תרגיל בית 1 – חלק יבש

Exercise 1:

1.) השורות שגורמות לרשימה להיות infinitely scrolling הן:

```
if (index >= _suggestions.length)
  _suggestions.addAll(generateWordPairs().take(10));
```

בעת הגלילה, המשתמש "דורש" את איברי הרשימה באינדקסים הולכים וגדלים.

בכל הגעה לסוף הרשימה (ע"י הבדיקה בשורה הראשונה), השורה השנייה **מוסיפה איברים נוספים לרשימה**, ואז המשתמש תמיד יוכל לגלול ולקבל עוד ועוד איברים (והאינדקס תקין), ומתקבל אפקט ה-infinitely scrolling.

לפי ההנחה, כשהמשתמש ימשיך לגלול לאחר קבלת 20 איברי _suggestions, ייזרק Range Error מכך שיהיה ניסיון לגשת לאיבר הבא ברשימה **שלא התארכה** (בגישה _suggestion[index] שמתחת עם index גדול מ-20), באינדקס שלא קיים.

2.) דרך נוספת ליצירת רשימה עם Dividers היא באמצעות Widget בשם ListView.separated.

ה-Widget מאפשר ליצור את איברי הרשימה ע"י פונקציה שמועברת כארגומנט בשם itemBuilder, וליצור מפרידים ע"י הארגומנט separatorBuilder באופן דומה, כאשר יצירת הרשימה עם המפרידים נעשית בקריאה לפונקציות הללו כך שנוצר מפריד בין כל שני איברים.

מימוש אפשרי לרשימה עם dividers תחת ההנחה של סופיות הרשימה:

```
return ListView.separated(
  padding: const EdgeInsets.all(16),
  itemCount: _suggestions.length,
  itemBuilder: (BuildContext _context, int index) {
    if (index >= _suggestions.length) {
      _suggestions.addAll(generateWordPairs().take(10));
    }
    return _buildRow(_suggestions[index]);
  },
  separatorBuilder: (BuildContext _context, int index) => const Divider(),
);
```

לדעתי השימוש ב-ListView.separate היא דרך טובה יותר, מכיוון שהיא מאפשרת הפרדה בין הלוגיקה של יצירת איברי הרשימה (המטרה העיקרית) לבין הלוגיקה של יצירת המפרידים (מעין תוצר לוואי).

הפרדה זו מאפשרת קוד נקי וברור יותר - אין צורך בשליטה ידנית של סדר הוספת האיברים והמפרידים שמתבצע אוטומטית, וההפרדה בין שתי המשימות יכולה להוביל לקיצור הקוד הדרוש ולהוסיף לקריאות.

בחלק הרטוב בתרגיל ניתן לראות דוגמה לכך – בשימוש ב-separated היינו נמנעים מ"האקים" של אינדקסים, והקוד של יצירת האיברים היה קצר וברור יותר.

3.) OnTap() משנה את הרשימה _saved ולכן משנה את ה-state של ה-RandomWords widget.

כפי שראינו בהרצאה, השימוש ב-setState() דרוש על מנת שהשינוי ב-state (דרך שינוי _saved) יגרום ל-re-rendering של ה-widget והצגתו על המסך בצורה המעודכנת שלו, לאחר שינוי ה-state.

ללא שימוש וביצוע השינויים בתוך פונקציה זו, השינוי שבוצע ע"י OnTap() לא היה גורם ל-re-rendering ואז לא יכולנו לראות את השינוי על המסך.

Exercise 2:

1.) כפי שראינו ב-2 part, Widget ששמו ה-Navigator מתחזק מחסנית של routes (מסכים/דפים שבהם המשתמש מבקר). על מנת לעבור אל route חדש, דוחפים את ה-route החדש אל המחסנית, ושולפים אותו על מנת לחזור ל-route הקודם.

לכן, הדרך שבה מימשתי את המעבר מהמסך הראשי אל מסך ה-login היא ליצור route חדש שמכיל את מבנה הדף ולדחוף אותו למחסנית ה-Navigator (הפונקציה _pushLogin).

דרך נוספת לממש את ה-Navigation עם אותה תוצאה היא בעזרת המתודה pushNamed של ה-Navigator:

מגדירים Widget-ים עבור ה-routes (מתודת ה-build יוצרת את ה-Scaffold המתאים) ורושמים אותם ב-Widget הראשי של האפליקציה – ממפים תחת "routes" את שם ה-route (שם של כתובת המשויכת ל-route, למשל '/example') אל המופע של ה-Widget המתאים.

כעת וכדי לגשת אל route מסוים קוראים ל-()pushNamed עם השם המתאים.

כלומר, במקום לבצע:

```
Navigator.of(context).push(
  MaterialPageRoute<void>(
    builder: (BuildContext context) {
      ...
    }),
);
```

מגדירים ורושמים מראש את ה-routes ואז עבור route מסוים מבצעים:

```
Navigator.of(context).pushNamed(<ROUTE_NAME>);
```

כאשר ROUTE_NAME הוא השם שהוגדר במיפוי ל-route הרצוי.

2.) הוספתי ל-builder שאחראי על בניית ה-Scaffold, הוא ה-Widget של המסך המוצג, את קטע הקוד שיוצר את ה-tiles של איברי הרשימה _saved.
שם, הוספתי לכל ListTile ב-trailing את הערך הבא:

```
trailing: IconButton(
    icon: Icon(Icons.delete),
    onPressed: () {
      final deletionSnackBar = SnackBar(
          content: Text('Deletion is not implemented yet',
              style: _biggerFont));
      Scaffold.of(context).showSnackBar(deletionSnackBar);
    })
```

הקוד הזה יוצר IconButton עם סמל Delete, ולו המתודה onPressed שמחפשת את ה-Scaffold המוצג תחת ה-context, וקוראת להצגת ה-snackbar ע"י המתודה showSnackBar של ה-Widget.

ה-Widget הנוסף הדרוש להצגת ה-snackBar הוא **Scaffold** שמציג בפועל את ה-snackBar, שכן ההצגה מתרחשת ע"י קריאה למתודה שלו.