Katedra Informatyki
i Aparatury Medycznej

Wydział Inżynierii Biomedycznej

POLITECHNIKA ŚLĄSKA

PROJEKT INŻYNIERSKI

Aplikacja do testowania pamięci krótkotrwałej

Kamil OLESZ
Kierunek studiów: Inżynieria Biomedyczna
Specjalność: Informatyka i Aparatura Medyczna

KIERUJĄCY PRACĄ
dr inż. Jacek Kawa

Spis treści

1. Wstęp

W dzisiejszych czasach nauka rozwija się w zawrotnym tempie. Mimo tego, sposób działania ludzkiego mózgu nie jest do końca poznany [3]. Jedną z fundamentalnych umiejętności mózgu, bez której człowiek nie mógłby funkcjonować, jest pamięć. Odpowiada ona za wszystkie zdolności, których nauczył się przez całe swoje życie. Bez pamięci, wszystkie wrażenia, które ludzie doznają, zacierałyby się i to, co człowiek odczuł, po chwili stałoby się dla niego nieistniejące [3, 4].

Za zdolność zapamiętywania odpowiedzialne są neurony, a dokładnie ich całe sieci połączeń. Przy doświadczeniu jakiegokolwiek bodźca, komórki nerwowe łączą się ze sobą. Takie połączenie zwane jest synapsą – może być mocniejsze lub słabsze, co wpływa na późniejsze utrzymanie tegoż wiązania. Po każdorazowym odebraniu tego samego wrażenia, synapsa wzmacnia się i pozostaje nieprzerwana. Przy jednorazowym, krótkim odebraniu bodźca, synapsa szybko zanika gdyż połączenie między komórkami jest zbyt słabe [5, 8]. Mózg sam potrafi ustalić ważność pobudzenia, zatem wybiórczo pozostawia rzeczy w swojej pamięci. Tak dla przykładu, pierwsze poparzenie się ogniem „uczy" człowieka, by ten starał się unikać kolejnych poparzeń. Wrażenie bólu jest bowiem ważnym aspektem dla mózgu. Z drugiej strony, mało ważną informacją dla człowieka jest wygląd mijającej go na chodniku osoby. Mózg przyswaja ten wygląd, lecz po chwili może dojść do wniosku, że nie jest to istotna informacja i połączenie neuronów się zerwie. Wszystkie operacje wiązań dzieją się w strukturze mózgu zwanej hipokampem. [1, 2].

Najbardziej pospolitym podziałem pamięci jest podział ze względu na czas trwania pamiętania. Wyróżniamy pamięć: ultrakrótkotrwałą (zwana również sensoryczną), krótkotrwałą oraz długotrwałą. W dalszej części pracy uwaga zostanie skupiona tylko pamięci krótkotrwałej [4].

Pamięć krótkotrwała jest wykorzystywana przez ludzi codziennie, przy zwykłych domowych czynnościach, do których nie przywiązuje się zbyt dużej wagi, np. przy zapamiętaniu czy herbata została już posłodzona albo drzwi wejściowe zostały zamknięte.Umiejętność zapamiętywania krótkotrwałego warto jest ćwiczyć, by sprawniej wykonywać codzienne czynności (również te w pracy) oraz lepiej przyswajać informacje otaczającego świata. Trening pamięci krótkotrwałej ma również wpływ na zdolność pamiętania długoterminowego: więcej neuronów jest wykorzystywanych w obrębie ośrodka pamięci, przez co człowiek jest w stanie zapamiętać więcej rzeczy. Pamiętając pewne informacje dłużej, łatwiej jest przenieść je z obszaru pamięci krótkotrwałej na długotrwałą [3–5].

Istnieje wiele sposobów testowania ludzkiej pamięci, a najwięcej odnosi się do badania pamięci krótkotrwałej. Przeprowadzenie takiego testu wiąże się z prostym zadaniem

zapamiętania pewnych rzeczy, np. figur geometrycznych, par wyrazów, krótkich słów czy ciągów cyfr [6].

Trening pamięci polega na regularnym powtarzaniu danych testów. W danym okresie czasu warto skupić się przy tym na pojedynczym zadaniu, gdyż powtarzane czynności są bardziej przyswajane przez mózg. Przykładowo: powtarzanie ciągu cyfr jest uniwersalne dla każdej osoby, w końcu każdy zna cyfry, a bardzo łatwo przy tym można określić jak długi ciąg zapamiętuje dana osoba [4, 6].

Jest mnóstwo form trenowania (przy tym i testowania) pamięci [6]. Jedną z najnowszych, ale i bardzo efektywnych są wszelkie aplikacje mobilne. Na sam trening pamięci konieczna jest tylko chwila czasu, ale najważniejsza jest regularność, bowiem efekty widoczne są dopiero po długim okresie. Łatwo jest włączyć aplikację mobilną, korzystać z niej przez określony czas po czym bez żadnego opóźnienia wrócić do codziennych obowiązków. Treningi takie można wykonywać w każdych wolnych chwilach, chociażby przy podróży autobusem. To daje ogromną wygodę i przez to, coraz więcej ludzi decyduje się na trenowanie swojej pamięci – dlatego też występuje popyt na tego rodzaju oprogramowania.

Dostępne na rynku aplikacje [1] [2] [3] oferują przerost formy nad treścią. Twórcy skupiają uwagę na przyciągnięciu potencjalnych klientów swoim wyglądem. Sama forma aplikacji może w znaczący sposób dekoncentrować trenującego (np. zmiana kolorów ekranu, zbyt długie animacje czy migające obrazki). Każdy oczekuje od aplikacji czegoś innego, tak więc przydatna byłaby możliwość personalizacji, lub ewentualnie możliwość łatwej edycji programu i sprzedaż spersonalizowanych form.

Celem niniejszego projektu jest zaprojektowanie i stworzenie aplikacji mobilnej, umożliwiającej test pamięci krótkotrwałej. Forma testu - powtarzanie wydłużającego się z każdą próbą ciągu cyfr - pozwoli na jednoczesny trening pamięci i samodoskonalenie. Planowana grupa użytkowników docelowych obejmuje

- osoby zdrowe, nie trenujące dotychczas pamięci,
- osoby zdrowe, trenujące regularnie pamięć krótkotrwałą,
- osoby chore na wszelkie ubytki pamięci – korzystanie w celach diagnostycznych i terapeutycznych,
- osoby młodsze – forma na tyle prosta by mogła korzystać z niej jak najmłodsza osoba,
- osoby starsze – podobnie jak dla osób młodszych, forma na tyle prosta by bez problemów mogły korzystać z aplikacji starsze osoby.

Realizacja celu wymagać będzie następujących kroków: projekt, implementacja i testowanie. Zostaną one opisane w dalszej części pracy.

[1] Brainwell — Brain Training Games By Monclarity, LLC – App Store, 17.12.2017
[2] Lumosity – Trening mózgu, Lumos Laps, Inc. – Google Play, 17.12.2017
[3] Gry na pamięć: Trening mózgu, Maple Media – Google Play, 17.12.2017

2. Projekt aplikacji

W niniejszym rozdziale przedstawiony zostanie projekt i specyfikacja wewnętrzna aplikacji – testu pamięci krótkotrwałej

2.1 Założenia wstępne

Zgodnie z założeniami początkowymi, opracowana aplikacja mobilna powinna umożliwić testowanie pamięci krótkotrwałej poprzez zadanie polegające na zapamiętaniu ciągu cyfr.

Dodatkowe założenia to:

- początkowa długość ciągu cyfr powinna być możliwa do ustawienia przez użytkownika,
- długość ciągu nie powinna być ograniczona,
- w przypadku pomyłki powinna istnieć możliwość powtórnego wykonania zadania.

Przy takich założeniach możliwy będzie również trening pamięci: poziom trudności będzie rosnąć wraz z umiejętnościami użytkownika. Powinny pojawiać się komunikaty informujące użytkownika o poprawnym lub błędnym zapamiętaniu wygenerowanej sekwencji.

2.1.1 Funkcje opcjonalne

W ramach analizy projektowej rozważono również dwie funkcje dodatkowe:

- wprowadzanie ciągu głosem,
- zapamiętywanie wyników.

Wprowadzanie ciągu głosem ułatwiłoby zadanie osobom starszym, dla których problemem byłoby sprawne wpisywanie z klawiatury numerycznej. Kolejnym plusem takiego rozwiązania, byłoby umożliwienie testowania pamięci krótkotrwałej osobom z dysfunkcjami ruchowymi, np. po udarze. Opcja wprowadzania głosowego byłaby również formą personalizacji. Dla pewnych osób wygodniejsze mogłoby być wypowiedzenie zapamiętanego ciągu niż wpisanie go.

Zapamiętywanie wyników nie jest istotne dla samego testu, ale może stanowić czynnik motywujący dla osób trenujących pamięć. Rozważano możliwość zapisu wyników w bazie danych lub w pliku tekstowym. Pierwszą z możliwości uznano za mniej korzystną ze względu na utrudniony eksport wyników.

2.2 Analiza założeń

Na podstawie założeń przyjęto, że w ramach pracy zostanie wykonana aplikacja dla systemu Android, czyli najbardziej popularnego systemu wśród obywateli Polski[1]. Jako środowisko programistyczne wybrano Android Studio i język Java.

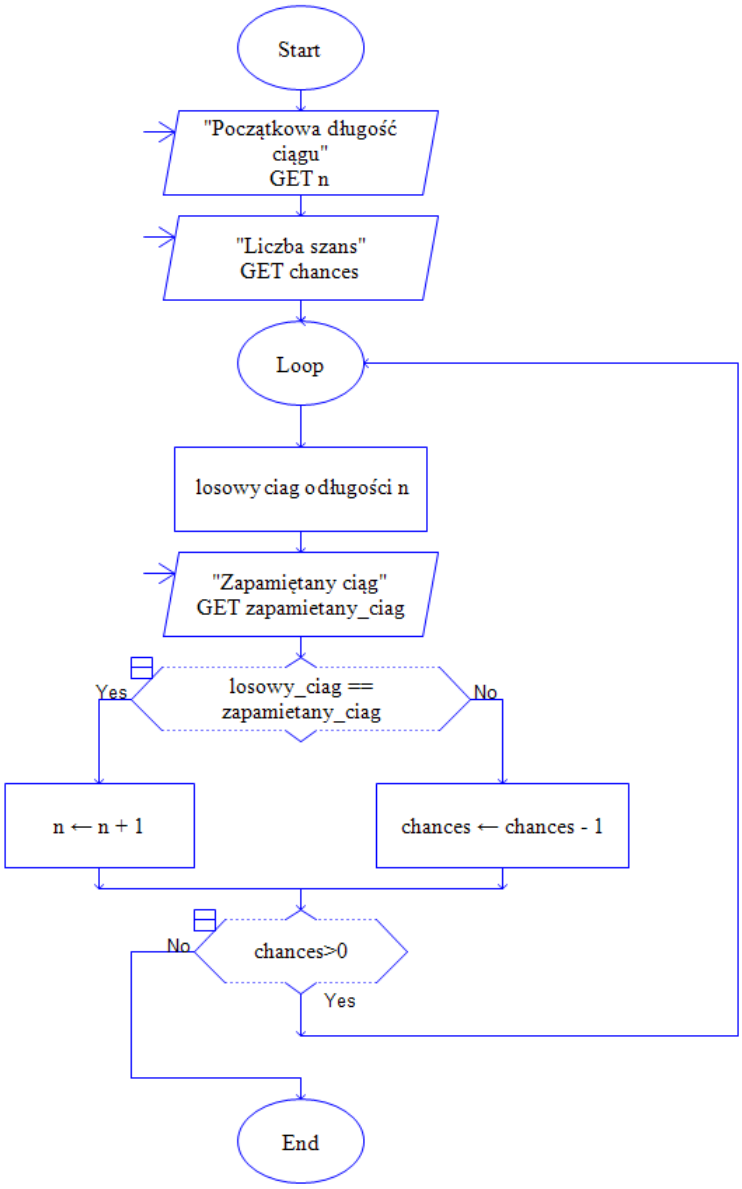
Aplikacja będzie posiadała ustawienia, umożliwiające wybranie początkowej długości ciągu, zadeklarowanie liczby szans – warunku kończącego rozgrywkę oraz wybór trybu wpisywania głosowego (zamień tekst lub dodaj do tekstu). Kolejnym aspektem będzie zapisywanie wyniku najdłuższego zapamiętanego ciągu cyfr w danej rozgrywce wraz z zapisem czasu zapamiętywania tej sekwencji.

Dla przejrzystości rozgrywki, program będzie posiadał samouczek, dostępny do uruchomienia z pozycji menu głównego. Wprowadzi on w panujące w trakcie testu zasady – co należy zrobić w danym etapie rozgrywki. Zaznajomi również użytkownika z funkcją wprowadzania głosowego.

Test pamięci będzie przebiegał według następującego schematu (Rys. 2.1):

1. Rozpoczęcie testu po uprzednim ustawieniu początkowej długości ciągu.
2. Wygenerowanie losowego ciągu cyfr od 0 do 9.
3. Zapamiętanie wyświetlonego ciągu przez użytkownika.
4. Wprowadzenie zapamiętanej sekwencji.
5. Sprawdzenie poprawności zapamiętania.
6. Jeżeli warunek końca testu nie został spełniony - przejście do punktu drugiego.
7. Jeżeli warunek został spełniony - zakończenie rozgrywki.

[1] Według statcounter GlobalStats, 18.12.2017



Rys. 2.1: Schemat blokowy testu pamięci.

2.3 Specyfikacja wewnętrzna

Aplikacja została napisana w środowisku Android Studio [7]. Klasy są napisane w języku Java a pliki definiujące interfejs mają rozszerzenie xml. Minimalna wspierana wersja API to 16, co odpowiada wersji 4.1 Jelly Bean systemu operacyjnego Android. Według Android Studio, daje to możliwość uruchomienia stworzonej aplikacji na, w przybliżeniu, 99.2% urządzeń aktywnych na Google Play Store.

Aplikacja korzysta z zewnętrznej biblioteki *opencsv*, pozwalającej na zapis i odczyt danych do/z plików o rozszerzeniu CSV (comma-separated values). Reszta wymaganych bibliotek dostarcza samo środowisko. Biblioteka *opencsv* jest biblioteką typ open source, możliwą do pobrania na stronie opencsv.sourceforge.net. Przy programowaniu zostały użyte następujące klasy i metody tej biblioteki:

- *CSVWriter* - klasa umożliwiająca zapisanie danych tablicowych do pliku typu CSV. Jako argumenty obiektu tej klasy podawany jest obiekt klasy *FileWriter* (przyjmuje ścieżkę oraz nazwę pliku, służy do ogólnego zapisywania plików) oraz znak separujący. Wykorzystywane metody to:

  1. Metoda *writeNext*, zapisująca kolejny wers do pliku. Wers jest podawany w formie tablicy z danymi typu String.
  2. Metoda *close*, zapisująca i zamykająca utworzony lub modyfikowany plik.

- *CSVReader* - klasa umożliwiająca odczyt z pliku CSV. Jako argumenty obiektu tej klasy podawany jest obiekt klasy *FileWriter* oraz znak separujący. Wykorzystywane metody to:

  1. Metoda *readAll*, odczytująca wszystkie wersy z danego pliku CSV. Zapisuje każdą komórkę do odpowiedniego miejsca w liście typu String tablicowy.
  2. Metoda *close*, zamykająca otwarty wcześniej plik CSV.

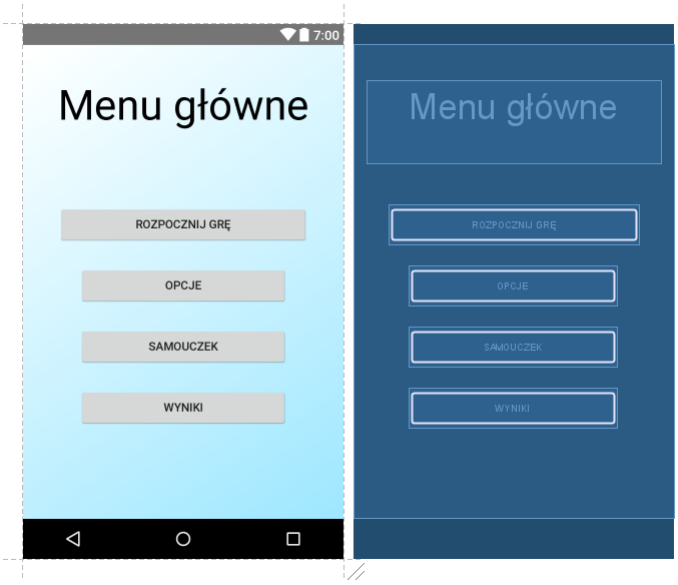
2.3.1 Interfejs użytkownika

Interfejs graficzny użytkownika tworzą następujące pliki:

- activity_menu.xml,
- activity_numbers.xml,
- activity_options.xml,
- activity_results.xml,
- list_row.xml,
- frame.xml,
- gradient.xml.

2.3.1.1 Widok activity_menu

Jest to widok ukazywany od razu przy uruchomieniu aplikacji (Rys. 2.2). Składa się z czterech przycisków i jednego pola tekstowego. Każdy przycisk odpowiedzialny jest za przejście do innej aktywności, ukazanej w nazwie przycisku. Pole tekstowe informuje użytkownika, że znajduje się w menu głównym.



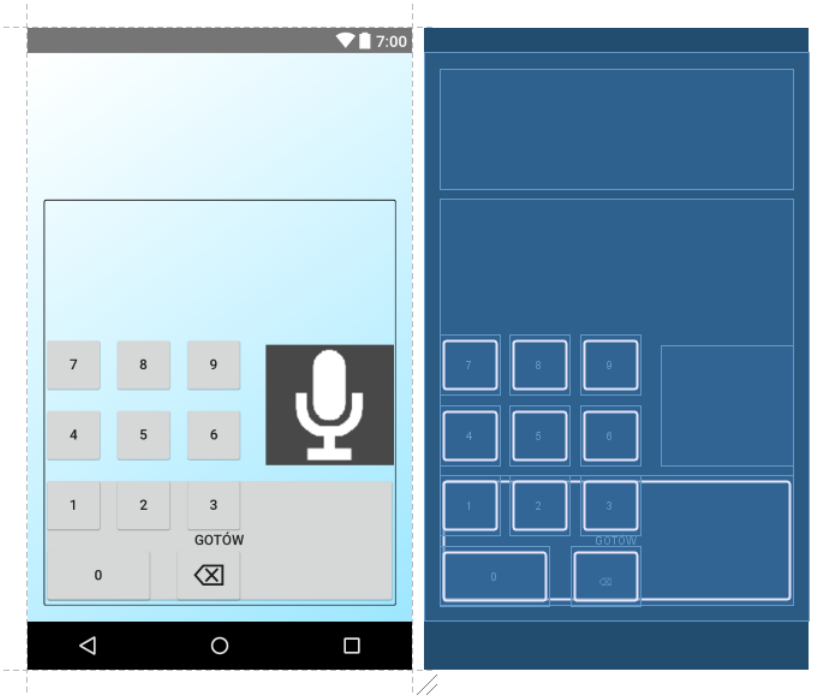
Rys. 2.2: Projekt wyglądu menu głównego aplikacji.

2.3.1.2 Widok activity_numbers

Widok pokazujący wygląd całego testu pamięci (Rys. 2.3). Zawiera 10 przycisków umożliwiających wpisywanie cyfr. Są ułożone w wygodny do użytkowania sposób, przypominający klawiaturę telefonu. Przyciski te są oznaczone odpowiednim symbolem, przez co użytkownik wie, jaka cyfra pojawi się po naciśnięciu na daną kontrolkę. Do usuwania wpisywanego przez omówione elementy ciągu, służy przycisk z wizerunkiem standardowego klawisza do usuwania tekstu w klawiaturach systemu Android. Kolejną kontrolką jest przycisk z napisem „GOTÓW". Służy on do wyrażenia gotowości i przejścia do następnego etapu rozgrywki. *ImageButton*, z wizerunkiem mikrofonu, służy do włączenia wprowadzania głosowego.

Widok zawiera dwa pola tekstowe. Pierwsze, na samej górze, służy do informowania użytkownika o aktualnym etapie testu. W drugim, poniżej, ukazuje się ciąg do zapamiętania, wpisywana przez użytkownika zapamiętana sekwencja oraz statystyki po zakończeniu rozgrywki. Dodatkowo, dookoła drugiego pola tekstowego znajduje się obramowanie. Wygląd ramki został utworzony w pliku frame.xlm opisanym w Rozdziale 2.3.1.6.

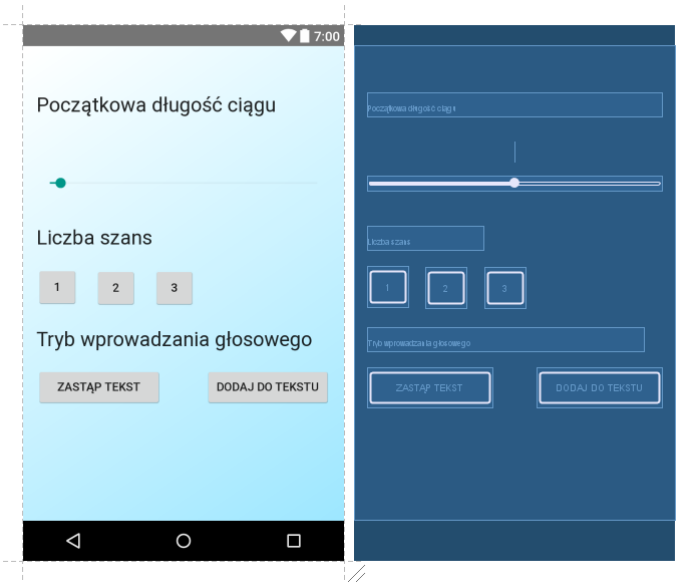
W projekcie widoku niektóre elementy nakładają się na siebie, jednakże podczas testu, kontrolki zostały tak zaprogramowane, że zmieniają swoje położenie w zależności od danego etapu rozgrywki. Wszystko jest zaprojektowane w możliwie czytelny dla użytkownika sposób.



Rys. 2.3: Projekt wyglądu okna rozgrywki.

2.3.1.3 Widok activity_options

Okno opcji pokazuje dostępną możliwość konfiguracji (Rys. 2.4). Trzy pola tekstowe informują użytkownika jakie ustawienia modyfikuje. Znajdujące się przyciski pokazują swoim tekstem skutek zmiany danej opcji. Przesunięcie suwaka zmienia początkową długość ciągu. Jest on przesuwalny w wartościach od 0 do 49, jednak do wybranej wartości suwaka dodawana jest jedynka. Da to możliwość ustawienia początkowej długości ciągu w zakresie od 1 do 50. Pole tekstowe, znajdujące się nad suwakiem, wyświetla zadaną wartość początkowej długości sekwencji.

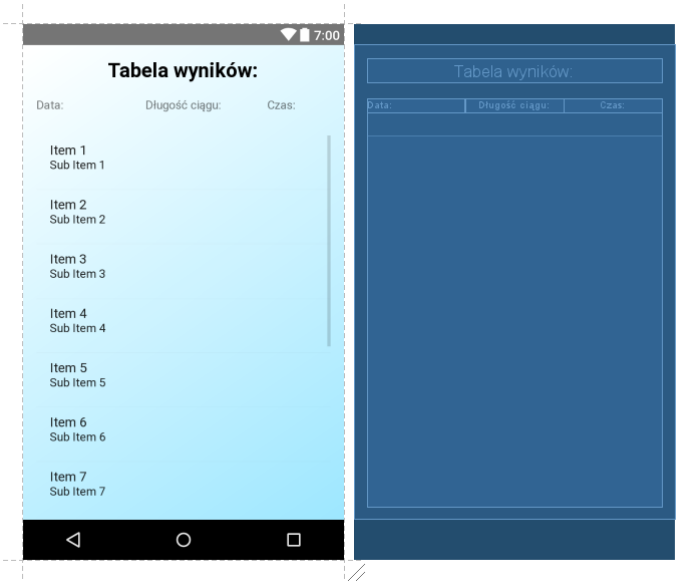


Rys. 2.4: Projekt wyglądu menu opcji.

2.3.1.4 Widok activity_results

Jest to plik widoku, odpowiedzialny za wyświetlenie wyników ukończonych testów (Rys. 2.5). Informuje o tym pole tekstowe, znajdujące się na samej górze okna. Poniżej znajduje się liniowy układ horyzontalny, w skład którego wchodzą trzy pola tekstowe. Informują one o typie wyświetlanej pod tekstem pozycji.

W celu wyświetlenia samych wyników, została dodana kontrolka *ListView*. Lista jest przewijalna i nieograniczona w ilości pokazywanych elementów. W celu prawidłowego wyświetlania żądanych wyników, został dodany plik formatujący pojedynczy wiersz listy. Został on opisany w Rozdziale 2.3.1.5.



Rys. 2.5: Projekt wyglądu listy wyników.

2.3.1.5 Widok list_row

Układ ustawiający wygląd pojedynczego wiersza listy z wynikami. Wiersz składa się z trzech pól – data wykonania testu, długość zapamiętanego ciągu oraz czas zapamiętywania najdłuższej, poprawnie odtworzonej sekwencji. Pole z datą jest dłuższe od pozostałych pól. Zapobiega to zawijaniu się tekstu w danym wierszu.

2.3.1.6 Widok frame

Plik zawiera styl ramki wokół pola tekstowego wyświetlającego np. ciąg cyfr do zapamiętania. Ramka jest koloru czarnego, ma lekko zaokrąglone rogi, a jej szerokość wynosi 1 dp (density independent pixel). Tło wnętrza ramki zostało ustawione na przezroczyste, aby nie zasłaniało tła całego okna.

2.3.1.7 Widok gradient

Odpowiedzialny jest za tło każdego okna aplikacji. Środowisko Android Studio umożliwia w łatwy sposób utworzenie gradientu o zadanych parametrach. Tak wygenerowany gradient należy załączyć do każdego pliku z widokiem, gdzie przeskaluje się na całe okno lub inny żądany wymiar.

2.3.2 Klasy i metody

Kod źródłowy aplikacji składa się z następujących klas:

- Menu.java,
- Numbers.java,
- Tutorial.java,
- Options.java,
- Results.java,
- MultirowLVAdapter.java.

Manifest aplikacji, plik AndroidManifest.xml, zawiera pozwolenia na zapis i odczyt danych z pamięci urządzenia. Plik ten zawiera również ustawienie początkowej aktywności, uruchamianej zaraz po włączeniu aplikacji. Domyślnym układem ekranu dla każdego z widoków jest „portrait" (układ pionowy).

2.3.2.1 Menu

Klasa obsługująca akcje wykonywane w menu głównym aplikacji. Pobiera kontrolki z widoku activity_menu (Opisany w Rozdziale 2.3.1.1). Do przycisków przyporządkowana jest akcja rozpoczęcia odpowiadającej przyciskowi aktywności. Przykładowo, po naciśnięciu klawisza z napisem „Samouczek", poprzez metodę *startActivity* (metoda środowiska Android Studio), zostaje uruchomiona aktywność odpowiedzialna za samouczek.

W tej aktywności, poprzez metodę *verifyStoragePermissions*, wysyłane jest żądanie o udzielenie pozwolenia na zapis i odczyt plików w pamięci urządzenia. Na początku sprawdzane jest czy aplikacja już nie ma przyznanego pozwolenia. W przypadku, gdy zezwolenia brak, zgłaszane jest stosowne żądanie. Taki zabieg umożliwia pominięcie wysłania żądania przy każdym uruchomieniu aplikacji, a nawet przy każdym wejściu do menu głównego.

Mimo, że przyzwolenia na zapis i odczyt są już przyznane w pliku AndroidManifest.xml, to dla nowszych wersji systemu Android (Android 6.0+), wymagane jest wyświetlenie tej informacji użytkownikowi i zapytanie go o zgodę.

2.3.2.2 Numbers

Klasa obsługująca cały przebieg testu pamięci krótkotrwałej. Pobiera on widok z pliku activity_numbers.xml (Opisany w Rozdziale 2.3.1.2). Głównym elementem działania programu jest metoda *onCreate*, a dokładniej nasłuchiwanie na naciśnięcie przycisku z tagiem 'ok'. Po jego naciśnięciu, w zależności od etapu testu (określonego przez zmienną 'steering'), uruchamiane są odpowiednie metody:

```
ok.setOnClickListener(new View.OnClickListener() {
 @Override
  public void onClick(View v) {
   generated_number.scrollTo(0,0);
   //przejscie z etapu zapamiętywania do wpisywania
   if(steering==0){
    ok.setText("Dalej");
    generated_number.setText("");
    inform.setText("Proszę wpisać zapamiętany ciag cyfr.");
    ivisibility(false);
    steering=1;
    // Reset czasu zapamiętywania.
    time[1]=time[0]; seconds=0; minutes=0;milliseconds=0;
    timer.cancel(); timer.purge();}
   //przejscie z etapu wpisywania do kolejnego zapamiętywania
   else if(steering==1){
    ok.setText("Gotów");
    check(generated_number.getText().toString(),number);
    number = intarray2string(random_number(length));
    ivisibility(true);
    steering=0;
    if(chances>0) {
     generated_number.setText(number);
     Timer(); }
    else { //zakończenie rozgrywki
     ok.setText("Zakończ"); steering=2;
     generated_number.setGravity(Gravity.TOP);
     generated_number.setText("");
     generated_number.setTextSize(TypedValue.COMPLEX_UNIT_SP,25);
     //sprawdzenie czy został zapamiętany jakikolwiek ciag
     if(remembered) {
      if(length-1==1) generated_number.append("Najdłuży zapamiętany
       ciag miał długosć " + (length - 1) + " cyfry.\n");
      else generated_number.append("Najdłuży zapamiętany
       ciag miał długosć " + (length - 1) + " cyfr.\n");
       generated_number.append("Został zapamiętany w czasie:
       " + time[2]);
       SaveToCSV(String.valueOf(length-1), time[2]); }
     else generated_number.append("Niestety
       nie udało się zapamiętać żadnego ciagu cyfr."); }}
   else{ finish(); }}});
```

Wygenerowanie losowego ciągu cyfr przebiega w metodzie *random_number*. Przyjmuje ona parametr 'length', który odpowiada długości wylosowanej sekwencji. Zadeklarowana zostaje zmienna tablicowa typu int o rozmiarze równym wartości przesłanego parametru. Następnie, do każdej komórki tejże tablicy zostaje wpisana losowa cyfra. Tak wygenerowany ciąg zostaje zwrócony do dalszych działań.

Kolejnym krokiem jest przekształcenie wygenerowanego ciągu z formy tablicy liczb całkowitych do typu String. Dzieje się to w metodzie *intarray2string*:

```
public static String intarray2string(int[] table){
   String str="";
   for (int i=0;i<table.length;i++) {
```

```
    str+=String.valueOf(table[i]);}
        return str;}
```

Metoda sprawdzająca poprawność zapamiętanej sekwencji to metoda *check*. Jest ona odpowiedzialna również za konsekwencje porównania:

```
protected void check(String generated_number,
String typed_number){
 TextView inform = (TextView) findViewById(R.id.inform);
 if(generated_number.equals(typed_number)) {
  length++;
  time[2]=time[1] \\zapisanie czasu zapamiętywania tego ciagu;
  inform.setText
  ("Ciag zostal poprawnie zapamiętany, oto kolejny:");
  emembered=true \\ oznaczenie, że użytkownik zapamiętał ciag;}
 else {
  chances--;
   if(chances>0) inform.setText
   ("Ciag nie został poprawnie zapamiętany. Pozostała
    liczba szans: " +chances +"\nOto kolejny cišg: ");
   else{
   inform.setGravity(Gravity.CENTER_HORIZONTAL
   | Gravity.CENTER_VERTICAL);
   inform.setTextSize(TypedValue.COMPLEX_UNIT_SP,40);
   inform.setText("Koniec gry.");}}}
```

Przy ekranie końcowym zapisywany jest wynik testu do pliku CSV – dzieje się to w metodzie *SaveToCSV*. Korzysta ona z zaimportowanej biblioteki *opencsv*. Początkowo sprawdzane jest, czy plik o danej nazwie w danej lokalizacji już istnieje. Jeżeli nie, to program go tworzy. Następnie do zmiennej tablicowej zostają umieszczone dane: data zakończenia rozgrywki, długość zapamiętanego ciągu oraz czas jego zapamiętywania. Poprzez metodę biblioteki *opencsv*, *writeNext* (opisana w Rozdziale 2.3), do pliku zapisywany jest kolejny wiersz z separatorem w postaci znaku ';'.

Pomiar czasu zapamiętywania odbywa się w metodzie *Timer*. Inkrementowane liczniki informują o aktualnym czasie.

```
public void Timer() {
 timer = new Timer();
 timer.schedule(new TimerTask() {
  @Override
   public void run() {
    runOnUiThread(new Runnable() {
     @Override
      public void run() {
       milliseconds++;
       if(milliseconds==100) { seconds ++;
        if (seconds == 60) {
         minutes ++; seconds = 0;}
        milliseconds= 0; }
       // przypisanie wartosci aktualnego czasu do zmiennej.
       time[0] = String.format(String.format ("%02d", minutes) +
       ":" + String.format ("%02d", seconds)+ "." +
       String.format("%02d",milliseconds));}});}},0,10);}
```

Przed rozpoczęciem testu pobierane są aktualne ustawienia. Odpowiedzialna za to funkcja nosi nazwę *Get_Config*. Otwiera on plik z ustawieniami i przypisuje do odpowiednich komórek tablicy typu String zapisane znaki, rozdzielone pionową kreską. Pobrane ustawienia to początkowa długość zapamiętywanego ciągu, liczbę szans oraz typ wprowadzania głosowego.

Sama obsługa wprowadzania głosowego jest zaimplementowana w metodach *promptSpeechInput* oraz **onActivityResult**.

```
private void promptSpeechInput() {
    Intent intent = new Intent
    (RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
     RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
     Locale.getDefault())
    try {
        startActivityForResult(intent, 100);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getApplicationContext(),
        "Wykrywanie mowy niewspomagane",
         Toast.LENGTH_LONG).show();
    }}
protected void onActivityResult
(int requestCode, int resultCode, Intent data) {
    TextView generated_number = (TextView)
     findViewById(R.id.generated_number);
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
     case 100: {
      if (resultCode == RESULT_OK && null != data) {
       ArrayList<String> result = data
      .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
      Voicenumber="";
      // pętla sprawdzajaca każdy znak w celu wykrycia tylko cyfr
     for(int i=0; i<result.get(0).toString().length();i++)
      {
       if(isValidInteger(result.get(0).toString().charAt(i)))
        {
         Voicenumber+=result.get(0).toString().charAt(i);
        }
      }
     if(Voicenumber != ""){
      if(voice==1)
       generated_number.setText(Voicenumber);
      else{
       generated_number.append(Voicenumber);}
      }
     else{
     Toast.makeText(getApplicationContext(),
     "Ciag nie został poprawnie wypowiedziany",
     Toast.LENGTH_SHORT).show(); }}
     break;}}}
```

Przefiltrowanie wykrytego tekstu odbywa się w metodzie *isValidInteger*. Jeżeli wystąpił jakikolwiek inny znak niż liczba całkowita, metoda zwraca fałsz i nie jest on w ogóle brany przy ukazywaniu na ekran. Po zakończonej filtracji, w zależności od wybranego trybu wprowadzania głosowego, pole tekstowe zostaje albo zamienione wprowadzonym ciągiem, albo do tekstu już wprowadzonego zostaje dopisane to, co użytkownik powiedział.

Metoda odpowiedzialna za zmianę wyglądu aplikacji w zależności od aktualnego etapu testu pamięci, to metoda *invisibility*. Pobiera ona zmienną typu bool, mówiącą o tym, czy elementy widoczne na ekranie mają się pokazać lub też stać się niewidoczne. Pierwszym etapem tej funkcji jest pobranie rozmiaru ekranu. Następnie pobiera granice przycisku wyrażającego gotowość, pola tekstowego wyświetlającego chociażby wygenerowaną sekwencję oraz przycisku z wizerunkiem mikrofonu. Służy to późniejszemu przesunięciu kontrolek. W zależności od wartości wprowadzonej do metody zmiennej, znikają lub pojawiają się przyciski odpowiedzialne za wprowadzanie zapamiętanego ciągu, kasowanie znaków oraz za wprowadzanie głosowe. Następnie przesuwany jest przycisk wyrażający gotowość oraz wspomniane wcześniej pole tekstowe, tak, aby nie nakładały się na siebie żadne elementy.

Obsługa wszystkich przycisków wprowadzających dzieje się w metodzie *ButtonClicked*. Sprawdzany jest który przycisk został wciśnięty i podejmowana jest odpowiadająca temu akcja - wprowadzenie czy kasowanie cyfry.

2.3.2.3 Tutorial

Jest to klasa dziedzicząca po klasie *Numbers* (opisana w Rozdziale 2.3.2.2), i korzystająca z tego samego widoku. Klasa ta ma za zadanie obsługę samouczka, uruchamianego po naciśnięciu odpowiedniego przycisku w menu głównym. Etapy samouczka określane są przez warunek wielokrotnego wyboru – switch. Zmienną poddaną temu warunkowi jest zmienna 'step', która informuje o aktualnym etapie samouczka. Po zakończonej jednej części, przy naciśnięciu na odpowiedni przycisk, 'step' przyjmuje inną wartość, przez co po następnym kliknięciu przejdzie do innego etapu. Cały samouczek opisany jest w Rozdziale 3.3.

2.3.2.4 Options

Klasa obsługująca wszystkie zmiany zachodzące w ustawieniach. Korzysta z widoku activity_options (opisany w Rozdziale 2.3.1.3). Na wybór opcji składają się trzy grupy - obsługa suwaku ustawiającego początkową długość ciągu, przyciski z liczbą szans oraz przyciski trybu wprowadzania głosowego.

Po jakiejkolwiek zmianie pozycji suwaka, pobiera się jej wartość, dodaje się do niej jeden i wpisuje ją do zmiennej 'lenght'. Również wartość ta jest wpisywana do pola tekstowego nad suwakiem w celu powiadomienia użytkownika o aktualnej wartości początkowej długości ciągu.

Wybór liczby szans jest określana w metodzie *click_chance*. Uruchamia się ona po każdorazowym kliknięciu w przycisk z określoną liczbą szans. Wartość tego przycisku wpisywana jest do zmiennej 'chances'.

Tryb wprowadzania głosowego działa na podobnej zasadzie co wybór liczby szans. kliknięcie w przycisk uruchamia metodę *click_voice*. Tam, w zależności od klikniętego przycisku, do zmiennej 'voice' przyporządkowana jest wartość 1 lub 2. Pierwsza wartość oznacza zamianę całego, widocznego tekstu po wprowadzeniu głosowym. Druga opcja pozwala użytkownikowi na dopisanie do wyświetlonego tekstu tego, co wprowadzi się głosowo.
Po każdej zmianie opcji uruchamia się metoda *Save_Config_to_File*, która przyjmuje omówione wcześniej zmienne konfiguracyjne. Metoda Zapisuje wartości zmiennych do pliku tekstowego, oddzielając każdą wartość kreską pionową.
Również po każdej zmianie opcji oraz zaraz po uruchomieniu menu ustawień, uruchamiana zostaje metoda *Configure_Layout*. Pobiera ona wartości zmiennych konfiguracyjnych i na ich podstawie dostosowuje wygląd aplikacji. Zapisuje wartość początkowej długości ciągu do pola tekstowego oraz podświetla przyciski odpowiadające aktualnie wybranej opcji - liczbie szans i trybu wprowadzania głosowego.

Przy pierwszym uruchomieniu ekranu opcji, sprawdzane jest, czy plik z opcjami istnieje. Dzieje się to w metodzie *Check_if_config_exists*. Jeżeli plik konfiguracyjny nie istnieje, tworzy go, wpisując wartości odpowiadające konkretnym opcjom. Domyślnie, początkowa długość ciągu wynosi trzy, liczba szans również jest równa trzem, a tryb wprowadzania głosowego jest ustawiony na zamianę tekstu.

Aktualne ustawienia pobierane są w metodzie *Get_Config*. Dzieje się to zaraz po sprawdzeniu czy plik konfiguracyjny istnieje. Z pliku zostają odczytane i przyporządkowane do zmiennych odpowiednie wartości. Zabieg ten jest konieczny, by zapamiętywać aktualne ustawienia, w celu uniknięcia każdorazowego resetu konfiguracji przy uruchomieniu opcji.

2.3.2.5 Results

Klasa obsługująca wyświetlenie listy wyników na ekranie. W metodzie *ReadFromCSV*, korzystającej z biblioteki *opencsv*, opisanej w Rozdziale 2.3, dochodzi do wczytania listy wyników z pliku csv. Początkowo zostaje pobrana nazwa pliku i jego ścieżka. Następnie, dzięki metodzie biblioteki *opencsv – readAll*, wczytane są wszystkie wiersze, i kolumny oraz zostają one dopisane do listy typu String tablicowy.

Pobrana lista jest wyświetlana na ekranie dzięki funkcji *setAdapter*. Jest to funkcja środowiska Android Studio, pozwalająca na dowolne formatowanie wyświetlanej listy według wcześniej ustalonego widoku. Przyporządkowanie adaptera dzieje się w klasie *MultirowLVAdapter*, opisanej poniżej.

2.3.2.6 MultirowLVAdapter

Skrypt ten odpowiada za formatowanie wyświetlanej listy. Zawiera kod wygenerowany automatycznie przez środowisko Android Studio wraz z modyfikacjami. Przeciążona metoda *getView* pobiera układ z pliku list_row.xml (opisany w Rozdziale 2.3.1.5). Następnie, wprowadzona lista z danymi zostaje zmapowana (każda komórka listy ma swoją pozycję), aby w końcowym etapie można było przyporządkować każdy element listy do odpowiedniej komórki.

3. Instrukcja użytkownika

W niniejszym rozdziale zawarto instrukcję użytkownika. Obejmuje ona wszystkie elementy związane tak z testem, jak konfiguracją, samouczkiem czy zapisem wyników.

3.1 Menu główne

Po uruchomieniu aplikacji użytkownikowi wyświetlane jest interaktywne menu główne. Menu główne aplikacji służy do szybkiego przejścia między poszczególnymi opcjami. Wygląd menu jest widoczny na Rysunku 3.1. Poprzez kliknięcie w odpowiednie przyciski można rozpocząć grę, zmienić ustawienia aplikacji, uruchomić samouczek lub zobaczyć tabelę wyników.

Rys. 3.1: Menu główne aplikacji.

3.2 Rozgrywka

Test pamięci krótkotrwałej rozpoczyna się po wciśnięciu pierwszego przycisku („rozpocznij grę") w menu głównym. Od razu po uruchomieniu wyświetla się ciąg cyfr do zapamiętania przez użytkownika. (Rys. 3.2). Po wyrażeniu gotowości, przechodzi się do następnego etapu, którym jest wpisanie zapamiętanej sekwencji (Rys 3.3). Gdy użytkownik stwierdzi, że odtworzył zapamiętany ciąg, powinien przejść do kolejnej części, naciskając odpowiedni przycisk. Tam dowie się, czy sekwencja została odtworzona poprawnie (Rys. 3.4), czy też nie (Rys. 3.5). Jeżeli ciąg został odtworzony poprawnie, stopień trudności zwiększy się (nowy ciąg będzie dłuższy o jedną cyfrę). W przeciwnym razie gracz traci jedną z szans, jednakże zapamiętywana sekwencja będzie tej samej długości. Ciąg do zapamiętania za każdym razem będzie inny, losowy.

Warunkiem zakończenia testu jest utrata wszystkich szans. Na ekranie końcowym wyświetla się informacja o końcu rozgrywki, a poniżej ukazują się statystyki związane z testem – długość najdłuższego zapamiętanego ciągu oraz czas, w którym został zapamiętany (Rys. 3.6). Statystyki zostają od razu zapisane do pliku znajdującego się w pamięci urządzenia. Po kliknięciu w przycisk u dołu ekranu, aplikacja powraca co menu głównego. Przerwanie gry poprzez wyjście z aplikacji nie powoduje zapisania wyników. Jest to forma zabezpieczenia przed oszukiwaniem. Limit 50 cyfr, obowiązujący przy ustawianiu początkowej długości w opcjach, nie ogranicza maksymalnej długości zapamiętywanego ciągu w grze.

Rys. 3.2: Rozpoczęcie testu pamięci krótkotrwałej.

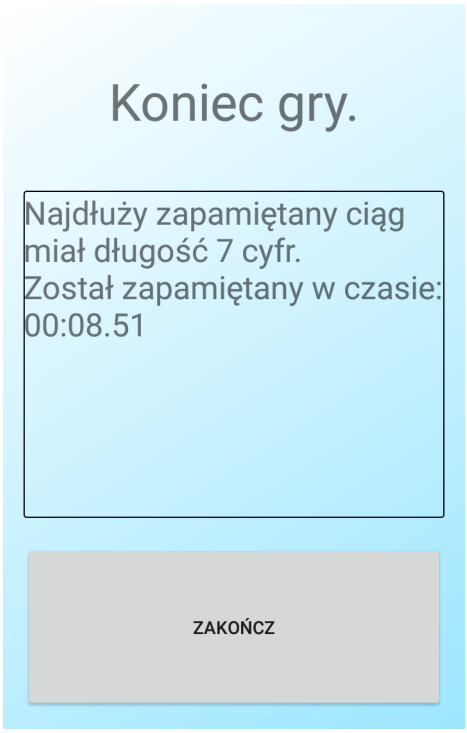
3.2.1 Odtwarzanie ciągu

Użytkownik ma dwie możliwości wprowadzania zapamiętanego ciągu. Albo poprzez ręczne wpisywanie z ukazanej klawiatury (Rys. 3.3), albo poprzez wprowadzenie głosowe. Po kliknięciu w przycisk z wizerunkiem mikrofonu, graczowi ukazuje się okno widoczne na Rysunku 3.7. Po takim komunikacie należy wypowiedzieć zapamiętywaną sekwencję i poczekać na analizę. Aplikacja z całego wypowiedzianego ciągu wybiera jedynie rozpoznane liczby lub pojedyncze cyfry, a w przypadku ich braku wyświetlany jest stosowny komunikat (Rys. 3.8). W zależności od opcji konfiguracyjnej. W zależności od opcji konfiguracyjnej, kolejne wybranie przycisku z klawiatury głosowej pozwala kontynuować wprowadzanie ciągu lub zastępuje ciąg poprzednio wprowadzony.

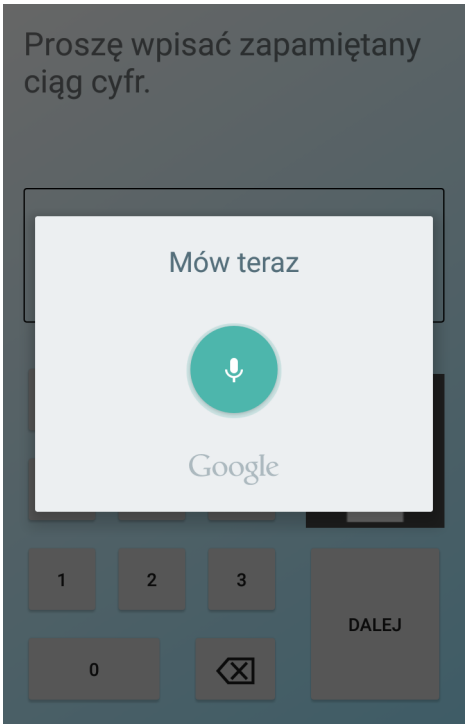
Rys. 3.3: Ekran wpisywania zapamiętanego ciągu.

Rys. 3.4: Komunikat o dobrze zapamiętanym ciągu oraz rozpoczęcie kolejnego etapu testu.

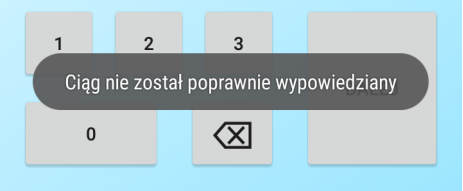
Rys. 3.5: Komunikat o źle zapamiętanym ciągu oraz rozpoczęcie kolejnego etapu.



Rys. 3.6: Zakończenie testu i wyświetlenie statystyk.



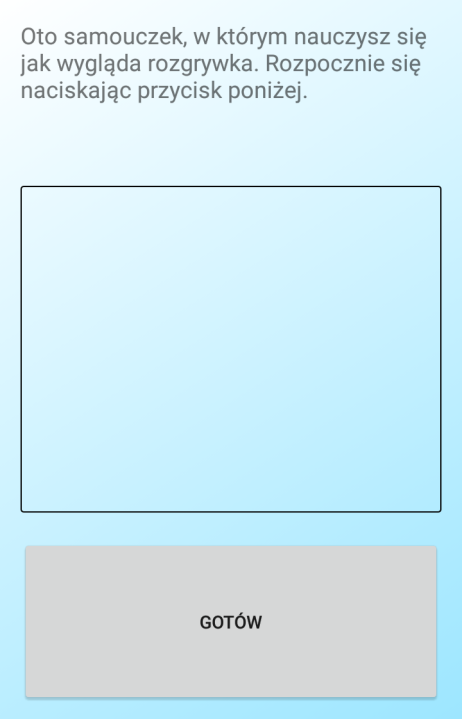
Rys. 3.7: Uruchomienie wprowadzania głosowego.



Rys. 3.8: Wychwycenie błędu złego wprowadzania głosowego.

3.3 Samouczek

Samouczek wprowadza użytkownika w zasady panujące podczas rozgrywki (Rys. 3.9). Na początku, użytkownik zostaje zaznajomiony z losowym generowaniem ciągu. Następnie jest instruowany co należy zrobić w kolejnych etapach rozgrywki. Podczas samouczka wprowadza się także użytkownika w możliwość wprowadzania głosowego zapamiętanego ciągu i instruuje się go, jak należy z tej opcji korzystać. Po zakończeniu samouczka, aplikacja przechodzi do właściwego testu pamięci.



Rys. 3.9: Rozpoczęcie samouczka.

3.4 Opcje

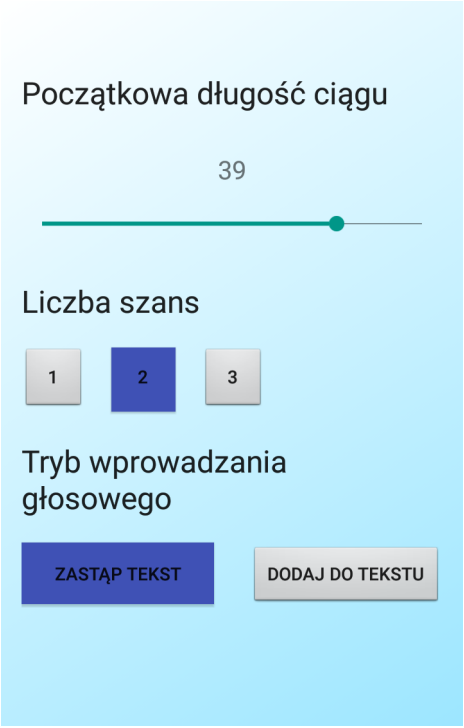
Ekran opcji pozwala wybrać tryb działania niektórych opcji programu (Rys. 3.10).

1. Opcja "początkowa długość ciągu" odpowiada za zmianę początkowej długości zapamiętywanego ciągu. Przesuwając suwak, zmienia się ta wartość. Dla zwiększenia czytelności, długość ta jest wypisana nad kontrolką.

2. Opcja "Liczba szans" odpowiada za zmianę liczby szans podczas testu. Przy trzech szansach użytkownik może pomylić się dwa razy, a trzecia pomyłka spowoduje zakończenie rozgrywki. Analogicznie, jedna szansa oznacza spowodowanie końca testu pamięci po pierwszej pomyłce.

3. Opcja "Tryb wprowadzania głosowego" odpowiada za zmianę trybu wprowadzania głosowego. Możliwe wybory to zastąpienie tekstu po wypowiedzeniu nowego lub dodanie kolejnej części do już wprowadzonego.

Zmiany opcji są od razu zapamiętywane i wprowadzone do rozgrywki. Użytkownik informowany jest o zmianie ustawień poprzez podświetlenie kliknięte go przycisku.



Rys. 3.10: Ekran opcji aplikacji.

3.5 Wyniki

Tabela wyników (Rys. 3.11) jest widoczna po wybraniu tak oznaczonego przycisku z menu głównego. Uzupełniana jest po każdorazowym, poprawnym zakończeniu testu pamięci. Przerwana gra nie skutkuje zapisaniem wyniku. Pola zawarte w tabeli wyników to:

- data zakończenia testu,
- największa długość zapamiętanego ciągu,
- czas w którym został zapamiętany najdłuższy ciąg.

Wyniki sortowane są po dacie zakończenia tekstu. Tabela jest przewijalna, co umożliwia odczytanie wielu wyników jednocześnie. Niemożliwe jest usunięcie pojedynczego wiersza lub całej tabeli.

Rys. 3.11: Tabela wyników.

4. Testy

W ramach weryfikacji poprawności zadania, opracowana aplikacja została intensywnie przetestowana na etapie implementacji. Zweryfikowano m.in. działanie wszystkich funkcji dla urządzeń o różnym rozmiarze ekranu i wersji systemu

Następnie gotowa aplikacja wykorzystana została w badaniu pamięci ochotników. Przebadano trzy grupy:

- osoby zdrowe, nie trenujące pamięci,
- osoby zdrowe, regularnie trenujące pamięć,
- osoby w różnych kategoriach wiekowych.

Liczebność grup była zróżnicowana:

- 9 osób zdrowych, nie trenujących pamięć,
- 3 osoby zdrowe, regularnie trenujące pamięć,
- 2 osoby w wieku poniżej 18 lat, 7 osób w wieku od 18 do 35 lat oraz 3 osoby w wieku powyżej 35 lat.

Każda osoba testująca miała wypełnić następujące pozycje (oprócz kwestionariusza osobowego):

1. Zadeklarowana i faktyczna długość zapamiętanego ciągu.
2. Zmierzony czas zapamiętywania najdłuższego ciągu.
3. Komentarz na temat ogólnego korzystania z aplikacji.
4. Komentarz dotyczący funkcji głosowej.

Uzyskane wyniki zostały zamieszczone w tabelach dla każdych grup (Tabela 4.1).

Każda grupa osób bezproblemowo poradziła sobie z korzystaniem z aplikacji. Nie wszyscy jednak postanowili skorzystać z samouczka i dopiero po pierwszej rozgrywce zaznajomili się z zasadami. Interfejs użytkownika został zaprojektowany w czytelny sposób, zatem nie utrudniał on użytkowania. Również samo tło nie odwracało uwagi i nie przerywało skupienia przy wykonywaniu testu.

Osoby trenujące pamięć nie były przychylne do korzystania z opcji wprowadzania głosowego. Wydłużało to prowadzenie testu pamięci, co skutkowało zapominaniem

Grupa	Śr. długość ciągu	Std. długości ciągu	Śr. czas zapamiętywania	Std. czasu zapamiętywania	Min. długość ciągu	Max. długość ciągu
Zdrowi, nie trenujący pamięci	8.56	0.83	11.67	2.49	7	10
Zdrowi, testujący pamięć	37.33	16.11	34.33	20.04	24	60
Wiek poniżej 18 lat	9.5	0.5	9.5	0.5	9	10
Wiek powyżej 18 i poniżej 35 lat	20.86	17.75	20.71	17.68	8	60
Wiek powyżej 35 lat	8	0.82	14.67	1.25	7	9

Tab. 4.1: tabelaryczne zestawienie statystyk dla każdej z grup.

części zapamiętanego ciągu. Dla reszty grup była to ciekawa opcja, pozwalająca na wykonanie testu pamięci nawet w ruchu.

Jedyną istotną uwagą do aplikacji była uwaga o możliwości rozegrania tylko jednego rodzaju testu. Jednakże, jako że kod aplikacji został napisany w sposób przystępny dla nawet początkującego programisty, istnieje łatwa możliwość rozszerzenia programu o kolejne testy. Kolejnym elementem rozwoju aplikacji jest zapisywanie wyników do zewnętrznej bazy danych, a nie do pliku na urządzeniu. Da to możliwość diagnozowania zdolności pamięciowych oraz dokonania ewentualnych badań naukowych.

5. Podsumowanie i wnioski

Celem pracy było stworzenie testu pamięci krótkotrwałej w formie aplikacji mobilnej. Dodatkowo, aplikacja miała posłużyć jako forma treningu pamięci krótkotrwałej.

W trakcie realizacji wykonano następujące prace: dokonano analizy założeń i zaprojektowano aplikacje (Rozdział 2). Następnie całość zaimplementowano jako aplikację systemu Android (Rozdział 2.3). W ramach aplikacji możliwe jest wykonanie testu pamięci polegającego na zapamiętywaniu losowego ciągu cyfr o różnych długościach (Rozdział 3). Dostępna jest także klawiatura głosowa, zapis wyników, zmiana opcji rozgrywki oraz samouczek wprowadzający w zasady testu. W kolejnym kroku całość przetestowano (Rozdział 4). Zweryfikowano m.in. działanie aplikacji w badaniu z grupą ochotników.

Dzięki zrobieniu testu w formie gry, użytkownik może również trenować pamięć. Poprzez listę wyników można w łatwy sposób stwierdzić, czy trening przynosi skutki. Dzięki temu aplikacja może posłużyć w celach diagnostycznych jak i terapeutycznych.

Podsumowując można stwierdzić, że zrealizowano cel pracy.

Bibliografia

[1] Anderson, J. R. *Uczenie się i pamięć – integracja zagadnień.* Wydawnictwa Szkolne i Pedagogiczne, Warszawa, 1998.

[2] Budohoska, W., W. Z. *Psychologia uczenia się.* Przegląd badań eksperymentalnych i teorii. Państwowe Wydawnictwo Naukowe, Warszawa, 1977.

[3] Gazzaniga, M. S. *The Cognitive Neurosciences*, 4 ed. The MIT Press, Londyn, 2009.

[4] Jagodzińska, M. *Psychologia pamięci. Badania, teorie, zastosowania.* Sensus, 2008.

[5] Jodzio, K. *Neuronalny Świat umysłu.* Oficyna wydawnicza Impuls, Kraków, 2005.

[6] Niedźwieńska, A. Rodzaje testów do badania pamięci.

[7] Stasiewicz, A. *Android Studio. Podstawy tworzenia aplikacji.* Helion, Gliwice, 2005.

[8] Vetulani, J. Pamięć: podstawy neurobiologiczne i możliwości wspomagania.