КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Мобільний застосунок для розширення тезаурусу користувача у формі гри Alias»

Виконав ___ _____
(Підпис)
Сокол Даніїл Юрійович
(прізвище, ім'я, по батькові)

Керівник Броварець Олександр Олександрович
(прізвище, ім'я, по батькові)
_____ _____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Засвідчую, що у цьому курсовому проекті немає запозичень з праць інших авторів без відповідних посилань. 90%

Студент _____

Київ – 2022

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Мобільний застосунок для розширення тезаурусу користувача у формі гри Alias»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Сокол Данііл Юрійович	

Назва роботи українською та англійською мовами

Мобільний застосунок для розширення тезаурусу користувача у формі гри Alias Mobile application Alias

Мета бакалаврської роботи, завдання

Мета бакалаврської роботи: Створення мобільного застосунку на базі платформи андроїд для розширення словника користувача План роботи: 1. Аналітичний огляд сучасних підходів до розробки та впровадження мобільних додатків на системі андроїд 2. Аналіз існуючих варіантів і вибір програмних засобів для самої реалізації андроїд додатків 3. Програмна реалізація додатку для андроїд

ПІБ, ступінь, звання наукового керівника роботи:
Броварець Олександр Олександрович, доц., к.т.н.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	09.10.2021	Виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	19.10.2021	Виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	21.10.2021	Виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	25.10.2022	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	01.11.2022	Виконано
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	21.12.2022	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	31.01.2022	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	30.03.2022	Виконано
9.	Подання роботи у першому варіанті	28.04.2022	Виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2022	Виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**23.05.2022**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	27.05.2022	Виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	10.06.2022	Виконано
14.	Захист кваліфікаційної роботи бакалавра	22.06.2022 23.06.2022 24.06.2022	

Здобувач вищої освіти _____ _____
(підпис)

Керівник _____ _____
(підпис)

ВІДОМІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1 сторінка
Календарний план дипломної роботи	1 сторінка
Відомість дипломної роботи	1 сторінка
Анотація	1 сторінка
Анотація (іноземною мовою-англійською)	1 сторінка
Зміст	1 сторінка
Вступ	1 сторінка
Розділ 1	2 сторінок
Розділ 2	1 сторінок
Розділ 3	11 сторінок
Висновки	1 сторінка
Перелік використаних джерел	1 сторінки

<table>
<tr><td></td><td></td><td></td><td></td><td colspan="3" rowspan="2">ДП ХХХХ 00.000.00</td></tr>
<tr><td></td><td>ПІБ</td><td>Підп.</td><td>Дата</td></tr>
<tr><td>Розробн.</td><td>Сокол Д.Ю.</td><td></td><td></td><td rowspan="4">Відомість дипломної роботи</td><td>Лист</td><td>Листів</td></tr>
<tr><td>Керівн.</td><td>Броварець О.О.</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Н/контр.</td><td>Базилюк А.М.</td><td></td><td></td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Зав.каф.</td><td>Плескач В.Л.</td><td></td><td></td></tr>
</table>

Анотація

Дипломна робота складається із 25 ст., 14 рис., 9 дж., 1 дод.

Метою дипломної роботи є розширення словника користувача розвиваючої гри на основі розробленого мобільного застосунку.

Для досягнення поставленої мети потрібно вирішити такі завдання:

- Визначити роль і перспективи мобільних застосунків для розширення словника на базі андроїд.
- Створити і описати концептуальну модель мобільного застосунку.
- Реалізувати та впровадити прототип мобільного застосунку Alias за допомогою Android studio.

Об'єктом дослідження є процес поповнення словника користувачів за допомогою гри**.**

Предметом дослідження є програмно-технічні, організаційні засади, принципи, підходи щодо реалізації мобільного застосунку за допомогою засобів Android studio, та призначених для організації ефективного навчання користувачів у формі гри.

Ключові слова: мобільний застосунок, Android studio, гра Alias.

Annotation

Thesis consists of 25 pages, 14 pictures, 9 sources, 1 complement.

The aim of the thesis is to create a mobile application based on the Android platform to expand the user's vocabulary.

To achieve this goal you need to solve the following tasks:

- Create and describe a conceptual model of a mobile application;
- Identify the role and prospects of mobile applications to expand the vocabulary based on android.
- Implement a software prototype of the Alias mobile application using Android studio.

The object of research is the process of replenishing the vocabulary of users with the help of games.

The subject of the study is software and hardware, organizational principles, principles, approaches to the implementation of mobile applications using Android studio, and designed to organize effective training of users in the form of games.

Keywords: mobile application, Android studio, Alias game.

ЗМІСТ

ВСТУП

Поява загальнодоступного Інтернету посприяла загальній соціалізації людства, поширенні різних культур та появі багатьох неологізмів. Хтось їх використовує, хтось ні, але все одно вони досить тісно вплелись в нашу повсякденність. Задля того, щоб пояснити ці нові терміни для себе або для іншої людини можна використати такий словник як UrbanDictionary або задля того щоб полегшити засвоєння скористатись такою грою як Alias. Ця гра заключається в тому, що необхідно на швидкість пояснити гравцям своєї команди як можливо більше слів, використовуючи інші слова для опису. Залежно від тематики гри і варіантів мови складність може варіюватись, але навіть в такому випадку це буде корисний і веселий відпочинок з друзями чи знайомими.

РОЗДІЛ 1
Вивчення та огляд існуючих підходів до проектування мобільних застосунків

1.1 Поняття мобільний застосунок

Мобільний застосунок - програмне забезпечення, призначене для роботи на мобільних пристроях, таких як смартфони, планшети, годинники та інше.

Спершу мобільні застосунки в основному використовувались для зручного доступу до пошти, але високий попит призвів до поширення мобільного ПЗ на такі області як: ігри, спілкування, карти, застосунки для бізнесу, редактори фото або відео, тощо.

Мобільні пристрої зачасту продаються з попередньо встановленими додатками, частину з яких при бажанні можна вилучити. Інші застосунки можна отримати завантаживши їх з Інтернету або з платформи розповсюдження. Такі платформи називаються магазинами застосунків. Існують як офіційні магазини такі як Apple AppStore чи Google Play Market, так і неофіційні (Cydia, F-Droid, тощо). Застосунки можуть бути як платними так і безкоштовними, безкоштовні додатки зачасту містять рекламу та/або мікротранзакції.

Спочатку мобільні застосунки пропонувалися як інструменти для контролю та оперування загальних потоків інформації, включаючи електронну пошту, календар, контакти, фондовий ринок та інформацію про погоду. А втім, попит та наявність інструментів для розробників зумовили швидке поширення застосунків і для інших категорій електронних пристроїв, які функціонують за допомогою настільних прикладних програм

1.2 Порядок розробки мобільного застосунку та його особливість

При розробці ПЗ для мобільних пристроїв необхідно враховувати їх обмеження та можливості. Мобільні пристрої працюють від маломісткого акумулятора, мають меншу потужність і деякі особливі функції на кшталт GPS або гіросенсору. Окрім цього не слід забувати про різноманіття розмірів і форм-факторів девайсів, їхню комплектацію та різницю між різними версіями систем.

Розробка програм для мобільних платформ вимагає використання спеціалізованих інтегрованих середовищ розробки. Мобільні застосунки спочатку тестуються в середовищі розробки за допомогою емуляторів, а потім перевіряються на справжніх пристроях. Емулятори забезпечують недорогий спосіб тестування програм на мобільних телефонах, яким розробники можуть не мати фізичного доступу.

1.3 Вивчення та огляд існуючих підходів до проектування мобільних застосунків

Нині двома самими популярними мобільними ОС є Android та iOS.

Для ОС Android розробка мобільних застосунків найчастіше виконується мовою Java – об’єктно-орієнтованій мові, на якій написано більша частина всіх додатків під Андроїд. Також більшої популярності набирає нова мова Kotlin. Поки доволі малий відсоток застосунків в Google Play написані мовою Kotlin, але з кожним роком кількість таких застосунків зростає.

Якщо розглядати iOS платформу, то тут також використовуються дві основних мови – Objective C та Swift. Нові застосунки все частіше пишуться саме на Swift, але Objective C все ще активний.

Головна особливість розробки мобільних застосунків полягає в форм-факторі пристроїв, під які пишеться програма.

Іншим важливим моментом в розробці є різноманітність пристроїв, під які створюється застосунок. Щоб він працював на різних мобільних пристроях, потрібно продумати різні способи взаємодії з застосунком, а також різні методи відображення інформації в ньому.

РОЗДІЛ 2

Аналіз можливих архітектурних рішень та вибір програмних засобів для реалізації прототипу мобільного застосунку

Беручи в увагу високу поширеність ОС Android застосунок отримає більше поширення саме на цій платформі. Відповідно до цього у нас з’являється два варіанти програмних засобів для реалізації застосунку: Android Studio та Unity.

Android Studio представляє собою інтегроване середовище для розробки мобільних застосунків на базі ОС Android. Це середовище було спроектоване на базі коду intelliJ, компанії JetBrains. Інструментарій надає змогу для розроблення застосунків не тільки для мобільних пристроїв та планшетів, а й для інших пристроїв з операційною системою Android, такі як телевізори, автомобілі, холодильники. Середовище для розробки було налаштоване для роботи з типовими задачами, які були обрані для виконання під час проектування мобільного застосунку для Android. У середовище були введені певні інструменти для більш гнучкого тестування та зручний спосіб перевірки на можливість роботи з старішими версіями. Також можливість гнучко спроектувати відображення для пристроїв з різними екранами. В Android Studio було створено декілька унікальних функцій, для прикладу можна навести нову підсистему для зручного тестування і розгортання мобільних застосунків, вона заснована за допомогою використання інструментарії Gradle.

РОЗДІЛ 3

Проектування та програмна реалізація мобільного застосунку Alias на базі андроїд

3.1 Створення архітектури

Архітектура створеного програмного застосунку на базі ОС Android містить у собі комплект фрагментів, кожен з яких прив’язаний до власного вікна керування.

Фрагмент - виступає як клас мови Java, який зберігає свої дані у файлі з такою ж назвою.

XML - файл який містить у собі опис дизайну та розташування об’єктів з використанням xml-коду, для кожної активності відводиться власний файл.

Після встановлення застосунку, користувачу потрібно лише його запустити. Після чого відразу потрапляє на головну сторінку з кнопками початку гри, продовження(якщо попередня гра не була завершена) та правилами. При виборі початку гри користувач потрапляє на сторінку налаштування гри.

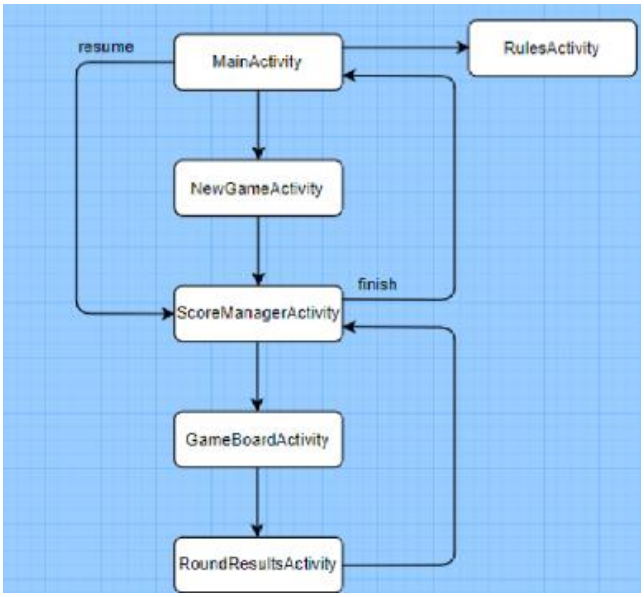

Рисунок 3.1.1 – Схема роботи програми

3.2 Макет інтерфейсу

Я створив макет GUI для застосунку таким чином:

- Головне меню
- Налаштування гри
- Меню між раундами
- Ігровий стіл
- Результати раунду

Першим користувачу відображається головне меню(рис. 3.1.2), на якому можна перейти на сторінку правил, почати нову гру або продовжити попередню

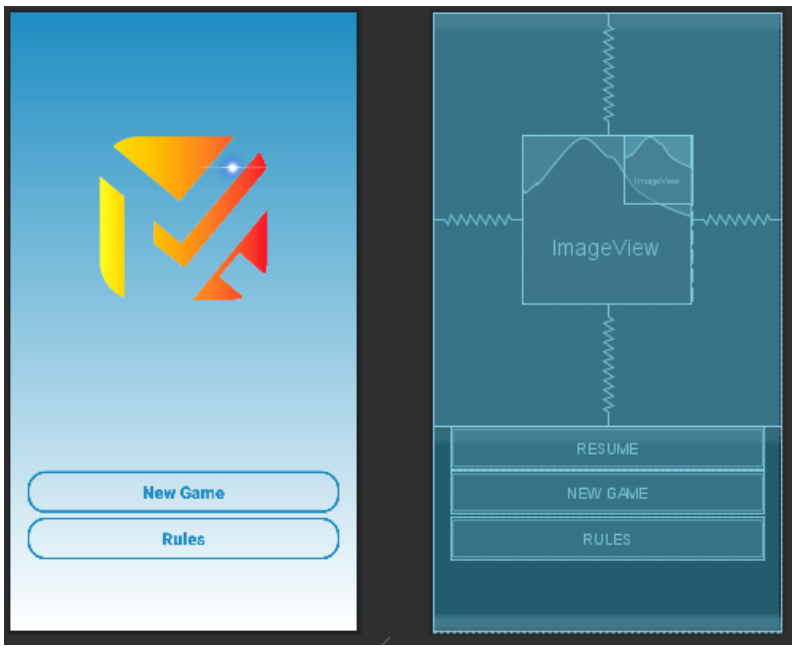

Рис. 3.1.2 Макет головного меню

Після натиснення кнопки “New Game” відкривається сторінка налаштувань гри(рис. 3.1.3), де користувач вибирає складність, час раунду, кількість очок для перемоги та формує команди.

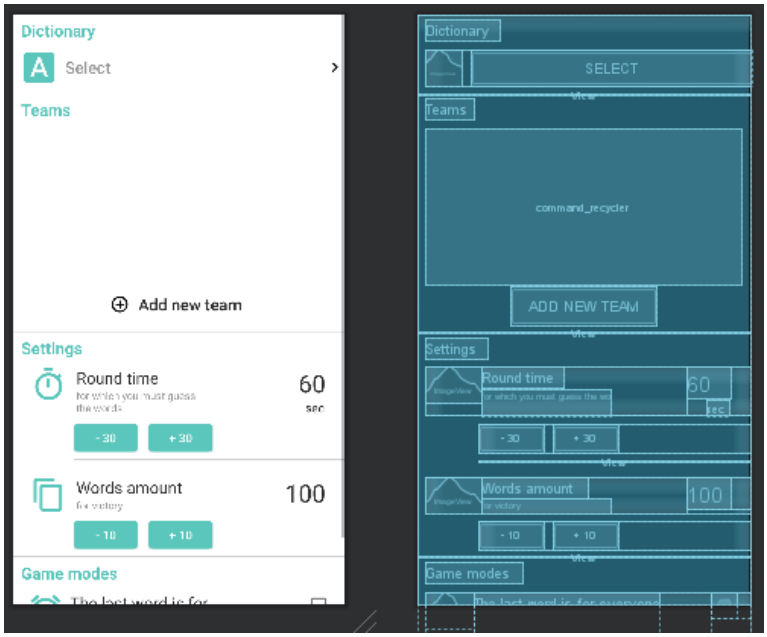

Рис. 3.1.3 Макет початку гри

Після завершення налаштування гри(а потім після кожного раунду) відкривається вікно з поточними результатами(рис. 3.1.4) кожної команди та показником команди яка грає в цьому раунді

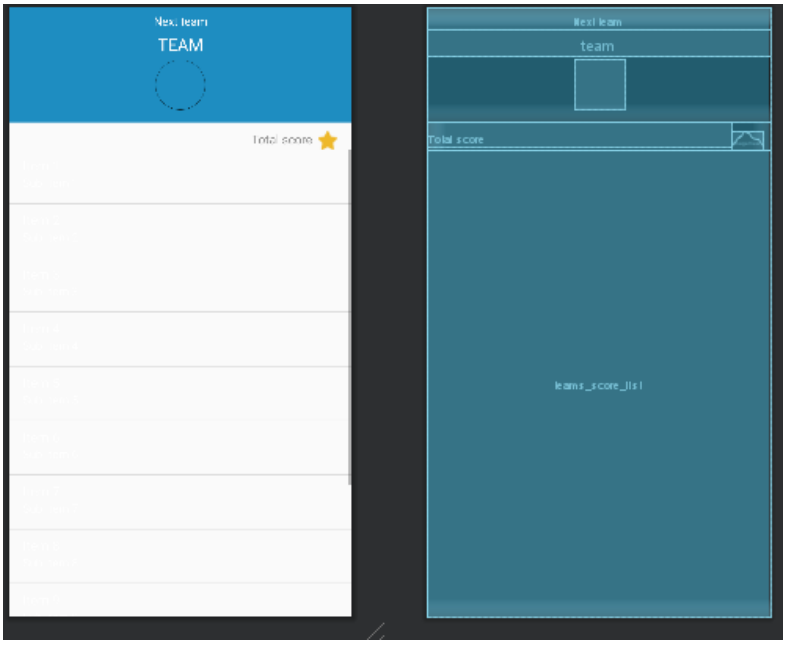

Рис. 3.1.4 Макет початку раунду

Коли починається раунд відкривається ігровий стіл(рис. 3.1.5) з таймером та картками зі словами. Для зручності користувача картки можна як переміщати, так і натискати на стрілки

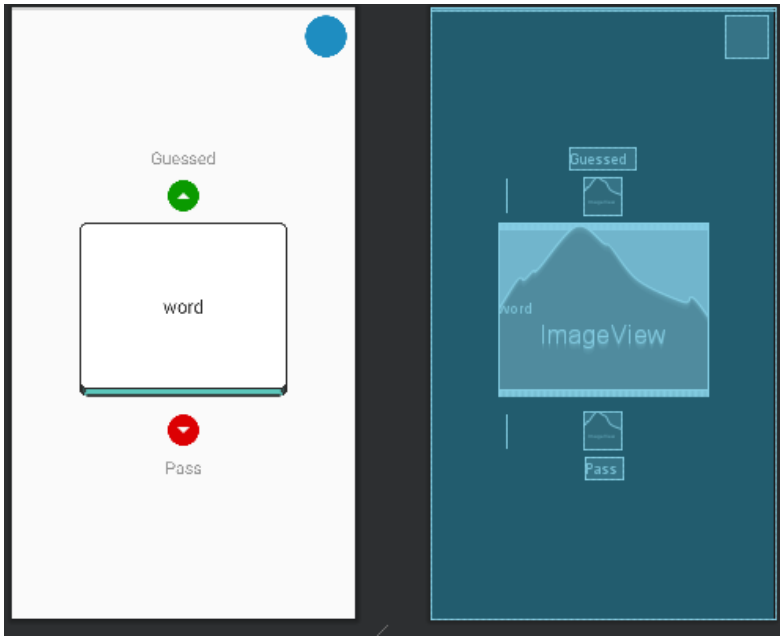

Рис. 3.1.5 Макет ігрового столу

Коли час спливає відкривається вікно з результатами раунду(рис. 3.1.6) і кількість набраних балів

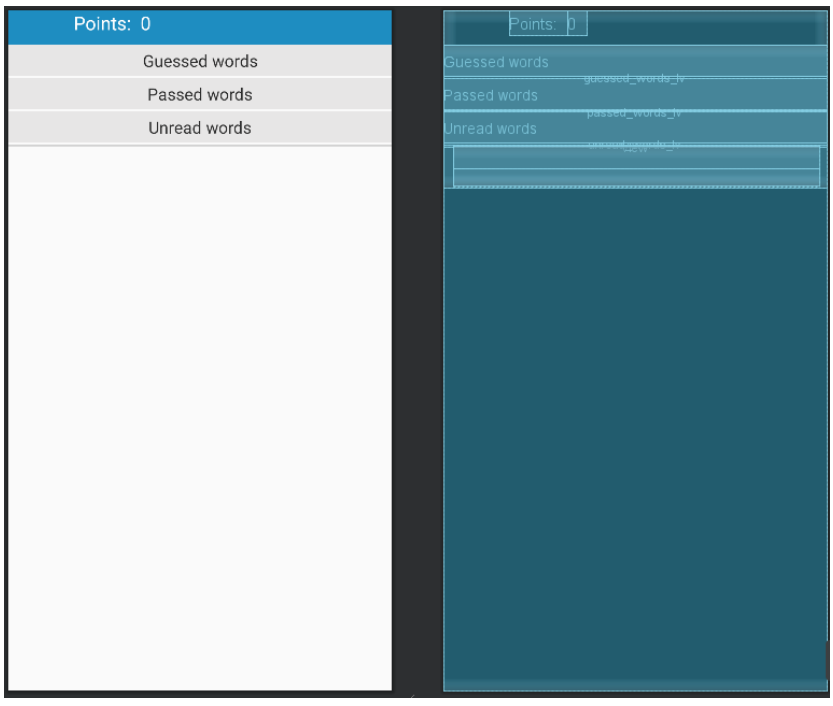

Рис. 3.1.6 Макет результатів раунду

3.3 Проведення тестування

Для перевірки був обраний спосіб через емулятор операційної системи Android вбудований в Android Studio.

При відкритті програми нас зустрічає головне меню.

Рис. 3.1.7 Головне меню

При натисненні кнопки “Rules” користувач може прочитати правила гри.

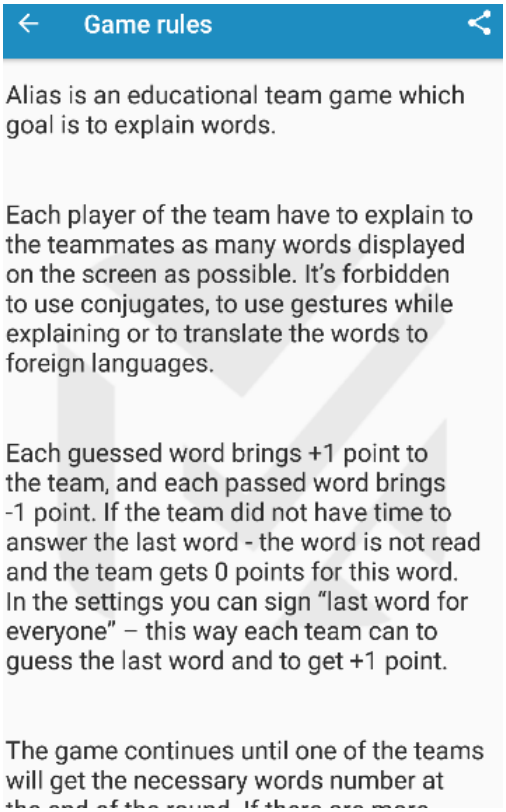

Рис 3.1.8 Сторінка правил

При натисненні кнопки “New Game” відкриваються налаштування нової гри. Користувач може вибрати складність словника, кількість і колір команд(2-4 команди), тривалість раунду та кількість правильних відповідей для завершення гри.

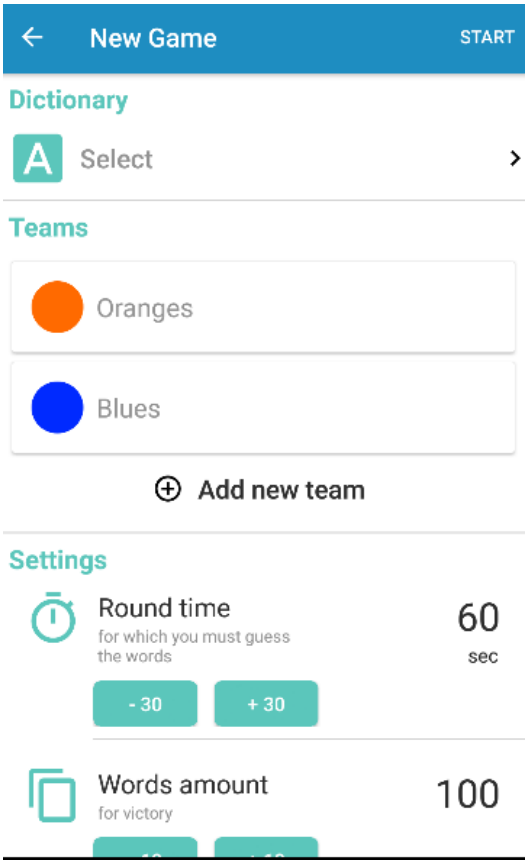

Рис. 3.1.9 Налаштування гри

Між раундами з’являється сторінка з проміжковими результатами і командою яка грає в цьому раунді

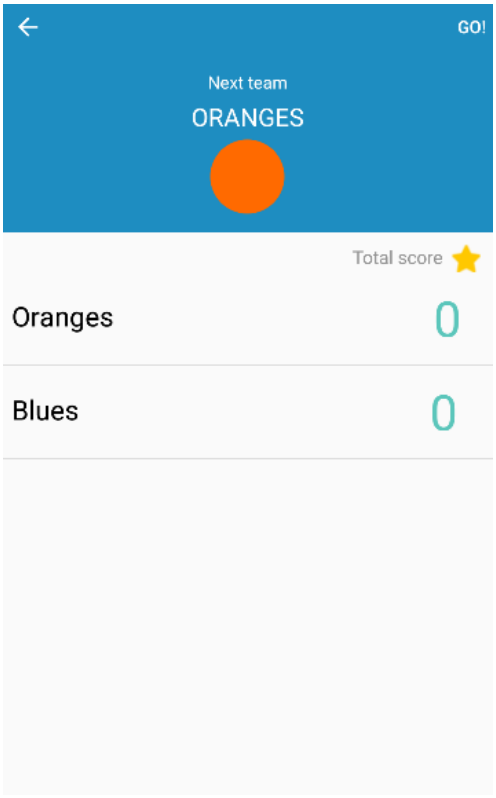

Рис. 3.1.10 Початок раунду

В процесі раунду користувачі з однієї команди поділяються: один пояснює слово, а інші намагаються його вгадати. Вірну відповідь можна помітити свайпом вгору або натисненням стрілки вгору, пропущене слово помічається відповідно свайпом вниз або натисненням стрілки вниз.

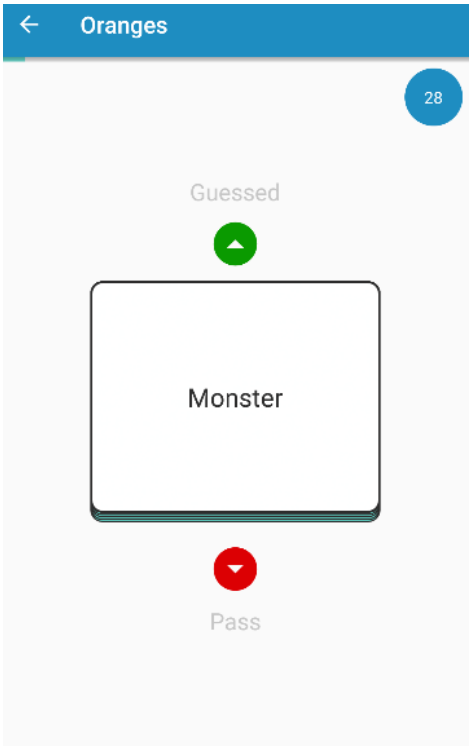

Рис. 3.1.11 Процесс раунду

В кінці раунду відбувається підрахунок балів, правильно вгадане слово дає 1 бал, пропущене віднімає один бал. Також якщо в процесі раунду стались якісь помилки, то можливо перенести слово з однієї категорії в іншу.

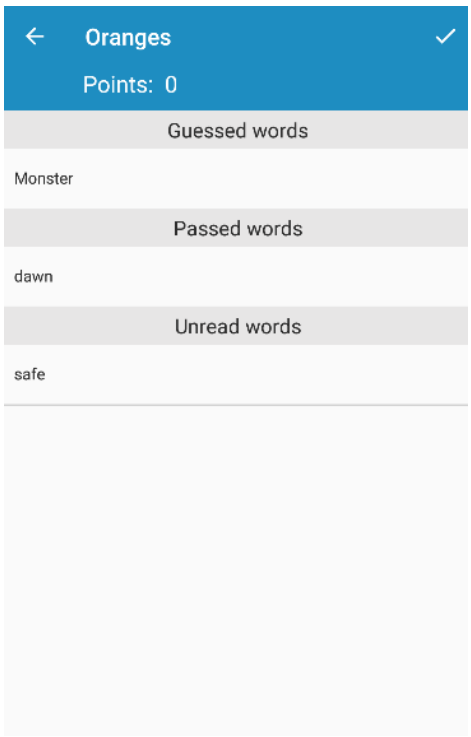

Рис. 3.1.12 Результати раунду

В момент початку раунду користувач може вийти з додатку зі збереженням кількості балів

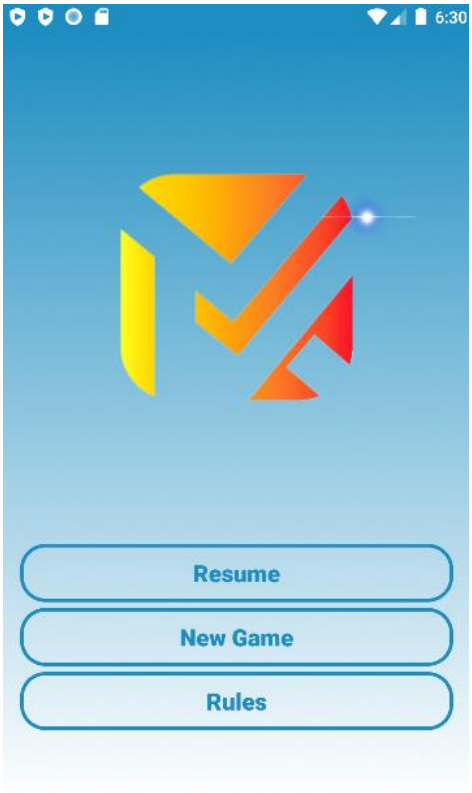

Рис. 3.1.13 Продовження гри

При досягненні вибраної в початку гри кількості балів з’являється відповідне меню зі звуковим ефектом.

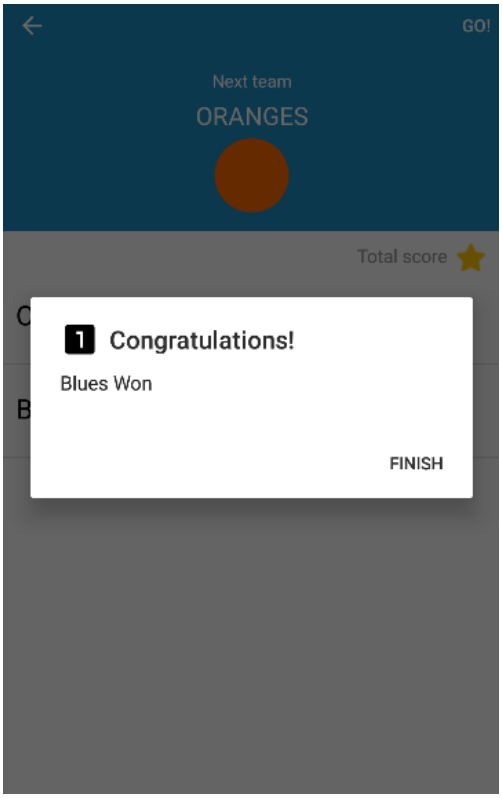

Рис. 3.1.14 Вікно перемоги

Пізніше такий же тест було проведено на власному мобільному пристрої Redmi Note 7, ніяких збоїв не спостерігалось, тестування пройшло успішно.

Висновки до розділу

У третьому розділі був зроблений опис побудови архітектури мобільного застосунку на операційній базі Android.

Також за допомогою додаткових програм для створення та редагування концепт дизайнів, був побудований макет графічного користувацького інтерфейсу.

Виконана робота з програмним кодом за допомогою обраної раніше середи розробки мобільних застосунків на базі операційної системи Android, а також використаний інструментарій, що був проаналізований.

Проведено тестування готового застосунку за допомогою емулятора ОС Android та на власному мобільному пристрої.

ВИСНОВКИ

У результаті виконаної дипломної роботи були досягнуті поставлені цілі та мета. Серед цілей можна виділити проведення аналізу та порівняння різноманітних платформ та операційних систем, та наскільки сьогодні вигідно.

Реалізований зручний мобільний застосунок з функціоналом: створення команд, налаштування параметрів гри, збереження проміжкових результатів. Надалі застосунок буде поповнюватись новим функціоналом для набуватиме все більш зручний та зрозумілий графічний інтерфейс.

Також було проведене тестування мобільного застосунку, як на емуляторі так і на звичайному мобільному пристрої що підтримує операційну систему Android.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка мобільних додатків від А до Я: повний гайд URL: https://dan-it.com.ua/uk/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/
2. Android Studio Guide URL: https://developer.android.com/studio/intro
3. Java tutorial URL: https://www.w3schools.com/java/default.asp
4. Документація по Java URL: https://www.tutorialspoint.com/java/index.htm
5. Розробка дизайну мобільних застосунків. Сім тонкостей, про котрі ви повинні знати URL: https://artjoker.ua/ru/blog/razrabotka-dizayna-mobilnykhprilozheniy-7-tonkostey-o-kotorykh-vy-dolzhny-znat/
6. Роджерс Р., Ломбардо Д. Android. Розробка застосунків, Роджерс Р., Ломбардо Д. – М.: ЕКОМ Паблішерз, 2010. – 400с.

ДОДАТКИ

ДОДАТОК А

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private SharedPreferences sharedPreferences;
    private Button resumeGameBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Objects.requireNonNull(getSupportActionBar()).hide();
        setContentView(R.layout.activity_main);
        sharedPreferences = getSharedPreferences(CURRENT_GAME_DETAILS, MODE_PRIVATE);

        resumeGameBtn = findViewById(R.id.resume_game_btn);
        Button newGameBtn = findViewById(R.id.new_game_btn);
        Button rulesBtn = findViewById(R.id.rules_btn);
        ImageView logoIv = findViewById(R.id.logo_iv);
        ImageView shineIv = findViewById(R.id.shine_iv);

        Animation alphaAndRotateAnim = AnimationUtils.loadAnimation(this, R.anim.logo_anim);
        Animation alphaShineAnim = AnimationUtils.loadAnimation(this, R.anim.shine_anim);
        alphaAndRotateAnim.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) { }
            @Override
            public void onAnimationRepeat(Animation animation) { }

            @Override
            public void onAnimationEnd(Animation animation) {
                MediaPlayer logoMp = MediaPlayer.create(MainActivity.this, R.raw.bit);
                logoMp.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
                    @Override
                    public void onCompletion(MediaPlayer mediaPlayer) {
                        mediaPlayer.release();
                    }
                });
                logoMp.start();
```

```
    }
});
logoIv.startAnimation(alphaAndRotateAnim);
shineIv.startAnimation(alphaShineAnim);

newGameBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        boolean ifNewGame = sharedPreferences.getBoolean(IF_NEW_GAME, true);
        if(!ifNewGame)
        {
            AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setTitle(R.string.sure_fore_new_game).setMessage(R.string.current_will_be_deleted)
                    .setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            SharedPreferences.Editor editor = sharedPreferences.edit();
                            editor.clear();
                            editor.apply();
                            Intent intent = new Intent(MainActivity.this, NewGameActivity.class);
                            startActivity(intent);
                        }
                    }).setNegativeButton(R.string.no,null).show();
        }
        else
        {
            Intent intent = new Intent(MainActivity.this, NewGameActivity.class);
            startActivity(intent);
        }
    }
});
resumeGameBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(MainActivity.this, ScoreManagerActivity.class);
        intent.putExtra(ON_RESUME, true);
        startActivity(intent);
    }
});
rulesBtn.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this, RulesActivity.class);
            startActivity(intent);
        }
    });
}

@Override
protected void onStart() {
    super.onStart();

    boolean ifNewGame = sharedPreferences.getBoolean(IF_NEW_GAME, true);
    if(!ifNewGame)
    {
        resumeGameBtn.setEnabled(true);
        resumeGameBtn.setVisibility(View.VISIBLE);
    }
}


@Override
public void onBackPressed() {
}
}
```

ConstantHolder.java

```
public class ConstantsHolder {

    public static final String CURRENT_GAME_DETAILS = "game_details";
    public static final String IF_NEW_GAME = "if_new_game";
    public static final String TEAMS = "teams_players";
    public static final String WORDS_AMOUNT = "words_amount";
    public static final String TIME_LIMIT = "time_limit";
    public static final String IF_LAST_WORD_FOR_EVERYONE = "last_word_for_all";
    public static final String WORDS_LIST = "words_list";
    public static final String CONST_WORDS_LIST = "const_words_list";
    public static final String CURRENT_TEAM = "curr_team";
    public static final String UNREAD_WORD = "unread_word";
    public static final String GUESSED_WORDS_LIST = "guessed_words_list";
```

```
    public static final String PASSED_WORDS_LIST = "passed_words_list";
    public static final String ROUND_TEAMS_TURN = "turn_of";
    public static final String SCORED_POINTS = "scored_points";
    public static final String START_GAME = "start_game";
    public static final String ON_RESUME = "onresume";
    public static final int MAX_TEAMS_NUMBER = 4;
}
```

GameBoardActivity.java

```
public class GameBoardActivity extends AppCompatActivity {

    private CountDownTimer timer;
    private MediaPlayer ticksMp;
    private MediaPlayer alarmMp;

    private MPCompleteListener mpListener;
    private SharedPreferences sharedPreferences;
    private TextView timerTv;
    private ProgressBar timerProgressBar;
    private TextView mainCardTv;
    private TextView positivePointsTv;
    private TextView negativePointsTv;
    private ImageView lastReversedCardIv;

    private String currTeamName;
    private int secondsPassed;
    private int secondsLimit;
    private ArrayList<String> words;
    private ArrayList<String> guessedWords;
    private ArrayList<String> passedWords;
    private String unreadLastWord;
    private Random random;
    private int screenHeight;

    private FrameLayout.LayoutParams mainCardConstParams;
    private int cardConstLeftMargin;
    private int cardConstTopMargin;

    @SuppressLint("ClickableViewAccessibility")
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_game_board);

TextView guessedTv = findViewById(R.id.guessed_tv);
TextView passTv = findViewById(R.id.pass_tv);
timerTv = findViewById(R.id.timer_service_tv);
timerProgressBar = findViewById(R.id.time_progress_bar);
mainCardTv = findViewById(R.id.main_card_tv);
ImageView pointUpIv = findViewById(R.id.point_plus_iv);
ImageView pointDownIv = findViewById(R.id.minus_point_iv);
positivePointsTv = findViewById(R.id.positive_points_tv);
negativePointsTv = findViewById(R.id.negative_points_tv);
lastReversedCardIv = findViewById(R.id.last_reversed_card_iv);

mpListener = new MPCompleteListener();
ticksMp = MediaPlayer.create(this, R.raw.timerthreesec);
alarmMp = MediaPlayer.create(this, R.raw.alarm);

guessedWords = new ArrayList<>();
passedWords = new ArrayList<>();
random = new Random();

mpListener = new MPCompleteListener();
ticksMp.setOnCompletionListener(mpListener);
alarmMp.setOnCompletionListener(mpListener);

getScreenDimensions();
mainCardConstParams = (FrameLayout.LayoutParams) mainCardTv.getLayoutParams();
cardConstLeftMargin = mainCardConstParams.leftMargin;
cardConstTopMargin = mainCardConstParams.topMargin;
sharedPreferences = getSharedPreferences(CURRENT_GAME_DETAILS, MODE_PRIVATE);
loadData();
setNewCard();

Animation alphaAnim = AnimationUtils.loadAnimation(this, R.anim.game_board_alpha_anim);
guessedTv.startAnimation(alphaAnim);
passTv.startAnimation(alphaAnim);

currTeamName = getIntent().getStringExtra(CURRENT_TEAM);
Objects.requireNonNull(getSupportActionBar()).setTitle(currTeamName);
```

```
        timerTv.setText(secondsLimit + "");
        timer = new CountDownTimer(secondsLimit * 1000L,1000) {

            @Override
            public void onTick(long l) {
                timerTv.setText(l / 1000 + "");
                secondsPassed = secondsLimit - (int)l/1000;
                if(secondsLimit - secondsPassed == 3)
                {
                    ticksMp.start();
                }

                int progress = secondsPassed * 100 / secondsLimit;
                timerProgressBar.setProgress(progress);
            }

            @Override
            public void onFinish() {
                timerTv.setText(0 + "");
                timerProgressBar.setProgress(100);
                unreadLastWord = mainCardTv.getText().toString();
                words.remove(unreadLastWord);
                alarmMp.start();

                final ObjectAnimator invisibleAnim =
ObjectAnimator.ofFloat(mainCardTv,"scaleY",1f,0f).setDuration(1000);
                final ObjectAnimator visibleAnim =
ObjectAnimator.ofFloat(lastReversedCardIv,"scaleY",0f,1f).setDuration(1000);
                invisibleAnim.addListener(new AnimatorListenerAdapter() {
                    @Override
                    public void onAnimationEnd(Animator animation) {
                        super.onAnimationEnd(animation);
                        visibleAnim.start();
                    }
                });
                visibleAnim.addListener(new AnimatorListenerAdapter() {
                    @Override
                    public void onAnimationEnd(Animator animation) {
                        super.onAnimationEnd(animation);
                        saveDataAndGoToNextActivity();
```

```
        }
    });
    invisibleAnim.start();
  }
};

PointBtnClickListener pointBtnClickListener = new PointBtnClickListener();
pointUpIv.setOnClickListener(pointBtnClickListener);
pointDownIv.setOnClickListener(pointBtnClickListener);

mainCardTv.setOnTouchListener(new View.OnTouchListener() {
  FrameLayout.LayoutParams layoutParams;
  int firstTouchX = 0, firstTouchY = 0;
  int firstY = 0;

  @Override
  public boolean onTouch(View view, MotionEvent motionEvent) {
    int currX = (int)motionEvent.getRawX();
    final int currY = (int)motionEvent.getRawY();

    switch (motionEvent.getAction())
    {
      case MotionEvent.ACTION_DOWN:
        layoutParams = (FrameLayout.LayoutParams) view.getLayoutParams();
        firstTouchX = currX - layoutParams.leftMargin;
        firstTouchY = currY - layoutParams.topMargin;
        firstY = currY;
        break;
      case MotionEvent.ACTION_MOVE:
        layoutParams = (FrameLayout.LayoutParams) view.getLayoutParams();
        layoutParams.leftMargin = currX - firstTouchX;
        layoutParams.topMargin = currY - firstTouchY;
        break;
      case MotionEvent.ACTION_UP:
        layoutParams = (FrameLayout.LayoutParams) view.getLayoutParams();
        int YToMove = currY < firstY ? -screenHeight : screenHeight;
        if(YToMove > 0)
        {
          increasePoints(negativePointsTv);
          passedWords.add(mainCardTv.getText().toString());
```

```
                }
                else
                {
                    increasePoints(positivePointsTv);
                    guessedWords.add(mainCardTv.getText().toString());
                }
                animateMainCard(YToMove);
                break;
            }

            view.requestLayout();
            return true;
        }
    });
    timer.start();
}

@Override
public void onBackPressed() {
    showAlertDlg();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {//on android.R.id.home clicked
    showAlertDlg();
    return true;
}

private void showAlertDlg(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(R.string.exit_tomenu_quest).setMessage(R.string.progress_will_be_deleted)
            .setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    timer.cancel();
                    Intent intent = new Intent(GameBoardActivity.this, MainActivity.class);
                    startActivity(intent);
                }
            }).setNegativeButton(R.string.no,null).setIcon(R.drawable.mylogo).show();
}
```

```
private void saveDataAndGoToNextActivity() {
    SharedPreferences.Editor editor = sharedPreferences.edit();
    Gson gson = new Gson();
    String JsonWordsList = gson.toJson(words);
    editor.putString(WORDS_LIST, JsonWordsList);
    editor.apply();

    Intent intent = new Intent(GameBoardActivity.this, RoundResultsActivity.class);
    intent.putExtra(CURRENT_TEAM, currTeamName);
    intent.putStringArrayListExtra(GUESSED_WORDS_LIST, guessedWords);
    intent.putStringArrayListExtra(PASSED_WORDS_LIST, passedWords);
    intent.putExtra(UNREAD_WORD, unreadLastWord);
    startActivity(intent);
}

private void loadData() {
    words = getListFromSP(WORDS_LIST);
    secondsLimit = sharedPreferences.getInt(TIME_LIMIT, 0);
}

private ArrayList<String> getListFromSP(String constStr)
{
    Gson gson = new Gson();
    String JsonWordsList = sharedPreferences.getString(constStr, null);
    Type type = new TypeToken<ArrayList<String>>() {}.getType();

    return gson.fromJson(JsonWordsList, type);
}

private void increasePoints(TextView tv)
{
    if(tv.getId() == R.id.positive_points_tv)
    {
        MediaPlayer pointUpMp = MediaPlayer.create(this, R.raw.pointup);
        pointUpMp.setOnCompletionListener(mpListener);
        pointUpMp.start();
    }
    else if(tv.getId() == R.id.negative_points_tv)
    {
```

```
        MediaPlayer pointDownMp = MediaPlayer.create(this, R.raw.pointdown);
        pointDownMp.setOnCompletionListener(mpListener);
        pointDownMp.start();
    }

    String pointsStr = tv.getText().toString();
    if(pointsStr.equals(""))
    {
        tv.setText("1");
    }
    else
    {
        int points = Integer.parseInt(pointsStr);
        points++;
        tv.setText(points + "");
    }
}

private void setNewCard()
{
    if(words.size() == 1)
    {
        words = getListFromSP(CONST_WORDS_LIST);
    }

    int wordIdx = random.nextInt(words.size());
    mainCardTv.setText(words.get(wordIdx));
    words.remove(wordIdx);

}

private void animateMainCard(int YToMove) {
    mainCardTv.animate().translationY(YToMove).setDuration(400).withEndAction(new Runnable() {
        @Override
        public void run() {
            setNewCard();
            mainCardTv.animate().translationY(0).setDuration(1).withEndAction(new Runnable() {
                @Override
                public void run() {
                    mainCardConstParams.topMargin = cardConstTopMargin;
```

```
                mainCardConstParams.leftMargin = cardConstLeftMargin;
                mainCardTv.setLayoutParams(mainCardConstParams);
            }
        });
    }
}).start();
}

private void getScreenDimensions() {
    Display display = getWindowManager().getDefaultDisplay();
    Point size = new Point();
    display.getSize(size);
    screenHeight = size.y;
}

private class PointBtnClickListener implements View.OnClickListener {

    @Override
    public void onClick(View view) {
        if(view.getId() == R.id.point_plus_iv)
        {
            increasePoints(positivePointsTv);
            guessedWords.add(mainCardTv.getText().toString());
            animateMainCard(-screenHeight);
        }
        else if(view.getId() == R.id.minus_point_iv)
        {
            increasePoints(negativePointsTv);
            passedWords.add(mainCardTv.getText().toString());
            animateMainCard(screenHeight);
        }
    }
}

private static class MPCompleteListener implements MediaPlayer.OnCompletionListener {
    @Override
    public void onCompletion(MediaPlayer mediaPlayer) {
        mediaPlayer.release();
    }
}
```

```
}
```

NewGameActivity.java

```
public class NewGameActivity extends AppCompatActivity implements View.OnClickListener {

    private final int minWords = 10;
    private final int maxWords = 200;
    private final int minTime = 30;
    private final int maxTime = 90;
    private String[] teamNames;
    private int[] teamColors;
    private TeamAdapter teamAdapter;
    private ArrayList<Team> teamsOnTheScreen = new ArrayList<>();
    private List<String> teamNamesNotOnScreen = new ArrayList<>();
    private Button moreTimeBtn;
    private Button lessTimeBtn;
    private Button moreWordsBtn;
    private Button lessWordsBtn;
    private TextView timeLimitTv;
    private TextView wordsAmountTv;
    private CheckBox lastWordCb;
    private Button selectDictionaryBtn;
    private SharedPreferences sharedPreferences;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

Objects.requireNonNull(getSupportActionBar()).setTitle(getResources().getString(R.string.play_new_game));
        setContentView(R.layout.activity_new_game);

        Random random = new Random();
        RecyclerView recyclerView = findViewById(R.id.command_recycler);
        Button addTeamBtn = findViewById(R.id.add_team_btn);
        selectDictionaryBtn = findViewById(R.id.select_dictionary_btn);
        timeLimitTv = findViewById(R.id.seconds_for_round_tv);
        wordsAmountTv = findViewById(R.id.words_amount_tv);
        moreTimeBtn = findViewById(R.id.more_time_btn);
        lessTimeBtn = findViewById(R.id.less_time_btn);
        moreWordsBtn = findViewById(R.id.more_words_btn);
        lessWordsBtn = findViewById(R.id.less_words_btn);
```

```
        lastWordCb = findViewById(R.id.last_word_all_cb);

        teamNames = getResources().getStringArray(R.array.teams_array);
        teamColors = getResources().getIntArray(R.array.color_array);

        moreTimeBtn.setOnClickListener(this);
        lessTimeBtn.setOnClickListener(this);
        moreWordsBtn.setOnClickListener(this);
        lessWordsBtn.setOnClickListener(this);

        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        final int first = random.nextInt(teamNames.length - 1);
        int second = random.nextInt(teamNames.length - 1);
        while ((second == first))
        {
            second = random.nextInt(teamNames.length - 1);/////////////////
        }

        teamsOnTheScreen.add(new Team(teamNames[first], teamColors[first]));
        teamsOnTheScreen.add(new Team(teamNames[second], teamColors[second]));
        teamAdapter = new TeamAdapter(teamsOnTheScreen);
        teamAdapter.setListener(new TeamAdapter.ITeamListener() {
            @Override
            public void onTeamClicked(View view) {
                String text = getResources().getString(R.string.swipe_to_delete);
                Toast.makeText(NewGameActivity.this, text, Toast.LENGTH_SHORT).show();
            }
        });

        //for swiping team item in list of items
        ItemTouchHelper.SimpleCallback callback = new
ItemTouchHelper.SimpleCallback(0,ItemTouchHelper.LEFT|ItemTouchHelper.RIGHT) {
            @Override
            public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
RecyclerView.ViewHolder target) {
                return false;
            }

            @Override
            public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
```

```
            teamsOnTheScreen.remove(viewHolder.getAdapterPosition());
            teamAdapter.notifyItemRemoved(viewHolder.getAdapterPosition());
        }
    };
    ItemTouchHelper itemTouchHelper = new ItemTouchHelper(callback);
    itemTouchHelper.attachToRecyclerView(recyclerView);
    ////
    recyclerView.setAdapter(teamAdapter);

    selectDictionaryBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            final AlertDialog dlg;
            final AlertDialog.Builder builder = new AlertDialog.Builder(NewGameActivity.this);
            LayoutInflater inflater = getLayoutInflater();
            final View dialogView  = inflater.inflate(R.layout.dictionaries_layout,null);
            builder.setView(dialogView);

            final ArrayList<String> dictionariesNamesList = new
ArrayList<>(Arrays.asList(getResources().getStringArray(R.array.categories)));
            final ArrayAdapter<String> dictionariesAdapter =
                    new
ArrayAdapter<String>(NewGameActivity.this,R.layout.custom_text_view,dictionariesNamesList);

            Button enDialogBtn = dialogView.findViewById(R.id.en_btn);
            enDialogBtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    dictionariesNamesList.clear();

dictionariesNamesList.addAll(Arrays.asList(getResources().getStringArray(R.array.categoriesEN)));
                    dictionariesAdapter.notifyDataSetChanged();
                }
            });
            ListView dictionariesListView = dialogView.findViewById(R.id.dictionaries_list);
            dictionariesListView.setAdapter(dictionariesAdapter);
            dlg = builder.create();
            dictionariesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
```

```
                selectDictionaryBtn.setText(dictionariesNamesList.get(i));
                dlg.dismiss();
            }
        });
        dlg.show();
    }
});

addTeamBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(teamsOnTheScreen.size() < MAX_TEAMS_NUMBER)//THINK IF NEED MORE
        {
            teamNamesNotOnScreen.clear();
            for(String teamName : teamNames){
                boolean ifOnScreen = false;
                for(Team team : teamsOnTheScreen)
                {
                    if(teamName.equals(team.getTeamName()))
                    {
                        ifOnScreen = true;
                        break;
                    }
                }

                if(!ifOnScreen)
                {
                    teamNamesNotOnScreen.add(teamName);
                }
            }

            AlertDialog.Builder builder = new AlertDialog.Builder(NewGameActivity.this);
            builder.setItems(teamNamesNotOnScreen.toArray(new String[0]), new
MyItemsDialogListener()).show();
        }
        else
        {
            String text = getResources().getString(R.string.teams_number_limit);
            Toast.makeText(NewGameActivity.this, text, Toast.LENGTH_LONG).show();
        }
```

```
        }
    });
    sharedPreferences = getSharedPreferences(CURRENT_GAME_DETAILS, MODE_PRIVATE);
}

@Override
public void onBackPressed() {
    Intent intent = new Intent(this, MainActivity.class);
    startActivity(intent);
}

private void saveData() {
    int secondsLimit = Integer.parseInt(timeLimitTv.getText().toString());
    int wordsAmount = Integer.parseInt(wordsAmountTv.getText().toString());
    boolean ifWithLastWord = lastWordCb.isChecked();
    String selectedDictionary = selectDictionaryBtn.getText().toString();

    String dictionaryName = selectedDictionary.replaceAll(" ", "");
    int resourceId = this.getResources().getIdentifier(dictionaryName, "array", this.getPackageName());
    String[] words = getResources().getStringArray(resourceId);
    ArrayList<String> wordsList = new ArrayList<String>(Arrays.asList(words));

    SharedPreferences.Editor editor = sharedPreferences.edit();
    Gson gson = new Gson();
    String JsonTeamList = gson.toJson(teamsOnTheScreen);
    editor.putString(TEAMS, JsonTeamList);
    String JsonWordsList = gson.toJson(wordsList);
    editor.putString(WORDS_LIST, JsonWordsList);
    editor.putString(CONST_WORDS_LIST, JsonWordsList);
    editor.putInt(TIME_LIMIT, secondsLimit);
    editor.putInt(WORDS_AMOUNT, wordsAmount);
    editor.putBoolean(IF_LAST_WORD_FOR_EVERYONE, ifWithLastWord);
    editor.putBoolean(IF_NEW_GAME, false);
    editor.apply();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.new_game_menu,menu);
    return super.onCreateOptionsMenu(menu);
```

```
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if(item.getItemId() == R.id.action_start)
        {
            if(selectDictionaryBtn.getText().equals(getResources().getString(R.string.select_dictionary)))
            {
                Toast.makeText(this,                    getResources().getString(R.string.dictionary_not_selected),
Toast.LENGTH_SHORT).show();
            }
            else if(teamsOnTheScreen.size() < 2)
            {
                Toast.makeText(this,                        getResources().getString(R.string.min_two_teams),
Toast.LENGTH_SHORT).show();
            }
            else
            {
                saveData();
                Intent intent = new Intent(this, ScoreManagerActivity.class);
                startActivity(intent);
            }
        }

        return super.onOptionsItemSelected(item);
    }

    private void makeButtonEnabled(Button btn)
    {
        if(!btn.isEnabled())
        {
            btn.setEnabled(true);
            btn.setAlpha(1f);
            btn.setClickable(true);
        }
    }

    private void makeButtonDisabled(Button btn)
    {
        btn.setEnabled(false);
```

```
    btn.setAlpha(.5f);
    btn.setClickable(false);
}

@Override
public void onClick(View view) {
    int viewId = view.getId();
    int seconds;
    int words;
    switch (viewId){
        case R.id.more_time_btn:
            seconds = Integer.parseInt(timeLimitTv.getText().toString());
            if(seconds < maxTime)
            {
                makeButtonEnabled(lessTimeBtn);
                seconds += 30;
                timeLimitTv.setText(seconds + "");
            }
            if(seconds == maxTime)
            {
                makeButtonDisabled(moreTimeBtn);
            }
            break;
        case R.id.less_time_btn:
            seconds = Integer.parseInt(timeLimitTv.getText().toString());
            if(seconds > minTime)
            {
                makeButtonEnabled(moreTimeBtn);
                seconds -= 30;
                timeLimitTv.setText(seconds + "");
            }
            if(seconds == minTime)
            {
                makeButtonDisabled(lessTimeBtn);
            }
            break;
        case R.id.more_words_btn:
            words = Integer.parseInt(wordsAmountTv.getText().toString());
            if (words < maxWords)
            {
```

```
            makeButtonEnabled(lessWordsBtn);
            words += 10;
            wordsAmountTv.setText(words + "");
        }
        if(words == maxWords)
        {
            makeButtonDisabled(moreWordsBtn);
        }
        break;
    case R.id.less_words_btn:
        words = Integer.parseInt(wordsAmountTv.getText().toString());
        if(words > minWords)
        {
            makeButtonEnabled(moreWordsBtn);
            words -= 10;
            wordsAmountTv.setText(words + "");
        }
        if(words == minWords)
        {
            makeButtonDisabled(lessWordsBtn);
        }
        break;
  }
}

private class MyItemsDialogListener implements DialogInterface.OnClickListener {
  @Override
  public void onClick(DialogInterface dialogInterface, int i) {
    String teamNameToAdd = teamNamesNotOnScreen.get(i);
    for(int j = 0; j < teamNames.length; j++)
    {
      if(teamNames[j].equals(teamNameToAdd))
      {
        teamsOnTheScreen.add(new Team(teamNames[j], teamColors[j]));
        teamAdapter.notifyItemInserted(teamsOnTheScreen.size() - 1);
        break;
      }
    }
  }
}
```

```
}
```

NonScrollListView.java

```
public class NonScrollListView extends ListView {

    public NonScrollListView(Context context) {
        super(context);
    }

    public NonScrollListView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public NonScrollListView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    public void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int heightMeasureSpec_custom = MeasureSpec.makeMeasureSpec(
                Integer.MAX_VALUE >> 2, MeasureSpec.AT_MOST);
        super.onMeasure(widthMeasureSpec, heightMeasureSpec_custom);
        ViewGroup.LayoutParams params = getLayoutParams();
        params.height = getMeasuredHeight();
    }
}
```

RoundResultsActivity.java

```
public class RoundResultsActivity extends AppCompatActivity{

    private SharedPreferences sharedPreferences;

    private ArrayList<String> guessedWords;
    private ArrayList<String> passedWords;
    private String unreadLastWord;
    private ArrayList<String> unreadWords = new ArrayList<>();
    private String currTeamName;
    private int score;//score for current team in the round
    private int turnOfTeam;
    private boolean lastWordForEveryone;

    private TextView lastForEveryoneTv;
```

```
    private TextView guessedByTv;
    private TextView scoreTv;
    private ListView guessedWordsLv;
    private ListView passedWordsLv;
    private ListView unreadWordsLv;

    private ArrayAdapter<String> guessedWordsAdapter;
    private ArrayAdapter<String> passedWordsAdapter;
    private ArrayAdapter<String> unreadWordsAdapter;
    private MyOnListItemLongClickListener listItemListener;
    private ArrayList<Team> teamsList;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_round_results);
        Objects.requireNonNull(getSupportActionBar()).setElevation(0);
        sharedPreferences = getSharedPreferences(CURRENT_GAME_DETAILS, MODE_PRIVATE);

        scoreTv = findViewById(R.id.points_in_round_tv);
        guessedWordsLv = findViewById(R.id.guessed_words_lv);
        passedWordsLv = findViewById(R.id.passed_words_lv);
        unreadWordsLv = findViewById(R.id.unread_words_lv);
        lastForEveryoneTv = findViewById(R.id.word_for_everyone_tv);
        guessedByTv = findViewById(R.id.last_guessed_by_tv);

        loadData();
        calcPoints();
        guessedWordsAdapter = new
ArrayAdapter<String>(this,R.layout.custom_text_view_for_gamewords,guessedWords);
        passedWordsAdapter = new
ArrayAdapter<String>(this,R.layout.custom_text_view_for_gamewords,passedWords);
        unreadWordsAdapter = new ArrayAdapter<String>(this,R.layout.custom_text_view_for_gamewords,
unreadWords);
        guessedWordsLv.setAdapter(guessedWordsAdapter);
        passedWordsLv.setAdapter(passedWordsAdapter);
        unreadWordsLv.setAdapter(unreadWordsAdapter);

        listItemListener = new MyOnListItemLongClickListener();///if not long click show toast
        guessedWordsLv.setOnItemLongClickListener(listItemListener);
```

```
passedWordsLv.setOnItemLongClickListener(listItemListener);
unreadWordsLv.setOnItemLongClickListener(listItemListener);

if(lastWordForEveryone)
{
   String[] teamNames = new String[teamsList.size()];
   for(int i = 0; i < teamNames.length; i++)
   {
      teamNames[i] = teamsList.get(i).getTeamName();
   }

   String msg = getResources().getString(R.string.last_word_guessed) + " " + unreadLastWord + ". "
         + getResources().getString(R.string.guessed_by) + ":";

   AlertDialog.Builder builder = new AlertDialog.Builder(this);
   builder.setTitle(msg).setItems(teamNames, new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialogInterface, int i) {
         lastForEveryoneTv.setText(unreadLastWord);
         String guessedMsg = getResources().getString(R.string.guessed_by) + " " +
teamsList.get(i).getTeamName();
         guessedByTv.setText(guessedMsg);
         unreadWords.remove(unreadLastWord);
         unreadWordsAdapter.notifyDataSetChanged();
         int score = teamsList.get(i).getTotalScore();
         score++;
         teamsList.get(i).setTotalScore(score);

         if(teamsList.get(i).getTeamName().equals(currTeamName))
         {
            int currScore = Integer.parseInt(scoreTv.getText().toString());
            currScore++;
            scoreTv.setText(currScore + "");
         }

         String msg = getResources().getString(R.string.plus_one_point_to) + " " +
teamsList.get(i).getTeamName();
         Toast.makeText(RoundResultsActivity.this, msg, Toast.LENGTH_SHORT).show();
      }
   }).setNegativeButton(R.string.nobody, null).setIcon(R.drawable.mylogo).show();
```

```
    }
}

private void loadData()
{
    Gson gson = new Gson();
    String JsonTeamsList = sharedPreferences.getString(TEAMS, null);
    Type type = new TypeToken<ArrayList<Team>>() {}.getType();
    teamsList = gson.fromJson(JsonTeamsList, type);

    turnOfTeam = sharedPreferences.getInt(ROUND_TEAMS_TURN, 0);
    currTeamName = getIntent().getStringExtra(CURRENT_TEAM);
    lastWordForEveryone = sharedPreferences.getBoolean(IF_LAST_WORD_FOR_EVERYONE, false);

    Objects.requireNonNull(getSupportActionBar()).setTitle(currTeamName);
    guessedWords = getIntent().getStringArrayListExtra(GUESSED_WORDS_LIST);
    passedWords = getIntent().getStringArrayListExtra(PASSED_WORDS_LIST);
    unreadLastWord = getIntent().getStringExtra(UNREAD_WORD);
    unreadWords.add(unreadLastWord);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(item.getItemId() == R.id.action_apply)
    {
        SharedPreferences.Editor editor = sharedPreferences.edit();
        Gson gson = new Gson();
        String JsonTeamList = gson.toJson(teamsList);
        editor.putString(TEAMS, JsonTeamList);
        editor.putBoolean(START_GAME, false);

        //team turn managing for score manager activity
        if(turnOfTeam == teamsList.size() - 1)
        {
            turnOfTeam = 0;
        }
        else
        {
            turnOfTeam++;
        }
```

```
        editor.putInt(ROUND_TEAMS_TURN, turnOfTeam);
        editor.commit();

        Intent intent = new Intent(this, ScoreManagerActivity.class);
        intent.putExtra(SCORED_POINTS, score);
        startActivity(intent);
    }
    else if(item.getItemId() == android.R.id.home)
    {
        showAlertDlg();
        return true;
    }

    return super.onOptionsItemSelected(item);
}

private void calcPoints() {
    score = guessedWords.size() - passedWords.size();
    scoreTv.setText(score + "");
}

@Override
public void onBackPressed() {
    showAlertDlg();
}

private void showAlertDlg(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(R.string.exit_tomenu_quest).setMessage(R.string.progress_will_be_deleted)
            .setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    Intent intent = new Intent(RoundResultsActivity.this, MainActivity.class);
                    startActivity(intent);
                }
            }).setNegativeButton(R.string.no,null).setIcon(R.drawable.mylogo).show();
}

@Override
```

```
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.round_results_menu,menu);
        return super.onCreateOptionsMenu(menu);
    }

    private class MyOnListItemLongClickListener implements AdapterView.OnItemLongClickListener {
        @Override
        public boolean onItemLongClick(AdapterView<?> adapterView, View view, final int i, long l) {
            Context wrapper = new ContextThemeWrapper(RoundResultsActivity.this, R.style.PopupMenu);
            final PopupMenu popupMenu = new PopupMenu(wrapper, view);
            int menuId = 0;
            switch (adapterView.getId())
            {
                case R.id.guessed_words_lv:
                    menuId = R.menu.guessed_word_menu;
                    break;
                case R.id.passed_words_lv:
                    menuId = R.menu.passed_word_menu;
                    break;
                case R.id.unread_words_lv:
                    menuId = R.menu.unread_word_menu;
                    break;
            }
            popupMenu.getMenuInflater().inflate(menuId, popupMenu.getMenu());
            popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                @Override
                public boolean onMenuItemClick(MenuItem item) {
                    switch (item.getItemId())
                    {
                        case R.id.from_guessed_to_passed:
                            updateDataInListViews(guessedWords, passedWords, i, passedWordsAdapter,
guessedWordsAdapter);
                            break;
                        case R.id.from_guessed_to_unread:
                            updateDataInListViews(guessedWords, unreadWords, i, unreadWordsAdapter,
guessedWordsAdapter);
                            break;
                        case R.id.from_passed_to_guessed:
                            updateDataInListViews(passedWords, guessedWords, i, passedWordsAdapter,
guessedWordsAdapter);
```

```
                    break;
                case R.id.from_passed_to_unread:
                    updateDataInListViews(passedWords, unreadWords, i, passedWordsAdapter,
unreadWordsAdapter);
                    break;
                case R.id.from_unread_to_guessed:
                    updateDataInListViews(unreadWords, guessedWords, i, guessedWordsAdapter,
unreadWordsAdapter);
                    break;
                case R.id.from_unread_to_passed:
                    updateDataInListViews(unreadWords, passedWords, i, passedWordsAdapter,
unreadWordsAdapter);
                    break;
            }

            calcPoints();

            return false;
        }
      });
      popupMenu.show();

      return false;
    }

    private void updateDataInListViews(ArrayList<String> listFrom, ArrayList<String> listTo, int itemIdx,
                                       ArrayAdapter<String> firstAdapter, ArrayAdapter<String> secondAdapter){
      listTo.add(listFrom.get(itemIdx));
      listFrom.remove(itemIdx);
      firstAdapter.notifyDataSetChanged();
      secondAdapter.notifyDataSetChanged();
    }
  }
}
```

RulesActivity.java

```
public class RulesActivity extends AppCompatActivity {
  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Objects.requireNonNull(getSupportActionBar()).setTitle(getResources().getString(R.string.game_rules));
```

```
        setContentView(R.layout.activity_rules);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.rules_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if(item.getItemId() == R.id.action_share)
        {
            Intent intent = new Intent(Intent.ACTION_SEND);
            intent.putExtra(Intent.EXTRA_TEXT, getResources().getString(R.string.sharing_txt));
            intent.setType("text/html");
            startActivity(Intent.createChooser(intent,""));
        }

        return super.onOptionsItemSelected(item);
    }
}
```

ScoreManagerActivity.java

```
public class ScoreManagerActivity extends AppCompatActivity {

    private int wordsAmount;
    private int turnOfNextTeam;
    private int playedTeamScore;
    private ArrayList<Team> teamsList;
    private SharedPreferences sharedPreferences;

    private ListView teamsListView;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_score_manager);
        Objects.requireNonNull(getSupportActionBar()).setDisplayShowTitleEnabled(false);
        getSupportActionBar().setElevation(0);
```

```
        TextView nextPlayerTeamTv = findViewById(R.id.next_player_team_tv);
        TextView getNextTeamColorTv = findViewById(R.id.team_color_for_score_tv);
        teamsListView = findViewById(R.id.teams_score_list);

        sharedPreferences = getSharedPreferences(CURRENT_GAME_DETAILS, MODE_PRIVATE);
        loadData();
        boolean resumed = getIntent().getBooleanExtra(ON_RESUME, false);
        if(!resumed)
        {
            manageScore();
            getIntent().putExtra(ON_RESUME, false);
        }

        Team nextPlayerTeam = teamsList.get(turnOfNextTeam);
        nextPlayerTeamTv.setText(nextPlayerTeam.getTeamName());
        getNextTeamColorTv.getBackground().setColorFilter(nextPlayerTeam.getTeamColor(),
PorterDuff.Mode.SRC_ATOP);
        fillListView();

        int[] colors = new int[]{
                ContextCompat.getColor(this,R.color.seaStar),
                ContextCompat.getColor(this,R.color.darkGoldStar),
                ContextCompat.getColor(this,R.color.goldStar)
        };

        ValueAnimator valueAnim = ValueAnimator.ofObject(new ArgbEvaluator(),colors[0],colors[1],colors[2]);
        valueAnim.setDuration(4000);

        valueAnim.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
            @Override
            public void onAnimationUpdate(ValueAnimator valueAnimator) {
                ImageView image = findViewById(R.id.star_iv);
                int color = (int)valueAnimator.getAnimatedValue();
                image.setColorFilter(color);
            }
        });
        valueAnim.start();
    }

    private void fillListView() {
```

```
    //map for each list item
    List<Map<String,Object>> teamsData = new ArrayList<>();
    String playedRes = getResources().getString(R.string.played_in_round);
    String teamConstStr = "team";//key for team name
    String playedConstStr = "played";//key for string that will include "played in round" or "" - second key
    String scoreConsStr = "score";//key for team score in the round

    int teamIdx = 0;
    for(Team team : teamsList)
    {
        HashMap<String, Object> map = new HashMap<>();
        //checks if the team played in the round already
        String ifPlayed = teamIdx < turnOfNextTeam ? playedRes : "";//second value in the map
        map.put(teamConstStr, team.getTeamName());
        map.put(playedConstStr, ifPlayed);
        map.put(scoreConsStr, team.getTotalScore());
        teamsData.add(map);
        teamIdx++;
    }

    String[] from = {teamConstStr, playedConstStr, scoreConsStr};
    int[] to = {R.id.team_name_score_tv, R.id.played_tv, R.id.score_tv};
    SimpleAdapter adapter = new SimpleAdapter(this, teamsData, R.layout.score_team_layout, from, to);
    teamsListView.setAdapter(adapter);
}

@Override
public void onBackPressed() {
    saveData();
    Intent intent = new Intent(this, MainActivity.class);
    startActivity(intent);
}

private void loadData() {
    Gson gson = new Gson();
    String JsonTeamsList = sharedPreferences.getString(TEAMS, null);
    Type type = new TypeToken<ArrayList<Team>>() {}.getType();
    teamsList = gson.fromJson(JsonTeamsList, type);
    playedTeamScore = getIntent().getIntExtra(SCORED_POINTS, 0);
    wordsAmount = sharedPreferences.getInt(WORDS_AMOUNT, 0);////////
```

```
    turnOfNextTeam = sharedPreferences.getInt(ROUND_TEAMS_TURN, 0);
}

private void manageScore(){
    boolean startGame = sharedPreferences.getBoolean(START_GAME, true);
    //if this is not starting point of the game(not new game)
    if(!startGame)
    {
        boolean finished = false;
        //idx of the team that played right now
        int playedTeamIdx = turnOfNextTeam == 0 ? (teamsList.size() - 1) : turnOfNextTeam - 1;

        if(playedTeamScore != 0)
        {
            int totalTeamScore = teamsList.get(playedTeamIdx).getTotalScore();
            totalTeamScore += playedTeamScore;
            teamsList.get(playedTeamIdx).setTotalScore(totalTeamScore);
        }

        if(playedTeamIdx == teamsList.size() - 1)
        {
            finished = checkIfGotToFinish();
        }

        if(finished)
        {
            findWinners();
        }
    }
}

private boolean checkIfGotToFinish(){
    boolean finished = false;

    for (Team team : teamsList)
    {
        if(team.getTotalScore() >= wordsAmount)
        {
            finished = true;
            break;
```

```
        }
    }

    return finished;
}

 @Override
public boolean onOptionsItemSelected(MenuItem item) {

    saveData();
    if(item.getItemId() == R.id.action_go)
    {
        Intent intent = new Intent(this, GameBoardActivity.class);
        intent.putExtra(CURRENT_TEAM, teamsList.get(turnOfNextTeam).getTeamName());
        startActivity(intent);
    }

    return super.onOptionsItemSelected(item);
}

private void findWinners() {
    ArrayList<Team> winnersList = new ArrayList<>();

    for (Team team : teamsList)
    {
        if (team.getTotalScore() >= wordsAmount)
        {
            winnersList.add(team);
        }
    }

    if(winnersList.size() > 1)
    {
        winnersList.sort(new Comparator<Team>() {
            @Override
            public int compare(Team t1, Team t2) {
                int returningValue = 0;

                if(t1.getTotalScore() > t2.getTotalScore())
                {
```

```
            returningValue = -1;
        }
        else if(t1.getTotalScore() < t2.getTotalScore())
        {
            returningValue = 1;
        }

        return returningValue;
    }
  });
}

Team bestResult = winnersList.get(0);
if(winnersList.size() == 1)
{
  showWinnerAndFinishGame(bestResult);

}
else if(bestResult.getTotalScore() > winnersList.get(1).getTotalScore())
{
  showWinnerAndFinishGame(bestResult);
}
else
{
  ArrayList<Team> bestResultList = new ArrayList<>();
  for (Team team : winnersList)
  {
    if(team.getTotalScore() == bestResult.getTotalScore())
    {
      bestResultList.add(team);
    }
    else
    {
      break;
    }
  }

  teamsList = bestResultList;
  saveData();
}
```

```
    }

    private void showWinnerAndFinishGame(Team bestResult)
    {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        String msg = bestResult.getTeamName() + " " + getResources().getString(R.string.won);
        builder.setTitle(R.string.congratulations).setMessage(msg).setPositiveButton(R.string.finish, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.clear();
                editor.apply();
                Intent intent = new Intent(ScoreManagerActivity.this, MainActivity.class);
                startActivity(intent);
            }
        }).setCancelable(false).setIcon(R.drawable.ic_looks_one_black_24dp).show();

        final MediaPlayer applaudsMp = MediaPlayer.create(this, R.raw.winner);
        applaudsMp.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mediaPlayer) {
                applaudsMp.release();
            }
        });
        applaudsMp.start();
    }

    private void saveData(){
        SharedPreferences.Editor editor = sharedPreferences.edit();
        Gson gson = new Gson();
        String JsonTeamList = gson.toJson(teamsList);
        editor.putString(TEAMS, JsonTeamList);
        editor.putInt(ROUND_TEAMS_TURN, turnOfNextTeam);
        editor.apply();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.score_manager_menu,menu);
```

```
        return super.onCreateOptionsMenu(menu);
    }
}
```

Team.java

```
public class Team {
    private String teamName;
    private int teamColor;
    private int totalScore;

    public Team(String teamName, int teamColor) {
        this.teamName = teamName;
        this.teamColor = teamColor;
        this.totalScore = 0;
    }

    public int getTotalScore() {
        return totalScore;
    }

    public void setTotalScore(int totalScore) {
        this.totalScore = totalScore;
    }

    public String getTeamName() {
        return teamName;
    }

    public void setTeamName(String teamName) {
        this.teamName = teamName;
    }

    public int getTeamColor() {
        return teamColor;
    }

    public void setTeamColor(int teamColor) {
        this.teamColor = teamColor;
    }
}
```

TeamAdapter.java

```
public class TeamAdapter extends RecyclerView.Adapter<TeamAdapter.TeamViewHolder>{

    private List<Team> teamsList;
    private ITeamListener listener;

    public TeamAdapter(List<Team> teamsList) {
        this.teamsList = teamsList;
    }

    interface ITeamListener {
        void onTeamClicked(View view);
    }

    public void setListener(ITeamListener listener)
    {
        this.listener = listener;
    }

    public class TeamViewHolder extends RecyclerView.ViewHolder{
        TextView teamColorTextView;
        TextView teamNameTextView;


        public TeamViewHolder(View itemView) {
            super(itemView);

            teamColorTextView = itemView.findViewById(R.id.team_color_tv);
            teamNameTextView = itemView.findViewById(R.id.team_name_tv);

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if(listener != null)
                    {
                        listener.onTeamClicked(view);
                    }
                }
            });
        }
    }
```

```
    @Override
    public TeamViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.team_layout,parent,false);
        TeamViewHolder teamViewHolder = new TeamViewHolder(view);

        return teamViewHolder;
    }

    @Override
    public void onBindViewHolder(TeamViewHolder holder, int position) {
        Team team = teamsList.get(position);
        holder.teamColorTextView.getBackground().setColorFilter(team.getTeamColor(),
PorterDuff.Mode.SRC_ATOP);//CHECK
        holder.teamNameTextView.setText(team.getTeamName());
    }

    @Override
    public int getItemCount() {
        return teamsList.size();
    }
}
```