_COUNTRY_VAR_AVAILABLE:
                    tts.setLanguage(language);
```

```
                Set<Voice> voices = tts.getVoices();
                for (Voice voice : voices)
                   if(voice != null && voice.getLocale() == language){
                      Set<String> features = voice.getFeatures();
                      if (features != null &&
!features.contains(TextToSpeech.Engine.KEY_FEATURE_NOT_INSTALLED))
                         available = true;
                   }
                tts.setLanguage(language);
                break;

             case TextToSpeech.LANG_MISSING_DATA:
             case TextToSpeech.LANG_NOT_SUPPORTED:
             default:
                break;
          }
          return available;
       }
       private void initButton(){
          //StartButtonMenu Init
          dictantStartMenuBack = findViewById(R.id.main_menu_start_dictant_menu_back);
          dictantStartMenuBack.setOnTouchListener(new View.OnTouchListener() {
             @Override
             public boolean onTouch(View v, MotionEvent event) {
                changeStartDictantMenuOpened(false);
                return false;
             }
          });
          dictantStartMenuBackAnimator = new ObjectAnimator();
          dictantStartMenuBackAnimator.setTarget(dictantStartMenuBack);
          dictantStartMenuBackAnimator.setDuration(300);
          dictantStartMenuBackAnimator.setPropertyName("alpha");
          dictantStartMenuBackAnimator.addListener(new AnimatorListenerAdapter() {
             @Override
             public void onAnimationStart(Animator animation) {
                if(isStartMenuOpened) dictantStartMenuBack.setVisibility(View.VISIBLE);
             }
             @Override
             public void onAnimationEnd(Animator animation) {
                if(!isStartMenuOpened) dictantStartMenuBack.setVisibility(View.GONE);
             }
          });

          FastOutSlowInInterpolator interpolator = new FastOutSlowInInterpolator();
          dictantStartMenu = new DictantStartMenuItem[]{
                new DictantStartMenuItem(
                      (FloatingActionButton)
findViewById(R.id.main_menu_start_normal_dictant),
                      (TextView)
findViewById(R.id.main_menu_start_normal_dictant_description),
                      CommonActivity.SIMPLE_DICTANT_INDEX, interpolator
                ),
```

```
                new DictantStartMenuItem(
                    (FloatingActionButton)
findViewById(R.id.main_menu_start_timed_dictant),
                    (TextView)
findViewById(R.id.main_menu_start_timed_dictant_description),
                    CommonActivity.TIMED_DICTANT_INDEX, interpolator
                )
            };

            startButton.setOnClickListener(new View.OnClickListener() {
                int fff;
                @Override
                public void onClick(View v) {
                    switch (checkNumber()) {
                        case 0:
                            Funcs.hideKeyBoard(MainMenuActivity.this);
                            changeStartDictantMenuOpened(!isStartMenuOpened);
                            break;
                        case MAX_WORDS_COUNT_THROWED:
                            wordsCountLayout.setError("Введіть кількість, меншу за " +
(Dictant.MAX_WORDS_COUNT + 1) + ".");
                            break;
                        case MIN_WORDS_COUNT_THROWED:
                            wordsCountLayout.setError("Введіть кількість, більшу за " +
(Dictant.MIN_WORDS_COUNT - 1) + ".");
                            break;
                        default:
                            wordsCountLayout.setError("Будь ласка, введіть кількість слів в
диктанті.");
                    }
                }
            });
        }

        private void changeStartDictantMenuOpened(boolean newIsStartMenuOpened){
            if(isStartMenuOpened == newIsStartMenuOpened) return;
            isStartMenuOpened = newIsStartMenuOpened;

            if(isStartMenuOpened) dictantStartMenuBackAnimator.setFloatValues(0, 1);
            else dictantStartMenuBackAnimator.setFloatValues(1, 0);
            dictantStartMenuBackAnimator.start();

            for(int i = 0; i < START_MENU_SIZE; i++)
                dictantStartMenu[i].changeStartDictantMenuOpened(isStartMenuOpened);
        }

        private class DictantStartMenuItem implements View.OnClickListener {
            public FloatingActionButton ItemButton;
            public TextView ItemDescription;
            public final int ItemID;
            private boolean IsMenuOpened = false;
```

```
private final float itemButtonMargin;
private final ObjectAnimator buttonAnimator;
private final ObjectAnimator descriptionAnimator;

public DictantStartMenuItem(FloatingActionButton itemButton,
                    TextView itemDescription, int itemID,
                    TimeInterpolator animationInterpolator) {
   ItemButton = itemButton;
   ItemDescription = itemDescription;
   ItemID = itemID;
   itemButtonMargin = ItemButton.getTranslationY();

   buttonAnimator = new ObjectAnimator();
   buttonAnimator.setPropertyName("translationY");
   buttonAnimator.setDuration(200);
   buttonAnimator.setTarget(ItemButton);
   buttonAnimator.setInterpolator(animationInterpolator);
   buttonAnimator.addListener(new AnimatorListenerAdapter() {
      @Override
      public void onAnimationEnd(Animator animation) {
         if (IsMenuOpened) {
            ItemDescription.setVisibility(View.VISIBLE);
            descriptionAnimator.setFloatValues(0, 1);
            descriptionAnimator.start();
         }
      }
   });

   descriptionAnimator = new ObjectAnimator();
   descriptionAnimator.setPropertyName("alpha");
   descriptionAnimator.setDuration(200);
   descriptionAnimator.setTarget(ItemDescription);
   descriptionAnimator.addListener(new AnimatorListenerAdapter() {
      @Override
      public void onAnimationEnd(Animator animation) {
         if(!isStartMenuOpened){
            buttonAnimator.setFloatValues(0, itemButtonMargin);
            buttonAnimator.start();
            ItemDescription.setVisibility(View.GONE);
         }
      }
   });

   ItemButton.setOnClickListener(this);
   ItemDescription.setOnClickListener(this);
}

public void changeStartDictantMenuOpened(boolean newIsOpened){
   if(newIsOpened == IsMenuOpened) return;
   if(newIsOpened){
      buttonAnimator.setFloatValues(itemButtonMargin, 0);
      buttonAnimator.start();
```

```
                }
                else{
                    descriptionAnimator.setFloatValues(1, 0);
                    descriptionAnimator.start();
                }
                IsMenuOpened = newIsOpened;
            }

            @Override
            public void onClick(View v) {
                if(isStartMenuClicked) return;
                isStartMenuClicked = true;

                Class<?> intentTarget;
                switch (ItemID){
                    case CommonActivity.SIMPLE_DICTANT_INDEX:
                        intentTarget = DictantActivity.class;
                        break;
                    case CommonActivity.TIMED_DICTANT_INDEX:
                        intentTarget = TimedDictantActivity.class;
                        break;
                    default:
                        return;
                }

                //Creating and starting intent
                //Створюємо намір переходу в нову активність
                Intent intent = new Intent(MainMenuActivity.this, intentTarget);
                //Змінна, в яку будуть записані координати кнопки початку
                int[] buffLoc = new int[2];
                //Запискємо значення змінній
                ItemButton.getLocationOnScreen(buffLoc);
                //Через намір передаємо значення координат кнопки, для створення анімації
переходу
                intent.putExtra("button_X", buffLoc[0] + startButton.getCustomSize()/2);
                intent.putExtra("button_Y", buffLoc[1] + startButton.getCustomSize()/2);
                //Та кількість слів, що вибрав користувач
                intent.putExtra("words_num", Integer.parseInt(wordsCount.getText().toString()));
                //Виконуємо намір
                startActivity(intent);
            }
        }
    }
```

SearchInPravopysActivity.java

```
package ua.roman.dictations;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.widget.Toolbar;
import androidx.drawerlayout.widget.DrawerLayout;
```

```
import android.content.Context;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.SpannableString;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.util.Pair;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Executor;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;
import java.util.regex.Pattern;

import ua.roman.dictations.algo.TextWork;
import ua.roman.dictations.helper.CommonActivity;

public class SearchInPravopysActivity extends CommonActivity {
    private static int MIN_DELTA = 1;
    private static int MAX_WORDS_TO_SHOW = 100;

    private EditText input;
    private ListView res;
    private FloatingActionButton searchButton;

    private String inputText;
    private ArrayList<Pair<Integer, String>> results = new ArrayList<>();
    private adapter adapter;

    private ExecutorService async = Executors.newFixedThreadPool(1);
```

```
private Handler handler = new Handler();
private Future asyncRunning = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.search_in_pravopys_with_drawer);

   Toolbar toolbar = findViewById(R.id.main_menu_toolbar);
   DrawerLayout layout = findViewById(R.id.searach_in_pravo_drawer_layout);
   initToolbarDrawer("Пошук по правопису", toolbar, layout, this);

   input = findViewById(R.id.search_in_prav_word);
   res = findViewById(R.id.search_in_prav_results);
   searchButton = findViewById(R.id.serch_in_prav_search);
   searchButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         input.setText("");
      }
   });

   adapter = new adapter(this);
   res.setAdapter(adapter);

   setRules();
   input.addTextChangedListener(new TextWatcher() {
      @Override
      public void beforeTextChanged(CharSequence s, int start, int count, int after) {

      }
      @Override
      public void onTextChanged(CharSequence s, int start, int before, int count) {
         if(asyncRunning != null) asyncRunning.cancel(true);
         asyncRunning = async.submit(new Runnable() {
            @Override
            public void run() {
               updateResults(input.getText().toString());
            }
         });
      }
      @Override
      public void afterTextChanged(Editable s) {
      }
   });
}

private void updateResults(String input){
   inputText = input;
   final boolean hasText = input != null && input.length() > 0;
   if(hasText) {
      ArrayList<Pair<Integer, String>> buff = new ArrayList<>();
```

```
                for (int i = 0; i < WORDS.length; i++) {
                    int delta = TextWork.longestCommonStringLength(
                            TextWork.formatWord(WORDS[i]), input);
                    if(delta >= MIN_DELTA)
                        buff.add(new Pair<>(delta, WORDS[i]));
                }
                Collections.sort(buff, new Comparator<Pair<Integer, String>>() {
                    @Override
                    public int compare(Pair<Integer, String> o1, Pair<Integer, String> o2) {
                        return o2.first.compareTo(o1.first);
                    }
                });
                results = buff;
            }
            else results.clear();
            handler.post(new Runnable() {
                @Override
                public void run() {
                    adapter.notifyDataSetChanged();
                    res.smoothScrollToPosition(0);
                    if(hasText) searchButton.setImageResource(R.drawable.ic_close);
                    else searchButton.setImageResource(R.drawable.ic_search);
                }
            });
        }

        private void setRules(){
            WORDS = TextWork.openFile(this, TextWork.WORDS_NAME)
                    .split(Pattern.quote(TextWork.WORDS_SPLITTER));
            setRULES();
        }

        class adapter extends ArrayAdapter<String>{
            LayoutInflater inflater;
            Context c;

            adapter(@NonNull Context context) {
                super(context, 0);
                inflater = LayoutInflater.from(context);
                c = context;
            }

            @NonNull
            @Override
            public View getView(int position, @Nullable View convertView, @NonNull
ViewGroup parent) {
                if(convertView == null)
                    convertView = inflater.inflate(R.layout.search_in_pravo_el,
                            parent, false);
                TextView text = convertView.findViewById(R.id.serch_in_prav_el_text);
                text.setText(TextUtils.concat(
                        new SpannableString((position+1) + ") "),
```

```
                TextWork.highlightErrors(
                        TextWork.formatWord(results.get(position).second),
                        inputText,
                        c, R.color.transparent, R.color.rearch_hightlight,
                        0
                ));
        ((View)text.getParent())
                .setTag(R.string.result_activity_tag, results.get(position).second);
        return convertView;
    }

    @Override
    public int getCount() {
        return Math.min(results.size(), MAX_WORDS_TO_SHOW);
    }
  }
}
```

SettingsActivity.java

```
package ua.roman.dictations;

import androidx.annotation.IdRes;
import androidx.appcompat.widget.Toolbar;
import androidx.drawerlayout.widget.DrawerLayout;

import android.app.AlertDialog;
import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.RadioGroup;
import android.widget.SeekBar;
import android.widget.Toast;

import java.util.Locale;

import ua.roman.dictations.algo.DictantResult;
import ua.roman.dictations.algo.Funcs;
import ua.roman.dictations.helper.CommonActivity;

import static ua.roman.dictations.algo.Funcs.isPackageInstalled;
import static ua.roman.dictations.algo.Funcs.showError;

public class SettingsActivity extends CommonActivity {
    private static float MIN_SPEED = 0.1f;
    private static float MAX_SPEED = 3f;
    private static float MIN_SPEACH = 0.1f;
    private static float MAX_SPEACH = 3f;
```

```
    private SeekBar speed;
    private SeekBar speach;
    private SharedPreferences sharedPreferences;
    private TextToSpeech tts;

    private RadioGroup timedDictantDifficulty;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.settings_layout_with_drawer);

        Toolbar toolbar = findViewById(R.id.main_menu_toolbar);
        DrawerLayout layout = findViewById(R.id.settings_drawer_layout);
        initToolbarDrawer("Налаштування", toolbar, layout, this);

        sharedPreferences = getSharedPreferences(CommonActivity.SHARED_PREF_NAME,
MODE_PRIVATE);
        speed = findViewById(R.id.settings_speed_bar);
        speach = findViewById(R.id.settings_pitch_bar);

        tts = new TextToSpeech(this, new TextToSpeech.OnInitListener() {
            int fff;
            @Override
            public void onInit(int status) {
                tts.setLanguage(Locale.forLanguageTag("uk"));
                SettingsActivity.this.findViewById(R.id.setting_check_button)
                        .setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        float newSpeed = MIN_SPEED + (MAX_SPEED - MIN_SPEED) *
speed.getProgress()/100f;
                        float newPitch = MIN_SPEACH + (MAX_SPEACH - MIN_SPEACH) *
speach.getProgress()/100f;
                        tts.setSpeechRate(newSpeed);
                        tts.setPitch(newPitch);
                        tts.speak("Тест диктування", TextToSpeech.QUEUE_FLUSH, null, null);
                    }
                });
            }
        });

        timedDictantDifficulty = findViewById(R.id.timed_dictant_difficulty_setting);

        showSettings();
        setupOnClicks();
    }

    private void setupOnClicks(){
        findViewById(R.id.delete_words_map_button).setOnClickListener(new
View.OnClickListener() {
```

```
            int fff;
            @Override
            public void onClick(View v) {
                new AlertDialog.Builder(SettingsActivity.this, R.style.AlertDialog)
                        .setTitle("Ви впевнені?").setMessage("Натиснувши ОК ви видалите
налаштування, що потрібні для персоналізації словникових диктантів.")
                        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                Funcs.writeWordsMap(SettingsActivity.this, null,
Funcs.WORDS_MAP_SAVE_NAME);
                                Funcs.writeWordsMap(SettingsActivity.this, null,
DictantResult.SHARED_PREF_FOR_DICTANT_RESULTS_COUNT);
                                getSharedPreferences(CommonActivity.SHARED_PREF_NAME,
Context.MODE_PRIVATE)
                                        .edit().clear().apply();
                                showSettings();
                                Toast.makeText(SettingsActivity.this, "Налаштування видалено.",
Toast.LENGTH_LONG).show();
                            }
                        }).setNegativeButton("Скасувати", null).create().show();
                }
            });

            SeekBar.OnSeekBarChangeListener listener = new
SeekBar.OnSeekBarChangeListener() {
                @Override
                public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
{}
                @Override
                public void onStartTrackingTouch(SeekBar seekBar) {}
                @Override
                public void onStopTrackingTouch(SeekBar seekBar) {
                    updateSettings();
                }
            };
            speed.setOnSeekBarChangeListener(listener);
            speach.setOnSeekBarChangeListener(listener);
            findViewById(R.id.speach_settings_def).setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {

sharedPreferences.edit().putFloat(CommonActivity.SHARED_PREF_TTS_PITCH, 1.0f)
                        .putFloat(CommonActivity.SHARED_PREF_TTS_SPEACH_RATE,
1.0f).apply();
                    showSettings();
                }
            });

            timedDictantDifficulty.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
```

```
                @Override
                public void onCheckedChanged(RadioGroup group, int checkedId) {
                    updateSettings();
                }
            });

            findViewById(R.id.install_offline_data).setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if(Funcs.isPackageInstalled("com.google.android.tts",
SettingsActivity.this.getPackageManager())){
                        Intent intent = new Intent();
                        intent.setComponent(new ComponentName("com.google.android.tts",
                                "com.google.android.tts.local.voicepack.ui.VoiceDataInstallActivity"));
                        Toast.makeText(SettingsActivity.this, "Виберіть українську мову.",
Toast.LENGTH_LONG).show();
                        SettingsActivity.this.startActivity(intent);
                    } else showError(TTS_IS_NOT_INSTALLED, SettingsActivity.this);
                }
            });

            /*findViewById(R.id.test_words_map).setOnClickListener(new
View.OnClickListener() {
                int fff;
                @Override
                public void onClick(View v) {
                    HashMap<Integer, Integer> buff = new HashMap<>();
                    buff.put(1, 1);
                    buff.put(2, 2);
                    Funcs.writeWordsMap(SettingsActivity.this, buff,
Funcs.WORDS_MAP_SAVE_NAME);
                    Toast.makeText(SettingsActivity.this, "Words map created",
Toast.LENGTH_SHORT).show();
                }
            });*/
        }

        private void updateSettings(){
            float newSpeed = MIN_SPEED + (MAX_SPEED - MIN_SPEED) *
speed.getProgress()/100f;
            float newPitch = MIN_SPEACH + (MAX_SPEACH - MIN_SPEACH) *
speach.getProgress()/100f;

            sharedPreferences.edit().putFloat(CommonActivity.SHARED_PREF_TTS_PITCH,
newPitch)
                    .putFloat(CommonActivity.SHARED_PREF_TTS_SPEACH_RATE, newSpeed)
                    .putInt(CommonActivity.SHARED_PREF_TIMED_DICTANT_DIFFICULTY,

getTimedDictantDifficulty(timedDictantDifficulty.getCheckedRadioButtonId()))
                    .apply();
        }
```

```
    private void showSettings(){
        float Speed =
sharedPreferences.getFloat(CommonActivity.SHARED_PREF_TTS_SPEACH_RATE, 1f);
        float Pitch =
sharedPreferences.getFloat(CommonActivity.SHARED_PREF_TTS_PITCH, 1f);

        speach.setProgress((int)((Pitch - MIN_SPEACH) * 100 / (MAX_SPEACH -
MIN_SPEACH)));
        speed.setProgress((int)((Speed - MIN_SPEED) * 100 / (MAX_SPEED -
MIN_SPEED)));

        timedDictantDifficulty.check(getRadioFromTimedDictantDifficulty(

sharedPreferences.getInt(CommonActivity.SHARED_PREF_TIMED_DICTANT_DIFFICULTY,
0)
        ));
    }

    private int getTimedDictantDifficulty(@IdRes int viewId){
        if (viewId == R.id.timed_dictant_middle)
            return 1;
        else if(viewId == R.id.timed_dictant_difficult)
            return 2;
        else
            return 0;
    }
    private @IdRes int getRadioFromTimedDictantDifficulty(int difficulty){
        switch (difficulty){
            case 0:
                return R.id.timed_dictant_eazy;
            case 1:
                return R.id.timed_dictant_middle;
            case 2:
                return R.id.timed_dictant_difficult;
            default:
                return -1;
        }
    }
}
```

ResultActivity.java

```
package ua.roman.dictations;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityOptionsCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.LayoutTransition;
```

```
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.content.Intent;
import android.os.Bundle;
import android.text.Spannable;
import android.text.SpannableString;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
import android.text.method.LinkMovementMethod;
import android.text.style.ClickableSpan;
import android.text.style.ForegroundColorSpan;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Map;
import java.util.TreeMap;

import ua.roman.dictations.algo.DictantResult;
import ua.roman.dictations.algo.Funcs;
import ua.roman.dictations.algo.TextWork;
import ua.roman.dictations.helper.CommonActivity;
import ua.roman.dictations.helper.PravopysDialog;

public class ResultActivity extends CommonActivity {

    private Toolbar toolbar;
    private DrawerLayout drawerLayout;
    private View rootLayout;
    private ProgressBar progressBar;
    private LinearLayout resultTable;

    private int selectedColumn = 4;
    private int startedFrom = 0;

    private DictantResult dictantResult;
    private int cx = 0;
    private int cy = 0;
    private float progress;
    private View[] columnsForResize;

    private static int PROGRESS_ANIMATION_DURATION = 800;
    private static int NEW_PARAGRAFS_TO_SHOW = 3;
    public static int STARTED_FROM_DICTANT = 0;
    public static int STARTED_FROM_RESULTS_ACTIVITY = 1;
```

```
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            overridePendingTransition(R.anim.dont_move, R.anim.dont_move);
            setContentView(R.layout.result_layout_with_drawer);

            rootLayout = findViewById(R.id.dictant_result_root_layout);
            readExtras();
            if (savedInstanceState == null) {
                rootLayout.setVisibility(View.INVISIBLE);

                ViewTreeObserver viewTreeObserver = rootLayout.getViewTreeObserver();
                if (viewTreeObserver.isAlive()) {
                    viewTreeObserver.addOnGlobalLayoutListener(new
ViewTreeObserver.OnGlobalLayoutListener() {
                        int fff;
                        @Override
                        public void onGlobalLayout() {
                            Funcs.circularRevealActivity(rootLayout, cx, cy, new
AnimatorListenerAdapter() {
                                @Override
                                public void onAnimationEnd(Animator animation) { animateProgress();}
                            });
                            rootLayout.getViewTreeObserver().removeOnGlobalLayoutListener(this);
                        }
                    });
                }
            }

            progressBar = findViewById(R.id.dictant_result_bar);
            resultTable = findViewById(R.id.dictant_result_table);
            columnsForResize = new View[]{
                    findViewById(R.id.dictant_result_table_column_0),
                    findViewById(R.id.dictant_result_table_column_1),
                    findViewById(R.id.dictant_result_table_column_2)
            };

            toolbar = findViewById(R.id.main_menu_toolbar);
            drawerLayout = findViewById(R.id.result_layout_drawer_layout);
            super.initToolbarDrawer("Твій результат :", toolbar, drawerLayout, this);

            setRULES();
            workingWithWordsMap();
            calculateAndShow();

            ((ViewGroup)columnsForResize[0].getParent()).getLayoutTransition()
                    .enableTransitionType(LayoutTransition.CHANGING);
            for(int i = 0; i < columnsForResize.length; i++) {
                final int buffi = i;
                columnsForResize[i].setOnClickListener(new View.OnClickListener() {
                    @Override
```

```
                public void onClick(View v) {
                    scaleUpColumn(buffi);
                }
            });
        }

        findViewById(R.id.dictant_result_close).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onBackPressed();
            }
        });
    }

    @Override
    public void onBackPressed() {
        ActivityOptionsCompat options =
ActivityOptionsCompat.makeSceneTransitionAnimation(
                this, findViewById(R.id.drawer_view),
                getResources().getString(R.string.shared_drawer_translation)
        );
        if(startedFrom == STARTED_FROM_DICTANT)
            startActivity(new Intent(this, MainMenuActivity.class), options.toBundle());
        else if(startedFrom == STARTED_FROM_RESULTS_ACTIVITY)
            startActivity(new Intent(this, AllResultsActivity.class), options.toBundle());
        super.mustBeFinished = true;
    }

    private void readExtras(){
        Intent enterIntent = getIntent();
        cx = enterIntent.getIntExtra("button_X", rootLayout.getWidth()/2);
        cy = enterIntent.getIntExtra("button_Y", rootLayout.getHeight()/2);
        dictantResult = DictantResult.fromString(enterIntent.getStringExtra("dictant_res"));
        startedFrom = enterIntent.getIntExtra("started_from", 0);
    }

    private void calculateAndShow(){
        if(dictantResult.words.size() != 0)
            progress = dictantResult.wrongWords.size() * 100 / dictantResult.words.size();
        else
            progress = 100;

        String markMessage = "Твоя оцінка : ";
        SpannableString spannable = new SpannableString(markMessage + Math.round(12 *
(1 - (progress  / 100))) + "б;");
        int buffMarkColor = Funcs.evaluateColor(
            getResources().getColor(R.color.app_color),
            getResources().getColor(R.color.red_app_color),
            progress/100
        );
        spannable.setSpan(new ForegroundColorSpan(buffMarkColor),
```

```
                markMessage.length(), spannable.length() - 1,
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
            ((TextView)findViewById(R.id.dictant_result_mark)).setText(spannable);

            spannable = new SpannableString("Час : " +
(int)Math.ceil((double)dictantResult.writingTime/60000) + " хв;");
            spannable.setSpan(new
ForegroundColorSpan(getResources().getColor(R.color.higher_than_secondary)),
                7, spannable.length() - 4, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
            ((TextView)findViewById(R.id.dictant_result_writing_time)).setText(spannable);

            int j = 0;
            LayoutInflater inflater = getLayoutInflater();
            for(int i = 0; i < dictantResult.words.size(); i++){
               View toAdd = inflater.inflate(R.layout.dictant_result_table_item, null);

((LinearLayout)toAdd).getLayoutTransition().enableTransitionType(LayoutTransition.CHANGIN
G);
               TextView wordNum = toAdd.findViewById(R.id.dictant_result_table_word_num);
               wordNum.setText((i + 1) + ")");
               boolean isGood = true;
               if(j < dictantResult.wrongWords.size() && dictantResult.wrongWords.get(j) == i){
                  j++;
                  isGood = false;
               }
               if(isGood)
                  wordNum.setBackgroundColor(getResources().getColor(R.color.app_color));
               else
                  wordNum.setBackgroundColor(getResources().getColor(R.color.red_app_color));

               toAdd.setTag(R.string.result_activity_tag, dictantResult.words.get(i));

((TextView)toAdd.findViewById(R.id.dictant_result_table_word)).setText(TextWork.formatWord(
dictantResult.words.get(i)));

((TextView)toAdd.findViewById(R.id.dictant_result_table_userinput)).setText(dictantResult.users
Answers.get(i));
               ((TextView)toAdd.findViewById(R.id.dictant_result_table_mistake))

.setText(TextWork.highlightErrors(TextWork.formatWord(dictantResult.words.get(i)),
                        TextWork.formatUserInput(dictantResult.usersAnswers.get(i)), this,
                        R.color.red_app_color, R.color.app_color, -1));

               resultTable.addView(toAdd);
            }
         }
         private void animateProgress(){
            final TextView percent = findViewById(R.id.dictant_result_percent);
            ObjectAnimator animator = ObjectAnimator.ofInt(progressBar, "progress", 100,
Math.round(progress));
            animator.setDuration(PROGRESS_ANIMATION_DURATION);
            animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
```

```
                @Override
                public void onAnimationUpdate(ValueAnimator animation) {
                    percent.setText((100 - (int)animation.getAnimatedValue()) + "%");
                }
            });
            animator.start();
        }

        private void scaleUpColumn(int num){
            for(int i = 0; i < resultTable.getChildCount(); i++){
                if (i != 1) {
                    LinearLayout row = (LinearLayout) resultTable.getChildAt(i);
                    for (int j = 2; j < row.getChildCount(); j += 2)
                        if (j / 2 - 1 == num && selectedColumn != num)
                            row.getChildAt(j).setLayoutParams(new LinearLayout.LayoutParams(
                                    0, ViewGroup.LayoutParams.WRAP_CONTENT, 3.0f));
                        else
                            row.getChildAt(j).setLayoutParams(new LinearLayout.LayoutParams(
                                    0, ViewGroup.LayoutParams.WRAP_CONTENT, 1.0f));
                }
            }
            if(num == selectedColumn)
                selectedColumn = 4;
            else
                selectedColumn = num;
        }

        private void workingWithWordsMap(){
            RULES = TextWork.openFile(this, TextWork.RULES_NAME);
            String recomendationBegining = "Рекомендовано прочитати :\n";
            Comparator<TreeMap.Entry<Integer, Integer>> reverseOrederComaparator =
                new Comparator<TreeMap.Entry<Integer, Integer>>() {
                    @Override
                    public int compare(Map.Entry<Integer, Integer> o1, Map.Entry<Integer, Integer>
o2) {
                        return o2.getValue().compareTo(o1.getValue());
                    }
                };

            final ArrayList<TreeMap.Entry<Integer, Integer>> sortedWordsMap =
                    new ArrayList<>(dictantResult.wordsMapDump.entrySet());
            Collections.sort(sortedWordsMap, reverseOrederComaparator);

            ArrayList<Integer> was = new ArrayList<>();
            SpannableStringBuilder string = new
SpannableStringBuilder(recomendationBegining);
            for(int i = 0; i < sortedWordsMap.size() && was.size() <
NEW_PARAGRAFS_TO_SHOW; i++){
                final int buff_par = sortedWordsMap.get(i).getKey();
                if(!was.contains(buff_par)){
                    was.add(buff_par);
                    String buff_s = TextWork.PARAGRAF_SYMBOL + buff_par;
```

```
            if(was.size() < NEW_PARAGRAFS_TO_SHOW - 1) buff_s += ", ";
            else buff_s += ". ";
            string.append(buff_s);
            string.setSpan(new ClickableSpan() {
               @Override
               public void onClick(@NonNull View widget) {
                  new PravopysDialog(ResultActivity.this, buff_par).show();
               }
            }, string.length() - buff_s.length(), string.length() - 2,
                  Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
         }
      }

      TextView recomendationText = findViewById(R.id.dictant_result_recomendation);
      recomendationText.setText(string);
      recomendationText.setMovementMethod(LinkMovementMethod.getInstance());
   }
}
```

AllResultsActivity.java

```
package ua.roman.dictations;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.widget.Toolbar;
import androidx.drawerlayout.widget.DrawerLayout;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import ua.roman.dictations.algo.DictantResult;
import ua.roman.dictations.helper.CommonActivity;

public class AllResultsActivity extends CommonActivity {

   private Toolbar toolbar;
   private DrawerLayout drawerLayout;
   private ListView listView;

   private DateFormat dateFormat = new SimpleDateFormat(
```

```
        "dd MMMM, yy року : ", Locale.forLanguageTag("uk"));
//DictantResult[] dictantResults;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.all_results_with_drawer);

    toolbar = findViewById(R.id.main_menu_toolbar);
    drawerLayout = findViewById(R.id.all_results_drawer_layout);
    super.initToolbarDrawer("Твої результати :", toolbar, drawerLayout, this);

    listView = findViewById(R.id.all_results_list);
    //dictantResults = new DictantResult[DictantResult.getNumberOfSaved(this)];
    fillListView();
}

private void fillListView(){
    int num = DictantResult.getNumberOfSaved(this);
    if(num > 0) {
        /*for (int i = 0; i < dictantResults.length; i++)
            dictantResults[i] = DictantResult.read(this, i);*/
        AllResultsListAdapter adapter = new AllResultsListAdapter(this);
        listView.setAdapter(adapter);
        Date prevDate = new Date(0);
        for(int i = num-1; i >= 0; i--) {
            DictantResult buff = DictantResult.read(this, i);
            if(!prevDate.equals(buff.date))
                adapter.add(new AllResultsListHeader(buff.date));
            prevDate = buff.date;
            adapter.add(new AllResultsListResult(buff, i));
        }
    }
    else
        findViewById(R.id.all_result_empty_message).setVisibility(View.VISIBLE);
}

enum AllResultsItemTypes { ITEM, HEADER }
interface AllResultsListItem{
    @NonNull
    int getItemType();
    @NonNull
    View getView(LayoutInflater inflater, View convertView);
}
class AllResultsListResult implements AllResultsListItem{
    private DictantResult dictantResult;
    private int pos;

    AllResultsListResult(DictantResult result, int position){
        dictantResult = result;
        pos = position;
    }
```

```
            @NonNull
            @Override
            public View getView(LayoutInflater inflater, View convertView) {
               if(convertView == null)
                  convertView = inflater
                        .inflate(R.layout.all_results_list_item, null);
               int wordsNum = dictantResult.words.size();
               float res = 1f - (float) dictantResult.wrongWords.size()/wordsNum;
               int writingTime = (int) Math.ceil((double) dictantResult.writingTime/60000);
               int progress = (int)((1 - res) * 100f);

((TextView)convertView.findViewById(R.id.all_results_item_number)).setText((pos + 1) + ")");

((TextView)convertView.findViewById(R.id.all_results_item_percent)).setText((int)(res * 100f) +
"%");

((ProgressBar)convertView.findViewById(R.id.all_results_item_bar)).setProgress(Math.round(pro
gress));

((TextView)convertView.findViewById(R.id.all_results_item_mark)).setText("Оцінка : " +
Math.round(res * 12f) + "б;");
               ((TextView)convertView.findViewById(R.id.all_results_item_writing_time))
                     .setText("Час написання : " + writingTime + "хв;");

((TextView)convertView.findViewById(R.id.all_results_item_words_num)).setText("Слів : " +
wordsNum + ";");
               convertView.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View view) {
                     int[] buff = new int[2];
                     view.getLocationInWindow(buff);
                     int cx = view.getWidth() / 2;
                     int cy = buff[1] + view.getHeight() / 2;
                     Intent intent = new Intent(AllResultsActivity.this, ResultActivity.class);
                     intent.putExtra("button_X", cx);
                     intent.putExtra("button_Y", cy);
                     intent.putExtra("dictant_res", dictantResult.toString());
                     intent.putExtra("started_from",
ResultActivity.STARTED_FROM_RESULTS_ACTIVITY);
                     startActivity(intent);
                     finish();
                  }
               });
               return convertView;
            }

            @Override
            public int getItemType() {
               return AllResultsItemTypes.ITEM.ordinal();
            }
         }
```

```
        class AllResultsListHeader implements AllResultsListItem{
            Date d;

            AllResultsListHeader(Date date){
                d = date;
            }

            @NonNull
            @Override
            public int getItemType() {
                return AllResultsItemTypes.HEADER.ordinal();
            }

            @NonNull
            @Override
            public View getView(LayoutInflater inflater, View convertView) {
                if(convertView == null)
                    convertView = inflater
                            .inflate(R.layout.all_results_list_header, null);
                ((TextView)convertView.findViewById(R.id.all_results_list_header_date))
                        .setText(dateFormat.format(d));
                return convertView;
            }
        }

        class AllResultsListAdapter extends ArrayAdapter<AllResultsListItem>{
            LayoutInflater inflater;

            AllResultsListAdapter(@NonNull Context context) {
                super(context, 0);
                inflater = LayoutInflater.from(context);
            }

            @Override
            public int getViewTypeCount() {
                return AllResultsItemTypes.values().length;
            }

            @Override
            public int getItemViewType(int position) {
                return getItem(position).getItemType();
            }

            @NonNull
            @Override
            public View getView(final int position, @Nullable View convertView, @NonNull
ViewGroup parent) {
                return getItem(position).getView(inflater, convertView);
            }
        }
    }
```

Dictant.java

```
package ua.roman.dictations.dictants;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.ColorStateList;
import android.os.Bundle;
import android.os.Handler;
import android.os.PersistableBundle;
import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.view.View;
import android.view.ViewPropertyAnimator;
import android.view.ViewTreeObserver;
import android.view.animation.AnticipateInterpolator;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.appcompat.widget.Toolbar;
import androidx.constraintlayout.widget.ConstraintLayout;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.sql.Date;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Random;
import java.util.regex.Pattern;

import ua.roman.dictations.MainMenuActivity;
import ua.roman.dictations.R;
import ua.roman.dictations.ResultActivity;
import ua.roman.dictations.algo.DictantResult;
import ua.roman.dictations.algo.Funcs;
import ua.roman.dictations.algo.TextWork;
import ua.roman.dictations.algo.Words_rule;
import ua.roman.dictations.helper.CommonActivity;

public class Dictant extends CommonActivity {
   //Circular reveal components
```

```
protected View rootLayout;
protected int cx = 0;
protected int cy = 0;

//Dictant details
protected int numberOfWords;
protected String wordChosen;
protected HashMap<Integer, Integer> wordsMap;
public DictantResult dictantResult;

//Dictant timers
protected Toolbar toolbar;
protected Handler timer;
protected Runnable tick;
protected long startTime = 0;
protected long endTime = -1;

//Necessary variables
protected SharedPreferences sharedPreferences;
protected Random r;

//Message panel variables
protected ConstraintLayout messagePanel;
protected int messagePanelMessageId;
protected static final int NO_MESSAGE_ID = 0;
protected static final int DICTANT_ENDED_MESSAGE_ID = 1;
protected static final int TOP_PANEL_USER_HINT_MESSAGE_ID = 2;
public boolean isToShowTopPanelHint = false;
protected Runnable removeMessagesRunnable;

protected static final int RANDOM_WORDS_COUNT = 100;
protected static final int WORDMAP_ADD_WHEN_ERROR = 5;
protected static final int WORDMAP_DIVIDE_BY_WHEN_GOOD = 2;
protected static final float SLOW_SPEED_MULTIPLIER = 0.6f;
protected static final int MIN_VIEW_EXIT_TIME = 500;
protected static final int MAX_VIEW_EXIT_TIME = 600;

public static final int MIN_WORDS_COUNT = 6;
public static final int MAX_WORDS_COUNT = 24;

//Text input views
protected ProgressBar progress;
protected FloatingActionButton dictantCheckButton;
protected FloatingActionButton repeatSpeakButton;
protected FloatingActionButton repeatSpeakSlowlyButton;
protected EditText userInput;
protected TextView inputHint;

protected float ttsSpeachRate = 1.0f;
protected TextToSpeech tts;
```

```
        protected void CommonVarsInit(
            Bundle savedInstanceState, final AnimatorListenerAdapter animatorListener){
            //Circular reveal init
            //Reading circularRevealInfo.
            Intent enterIntent = getIntent();
            rootLayout = findViewById(R.id.dictant_root_layout);
            cx = enterIntent.getIntExtra("button_X", rootLayout.getWidth()/2);
            cy = enterIntent.getIntExtra("button_Y", rootLayout.getWidth()/2);
            numberOfWords = enterIntent.getIntExtra("words_num", 0);

            if (savedInstanceState == null) {
                rootLayout.setVisibility(View.INVISIBLE);
                ViewTreeObserver viewTreeObserver = rootLayout.getViewTreeObserver();
                if (viewTreeObserver.isAlive())
                    viewTreeObserver.addOnGlobalLayoutListener(new
ViewTreeObserver.OnGlobalLayoutListener() {
                        int fff;
                        @Override
                        public void onGlobalLayout() {
                            Funcs.circularRevealActivity(rootLayout, cx, cy, animatorListener);
                            rootLayout.getViewTreeObserver().removeOnGlobalLayoutListener(this);
                        }
                    });
            }

            //Common vars init
            Funcs.StatusBarStyling(this);
            r = new Random();
            sharedPreferences = getSharedPreferences(CommonActivity.SHARED_PREF_NAME,
Context.MODE_PRIVATE);
            messagePanel = findViewById(R.id.dictant_message_panel);

            //Init timers
            toolbar = findViewById(R.id.main_menu_toolbar);
            toolbar.setTitle("00:00");


            removeMessagesRunnable = new Runnable() {
                @Override
                public void run() {
                    removeMessage();
                }
            };

            dictantResult = new DictantResult();

            progress = findViewById(R.id.dictant_user_progress_bar);
            progress.setProgress(0);

            userInput = findViewById(R.id.dictant_user_input);
            dictantCheckButton = findViewById(R.id.dictant_check_button);
            inputHint = findViewById(R.id.dictant_input_hint);
```

```
    repeatSpeakButton = findViewById(R.id.dictant_speak);
    repeatSpeakButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            speak(false);
        }
    });
    repeatSpeakSlowlyButton = findViewById(R.id.dictant_slow_speak);
    repeatSpeakSlowlyButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            speak(true);
        }
    });

    //Loading rules file
    String[] buff = TextWork.openFile(this, TextWork.WORDS_NAME)
            .split(Pattern.quote(TextWork.WORDS_SPLITTER));
    setRULES();
    wordsMap = (HashMap<Integer, Integer>) Funcs.readWordsMap(
            getApplicationContext(), Funcs.WORDS_MAP_SAVE_NAME);
    if(wordsMap == null) wordsMap = new HashMap<>();

    WORDS = new String[RANDOM_WORDS_COUNT];
    int buff_s = buff.length, a;
    ArrayList<Integer> was = new ArrayList<>();
    for(int i = 0; i < RANDOM_WORDS_COUNT; i++){
        while (was.contains(a = r.nextInt(buff_s)));
        WORDS[i] = buff[a];
        was.add(a);
    }
}

protected void startTimer(){
    timer = new Handler();
    tick = new Runnable() {
        @Override
        public void run() {
            toolbar.setTitle(new SimpleDateFormat("mm:ss")
                    .format(new Date(System.currentTimeMillis() - startTime)));
            timer.postDelayed(tick, 1000);
        }
    };
    timer.postDelayed(tick, 1000);
}
protected void stopTimer(){
    timer.removeCallbacksAndMessages(null);
}

protected void showMessage(final int messageId){
    if(messageId == messagePanelMessageId) return;
```

```
            removeMessage();
            messagePanelMessageId = messageId;
            switch (messageId){
               case TOP_PANEL_USER_HINT_MESSAGE_ID:
                  getLayoutInflater().inflate(
                        R.layout.dictant_use_top_panel_message,
                        messagePanel, true);
                  messagePanel.setVisibility(View.VISIBLE);
                  break;
               case DICTANT_ENDED_MESSAGE_ID:
                  getLayoutInflater().inflate(
                        R.layout.dictant_dictant_ended_message,
                        messagePanel, true);
                  messagePanel.setVisibility(View.VISIBLE);
                  break;
               default:
                  messagePanelMessageId = NO_MESSAGE_ID;
                  messagePanel.setVisibility(View.GONE);
                  break;
            }
         }
         public void removeMessage(){
            if(messagePanelMessageId != NO_MESSAGE_ID) {
               messagePanelMessageId = NO_MESSAGE_ID;
               messagePanel.removeAllViews();
               messagePanel.setVisibility(View.GONE);
            }
         }

         protected void LoadNewWord(){
            wordChosen = WORDS[Funcs.getBestWordIndex(wordsMap, WORDS,
dictantResult)];
            int checkForHintRes = TextWork.checkForHint(wordChosen);
            if(checkForHintRes == TextWork.CONTAINS_CAPITAL_LETTER){
               inputHint.setVisibility(View.VISIBLE);
               inputHint.setText(R.string.contains_capital_letter);
            }
            else if(checkForHintRes == TextWork.CONTAINS_NUMER){
               inputHint.setVisibility(View.VISIBLE);
               inputHint.setText(R.string.contains_number);
            }
            else inputHint.setVisibility(View.INVISIBLE);
            speak(false);
         }
         protected boolean checkEnteredWord(String userInput){
            Words_rule[] rules = TextWork.getWordsRuleFromString(wordChosen);
            boolean ret = dictantResult.add(wordChosen, userInput);
            if(!ret){
               for(Words_rule rule : rules) {
                  if (wordsMap.containsKey(rule.paragraf))
                     wordsMap.put(rule.paragraf, wordsMap.get(rule.paragraf) +
```

```
WORDMAP_ADD_WHEN_ERROR);
                    else
                        wordsMap.put(rule.paragraf, WORDMAP_ADD_WHEN_ERROR);
                }
            }
            else{
                for(Words_rule rule : rules)
                    if(wordsMap.containsKey(rule.paragraf))
                        wordsMap.put(rule.paragraf, wordsMap.get(rule.paragraf) /
WORDMAP_DIVIDE_BY_WHEN_GOOD);
            }
            //#DEBUG
            Log.d("Words", wordsMap.toString());
            return ret;
        }

        protected void onDisposeVars(){
            tts.stop();
        }
        protected void forceDictantStop() {
            onDisposeVars();
            Funcs.unCircularRevealActivity(MainMenuActivity.class, Dictant.this, rootLayout,
                    rootLayout.getWidth()/2, rootLayout.getHeight()/2);
            Dictant.super.mustBeFinished = true;
        }
        protected void endDictant(){
            if(dictantResult.words.size() >= MIN_WORDS_COUNT) {
                onDisposeVars();
                Funcs.writeWordsMap(this, wordsMap, Funcs.WORDS_MAP_SAVE_NAME);
                if (endTime != -1)
                    dictantResult.end(endTime - startTime, wordsMap);
                else
                    dictantResult.end(System.currentTimeMillis() - startTime, wordsMap);
                dictantResult.save(this);
                Intent intent = new Intent(Dictant.this, ResultActivity.class);
                int[] outButtonLoc = new int[2];
                dictantCheckButton.getLocationInWindow(outButtonLoc);
                intent.putExtra("button_X", (int) (outButtonLoc[0] + (float)
dictantCheckButton.getWidth() / 2));
                intent.putExtra("button_Y", (int) (outButtonLoc[1] + (float)
dictantCheckButton.getHeight() / 2));
                intent.putExtra("dictant_res", dictantResult.toString());
                intent.putExtra("started_from", ResultActivity.STARTED_FROM_DICTANT);
                startActivity(intent);
                finish();
            }
            else{
                new AlertDialog.Builder(this, R.style.AlertDialog).setTitle("Ваш результат буде
втрачено!")
                        .setMessage("Ваш результат буде втрачено, так як ви написали замалу
кількість слів. " +
                                "Ви дійсно хочете закінчити диктант").setPositiveButton("Так",
```

```
                    new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) { forceDictantStop();
}
                    }).setNegativeButton("Hi", null).create().show();
            }
        }
        protected void inputPanelDictantEndAnimation() {
            inputHint.setVisibility(View.INVISIBLE);
            showMessage(DICTANT_ENDED_MESSAGE_ID);
            final AnimatorSet dictantCheckAnimator = new AnimatorSet();
            ObjectAnimator backgroundTintAnim = ObjectAnimator.ofArgb(dictantCheckButton,
"backgroundTint",
                    getResources().getColor(R.color.app_color),
getResources().getColor(R.color.paragraf_highlight_bg));
            backgroundTintAnim.addUpdateListener(new
ValueAnimator.AnimatorUpdateListener() {
                @Override
                public void onAnimationUpdate(ValueAnimator animation) {

dictantCheckButton.setBackgroundTintList(ColorStateList.valueOf((int)animation.getAnimatedVal
ue()));
                }
            });
            ObjectAnimator sizeAnim = ObjectAnimator.ofInt(dictantCheckButton, "customSize",
                    Funcs.convertDpToPixel(70, this), Funcs.convertDpToPixel(140, this));
            sizeAnim.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
                @Override
                public void onAnimationUpdate(ValueAnimator animation) {
                    if((int)animation.getAnimatedValue() >= Funcs.convertDpToPixel(71,
Dictant.this))
                        dictantCheckButton.setImageResource(R.drawable.ic_check);
                }
            });
            ObjectAnimator rotationAnim = ObjectAnimator.ofFloat(dictantCheckButton,
"rotation", 0, 720);

dictantCheckAnimator.play(backgroundTintAnim).after(500).with(sizeAnim).with(rotationAnim);
            dictantCheckAnimator.setDuration(MAX_VIEW_EXIT_TIME);
            dictantCheckAnimator.setInterpolator(new AnticipateInterpolator());
            dictantCheckAnimator.start();
            Funcs.animateViewExitX(repeatSpeakButton, r.nextInt(MAX_VIEW_EXIT_TIME-
MIN_VIEW_EXIT_TIME) + MIN_VIEW_EXIT_TIME);
            Funcs.animateViewExitX(repeatSpeakSlowlyButton,
r.nextInt(MAX_VIEW_EXIT_TIME-MIN_VIEW_EXIT_TIME) + MIN_VIEW_EXIT_TIME);
            Funcs.animateViewExitX(userInput, r.nextInt(MAX_VIEW_EXIT_TIME-
MIN_VIEW_EXIT_TIME) + MIN_VIEW_EXIT_TIME);
            Funcs.hideKeyBoard(this);
        }

        protected void speak(boolean slow){
            if(slow) tts.setSpeechRate(ttsSpeachRate * SLOW_SPEED_MULTIPLIER);
```

```
        else tts.setSpeechRate(ttsSpeachRate);
        String toSpeak = TextWork.formatWord(wordChosen);
        tts.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null, "");
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        overridePendingTransition(R.anim.dont_move, R.anim.dont_move);
    }
    @Override
    protected void onStop() {
        stopTimer();
        super.onStop();
    }
    @Override
    protected void onResume() {
        if(endTime == -1)
            startTimer();
        super.onResume();
    }
}
```

DictantActivity.java

```
package ua.roman.dictations.dictants;

import android.animation.ObjectAnimator;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.os.Looper;
import android.speech.tts.TextToSpeech;
import android.speech.tts.UtteranceProgressListener;
import android.view.MotionEvent;
import android.view.View;
import android.widget.LinearLayout;

import java.util.Locale;

import ua.roman.dictations.R;
import ua.roman.dictations.algo.TextWork;
import ua.roman.dictations.helper.CommonActivity;

public class DictantActivity extends Dictant {
    private DictantMovableTopPanelController topPanelController;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dictant_layout);
```

```
            CommonVarsInit(savedInstanceState, null);
            userInput.setOnTouchListener(new View.OnTouchListener() {
                int fff;
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                    topPanelController.setIsOpened(false);
                    return false;
                }
            });

            topPanelController = new DictantMovableTopPanelController(
                    (LinearLayout) findViewById(R.id.top_panel),
                    this, numberOfWords,
                    new DictantTopPanelPagerAdapter(
                            this, dictantResult
                    ));

            //Checking messages

if(!sharedPreferences.getBoolean(CommonActivity.SHARED_PREF_TOP_PANEL_MESSAGE_
SHOWN, false)) {
                isToShowTopPanelHint = true;
                sharedPreferences.edit().putBoolean(
                        CommonActivity.SHARED_PREF_TOP_PANEL_MESSAGE_SHOWN, true
                ).apply();
            }

            tts = new TextToSpeech(this, new TextToSpeech.OnInitListener() {
                int fff;
                @Override
                public void onInit(int status) {
                    tts.setLanguage(Locale.forLanguageTag("uk"));
                    ttsSpeachRate =
sharedPreferences.getFloat(CommonActivity.SHARED_PREF_TTS_SPEACH_RATE, 1.0f);

tts.setPitch(sharedPreferences.getFloat(CommonActivity.SHARED_PREF_TTS_PITCH, 1.0f));
                    LoadNewWord();
                    startTime = System.currentTimeMillis();
                    startTimer();
                    setupButtonOnClick();
                }
            });
            tts.setOnUtteranceProgressListener(new UtteranceProgressListener() {
                @Override
                public void onStart(String utteranceId) {}
                @Override
                public void onDone(String utteranceId) {}
                @Override
                public void onError(String utteranceId) {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
```

```
                        new AlertDialog.Builder(DictantActivity.this,
R.style.AlertDialog).setTitle("Помилка").setCancelable(false)
                                .setMessage("Через помилку з'єднання з інтернетом, синтезатор
голосу не може виконувати свою роботу." +
                                        "Будь ласка, відновіть з'єднання з інтернетом, або завершіть
диктант.")
                                .setPositiveButton("Відновити з'єднання", null)
                                .setNegativeButton("Завершити диктант", new
DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog, int which) { endDictant();
}
                                }).create().show();
                        }
                    });
                }
            });
        }

        @Override
        protected void onResume() {
            topPanelController.setIsOpened(false);
            super.onResume();
        }
        @Override
        public void onBackPressed() {
            if(topPanelController.getIsOpened())
                topPanelController.setIsOpened(false);
            else
                new AlertDialog.Builder(this, R.style.AlertDialog).setTitle("Ви дійсно хочете
закінчити диктант ?")
                    .setPositiveButton("Так", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            endDictant();
                        }
                    }).setNegativeButton("Hi", null).setCancelable(false)
                    .create().show();
        }

        @Override
        protected void LoadNewWord(){
            if(numberOfWords > dictantResult.words.size())
                super.LoadNewWord();
            else{
                endTime = System.currentTimeMillis();
                stopTimer();
                toolbar.setTitleTextColor(getResources().getColor(R.color.word_highlight_bg));

                inputPanelDictantEndAnimation();

                dictantCheckButton.setOnClickListener(new View.OnClickListener() {
```

```
                @Override
                public void onClick(View v) {
                    endDictant();
                }
            });
        }
    }
    private void checkEnteredWord(){
        boolean res = checkEnteredWord(userInput.getText().toString());
        topPanelController.updateData(res);
        if(isToShowTopPanelHint)
            showMessage(TOP_PANEL_USER_HINT_MESSAGE_ID);

        userInput.setText("");
        ObjectAnimator.ofInt(progress, "progress", 100 * dictantResult.words.size() /
numberOfWords)
                .setDuration(300).start();
        topPanelController.setCurrentItem(dictantResult.words.size() - 1);
        topPanelController.setIsOpened(false);
        topPanelController.isScrollingEnabled = true;
    }
    private void setupButtonOnClick(){
        dictantCheckButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!TextWork.formatUserInput(userInput.getText().toString()).equals("")
                        && userInput.getText() != null) {
                    checkEnteredWord();
                    LoadNewWord();
                }
                else{
                    userInput.setError("Будь ласка, введіть слово!");
                }
            }
        });
    }

    /*private void initTopPanel(){
        afterAns = new Handler();
        //topPanel = findViewById(R.id.top_panel);
        topPanel.setOnTouchListener(new View.OnTouchListener() {
            int fff;
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                Funcs.hideKeyBoard(DictantActivity.this);
                afterAns.removeCallbacksAndMessages(null);
                return topPanel.onTouchEvent(event);
            }
        });

        //TODO: implement class for working with top panel
        topPanelPager = findViewById(R.id.top_panel_pager);
```

```
        topPanelAdapter = new DictantTopPanelPagerAdapter(this);

        topPanelAdapter.setWord("Hello word");
        topPanelAdapter.addItem("Helo world");

        topPanelPager.setAdapter(topPanelAdapter);
        topPanelPager.setCurrentItem(0);
    }*/
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи