ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Кваліфікаційна робота
Перший (бакалаврський рівень)

Програмний застосунок гри-вікторини

Студент групи КІУКІ-21-4
Даніїл Ісамутдінов

Керівник: ст. викл. Наталія Єрьоміна

2025

Актуальність теми: навчання та змагання

Гейміфікація в освіті

Зростаючий інтерес до ігрових елементів для залучення студентів.

Автономність

Потреба в офлайн-інструментах, особливо після COVID-19 та в умовах війни.

Вікторини

Поєднання навчання, змагання та самоперевірки для ефективного засвоєння матеріалу.

Мета та завдання: створення гри-вікторини

Створення застосунку

Розробка додатка-вікторини для мобільних пристроїв.

Зручний інтерфейс

Реалізація інтуїтивно зрозумілого та легкого у використанні інтерфейсу.

Парсер файлів

Розробка ефективного парсера для обробки текстових файлів з питаннями.

Редактор тестів

Створення функціоналу для розробки та редагування власних тестів.

Автономна робота

Забезпечення повної функціональності без необхідності підключення до інтернету.

Аналіз аналогів: переваги Challenging Game

Оглянуті існуючі платформи

- Kahoot!, Quizizz, Socrative, Mentimeter…

Наш застосунок

- Безкоштовний
- Повністю офлайн
- Можливість створення власних наборів питань

Обмеження аналогів

- Обов'язковий інтернет
- Відсутність локального збереження
- Обмежені можливості редагування

Вибір технологій: Dart та Flutter для ефективності

Dart

Обрано за швидкість розробки та ефективність.

Flutter

Фреймворк для кросплатформної розробки та простоти адаптації.

Майбутня Мультиплатформність

Підтримка розробки для різних операційних систем.

Android Studio

Інтегроване середовище для розробки та налагодження.

Функціональні можливості:

1. **Завантаження файлів**
   Підтримка .txt-файлів з питаннями для зручного імпорту.
2. **Ігровий процес**
   Вибір тем, проходження питань, облік відповідей та балів.
3. **Редактор тестів**
   Можливість створювати та змінювати власні вікторини в застосунку.
4. **Збереження даних**
   Зберігання створених тестів у локальному файлі на пристрої.
5. **Повна автономність**
   Застосунок працює без підключення до мережі інтернет.

Інтерфейс користувача: простота та інтуїтивність

Головне меню

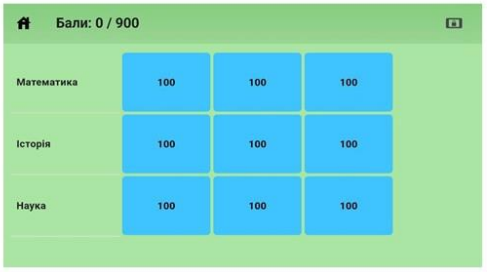
- Відкрити файл
- Створити тест
- Інформація про автора

Процес гри

- Вибір файлу → Валідація → Початок гри
- Таблиця питань, підказки, підрахунок балів

Редактор тестів

- Поля для тем, питань та відповідей
- Функція експорту у локальний файл



Тестування: Забезпечення стабільності та надійності

Логіка

Ретельна перевірка ігрової логіки та функціоналу.

UI/UX

Повністю перевірено користувацький інтерфейс та досвід.

Виправлення

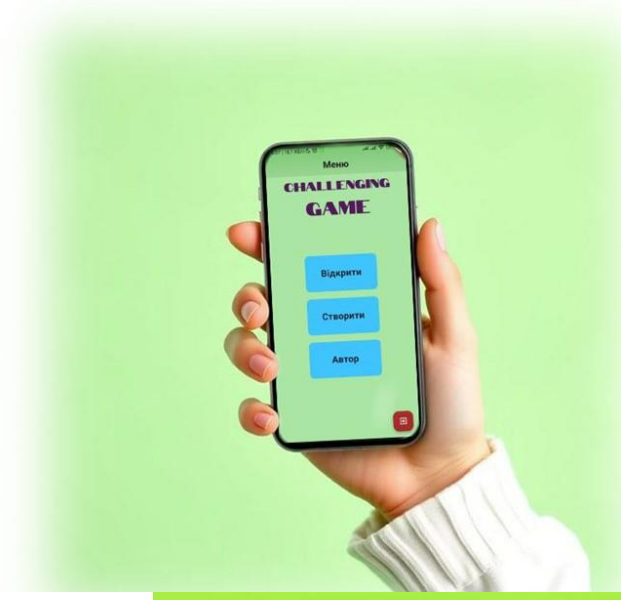
Усі виявлені баги (скрол, орієнтація, формат файлів) усунуто.

Крайові випадки

Тестування нестандартних сценаріїв та введення даних.

Результати: готовий та функціональний застосунок

Розроблено повністю функціональний мобільний застосунок «Challenging Game». Він є простим у використанні, стабільним та не потребує підключення до інтернету. Дозволяє користувачам ефективно створювати, редагувати та проходити вікторини.



Висновки та перспективи: майбутнє Challenging Game

Застосування

Використання в освітніх закладах та на тренінгах.

Статистика

Додавання функціоналу для відстеження результатів.

Таймер

Реалізація таймера для питань та загального часу вікторини.

Кросплатформенність

Розширення на iOS та Web-версії застосунку.

Розробку успішно завершено, всі поставлені завдання виконано. Застосунок є готовим до впровадження та подальшого розвитку.

ДОДАТОК Б
ПРОГРАМНИЙ КОД

Б.1 main.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:projects/question_editor_screen.dart';
import 'package:projects/pick_file.dart';

int selectedThemeCount = 0;
int selectedMaxQuestionsPerTheme = 0;

void main() => runApp(CGameApp());

class CGameApp extends StatelessWidget {
  const CGameApp({super.key});

  @override
  Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations(
        [DeviceOrientation.portraitUp]);
    return MaterialApp(
        theme: ThemeData(scaffoldBackgroundColor: const
Color(0xffaae5a4)),
        title: "CGame",
        home: Menu());
  }
}

class Menu extends StatelessWidget {
  const Menu({super.key});


  @override
  Widget build(BuildContext context) {
    return Scaffold(

      appBar: AppBar(
        title: Text("Меню", style: TextStyle(fontSize:
30,fontWeight: FontWeight.bold),),
        centerTitle: true,
        // backgroundColor: const Color(0x11000000),
        flexibleSpace: Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(
                begin: Alignment.topCenter,
                end: Alignment.bottomCenter,
                colors: <Color>[Color(0xff7ac972),
```

```
Color(0xff9ad594)]),
          ),
        ),
      ),
      body: Center(
        child: Column(
          // mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            // const SizedBox(
            //   height: 40,
            // ),
            Text(
              'Challenging',
              style: TextStyle(
                fontFamily: "Notable",
                fontSize: 40.0,
                color: Color(0xff551a5b),
              ),
            ), //challenging
            Text(
              'game',
              style: TextStyle(
                fontFamily: "Notable",
                fontSize: 60.0,
                color: Color(0xff551a5b),
              ),
            ), //game
            const SizedBox(height: 100,),
            // InkWell(
            //   onTap: () {
            //
SystemChrome.setPreferredOrientations([DeviceOrientation.landsca
peLeft]);
            //     Navigator.push(
            //         context, MaterialPageRoute(builder:
(context) => const QuestionsScreen()));
            //   },
            //   customBorder: RoundedRectangleBorder(
            //       borderRadius: BorderRadius.circular(12.0)),
            //   child: Ink(
            //     width: 200,
            //     height: 100,
            //     // alignment: Alignment.center,
            //     decoration: BoxDecoration(
            //       color: Colors.lightBlueAccent,
            //       borderRadius: BorderRadius.circular(12.0),
            //     ),
            //     child: const Center(
            //       child: Text(
            //         "Грати",
            //         style: TextStyle(fontSize: 30,
fontWeight: FontWeight.bold),
            //       ),
```

```
            //     ),
            //   ),
            // ), // Грати
            InkWell(
              onTap: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(builder: (context) =>
const FilePickerS()));
              },
              customBorder: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(12.0)),
              child: Ink(
                width: 200,
                height: 100,
                // alignment: Alignment.center,
                decoration: BoxDecoration(
                  color: Colors.lightBlueAccent,
                  borderRadius: BorderRadius.circular(12.0),
                ),
                child: const Center(
                  child: Text(
                    "Відкрити",
                    style: TextStyle(fontSize: 30, fontWeight:
FontWeight.bold),
                  ),
                ),
              ),
            ),
            const SizedBox(height: 20,),
            InkWell(
              onTap: (){
                showDialog(context: context, builder: (context)
=> AlertDialog(
                    title: Text('Введіть кількість тем та їх
максимальний розмір:'),
                    content:
                    Column(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        TextField(
                          decoration: const
InputDecoration(labelText: 'Кількість тем'),
                          keyboardType: TextInputType.number,
                          onChanged: (val) => selectedThemeCount
= int.tryParse(val) ?? 0,
                        ),
                        TextField(
                          decoration: const
InputDecoration(labelText: 'Максимальний розмір'),
                          keyboardType: TextInputType.number,
                          onChanged: (val) =>
selectedMaxQuestionsPerTheme = int.tryParse(val) ?? 0,
```

```
                        ),
                      ],
                    ),
                    actions: [
                      TextButton(
                        onPressed: (){Navigator.push(context,
MaterialPageRoute(builder: (context) => QuestionEditorScreen(
                            themeCount: selectedThemeCount,
                            maxQuestionsPerTheme:
selectedMaxQuestionsPerTheme,
                        )));},
                        child: Text('Відкрити'),
                      )
                    ],
                  ));
                },
                customBorder: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12.0)),
                child: Ink(
                  width: 200,
                  height: 100,
                  // alignment: Alignment.center,
                  decoration: BoxDecoration(
                    color: Colors.lightBlueAccent,
                    borderRadius: BorderRadius.circular(12.0),
                  ),
                  child: const Center(
                    child: Text(
                      "Створити",
                      style: TextStyle(fontSize: 30, fontWeight:
FontWeight.bold),
                    ),
                  ),
                ),
              ),
              const SizedBox(height: 20,),
              InkWell(
                onTap: () {
                  showDialog(
                    context: context,
                    builder: (context) {
                      return AboutDialog(
                        applicationVersion: "1.0.0",
                        applicationName: '''Challenging Game
by Ісамутдінов Даніїл''',
                      );
                    },
                  );
                },
                customBorder: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12.0)),
                child: Ink(
                  width: 200,
```

```
                height: 100,
                // alignment: Alignment.center,
                decoration: BoxDecoration(
                  color: Colors.lightBlueAccent,
                  borderRadius: BorderRadius.circular(12.0),
                ),
                child: const Center(
                  child: Text(
                    "Автор",
                    style: TextStyle(fontSize: 30, fontWeight:
FontWeight.bold),
                  ),
                ),
              ),
            ), //about

            // Container(
            //   width: 200,
            //   height: 100,
            //   alignment: Alignment.center,
            //   decoration: BoxDecoration(
            //     color: Colors.lightBlueAccent,
            //     borderRadius: BorderRadius.circular(12.0),
            //   ),
            //   child: Text(
            //     "Грати",
            //     style: TextStyle(fontSize: 30, fontWeight:
FontWeight.bold),
            //   ),
            // ),
            // Container(
            //   width: 200,
            //   height: 100,
            //   alignment: Alignment.center,
            //   decoration: BoxDecoration(
            //     color: Colors.lightBlueAccent,
            //     borderRadius: BorderRadius.circular(12.0),
            //   ),
            //   child: Text("Автор", style: TextStyle(
            //       fontSize: 30, fontWeight:
FontWeight.bold),),
            // ),
            // const SizedBox(height: 20,),
            // Container(
            //   width: 200,
            //   height: 100,
            //   alignment: Alignment.center,
            //   decoration: BoxDecoration(
            //     color: Colors.lightBlueAccent,
            //     borderRadius: BorderRadius.circular(12.0),
            //   ),
            //
            //   child: Text("Вийти", style: TextStyle(fontSize:
```

```
30, fontWeight: FontWeight.bold),),
            // ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => SystemNavigator.pop(),
        tooltip: 'Exit',
        backgroundColor: Color(0xeeaa2222),
        child: const Icon(Icons.exit_to_app_sharp, color:
Color(0xffffffff),),
      ), // Вихід
    );
  }
}
```

Б.2 pick_file.dart

```
     import 'dart:io';
import 'package:flutter/material.dart';
import 'package:file_picker/file_picker.dart';
import 'package:flutter/services.dart';
import 'package:projects/questions.dart';

bool picked = false;

class FilePickerS extends StatefulWidget {
  const FilePickerS({super.key});

  @override
  _FilePickerExampleState createState() =>
_FilePickerExampleState();
}

class _FilePickerExampleState extends State<FilePickerS> {
  String? fileContent;

  Future<void> pickAndReadFile() async {
    FilePickerResult? result = await
FilePicker.platform.pickFiles(
      type: FileType.custom,
      allowedExtensions: ['txt'],
    );

    if (result != null && result.files.single.path != null) {
      final path = result.files.single.path!;
      final file = File(path);

      final content = await file.readAsString();

      setState(() {
        picked = true;
        fileContent = content;
```

```
      });
    } else {
      setState(() {
        picked = false;
        fileContent = 'Файл не обраний.';
      });
    }
  }

  void validateAndOpen(String content) {
    try {
      final testThemes = parseSimpleQuiz(content); // просто
перевірка
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => QuestionsScreen(fileContent:
content),
        ),
      );
    } catch (e) {
      showDialog(
        context: context,
        builder: (_) => AlertDialog(
          title: Text('Помилка у файлі'),
          content: Text(e.toString()),
          actions: [
            TextButton(
              onPressed: () => Navigator.pop(context),
              child: Text('OK'),
            ),
          ],
        ),
      );
    }
  }



  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text('Завантаження файлу', style:
TextStyle(fontSize: 30,fontWeight: FontWeight.bold)),
          flexibleSpace: Container(
            decoration: const BoxDecoration(
              gradient: LinearGradient(
                  begin: Alignment.topCenter,
                  end: Alignment.bottomCenter,
                  colors: <Color>[Color(0xff7ac972),
Color(0xff9ad594)]),
            ),
```

```
          ),
          leading: IconButton(
              icon: Icon(Icons.house, color: Colors.black),
              onPressed: () => setState(()
{Navigator.of(context).pop();},)
          ),
      ),
      body: Center(
        child: Container(
          margin: EdgeInsets.all(10),
          child: Column(
            children: [
              ElevatedButton(
                onPressed: pickAndReadFile,
                style: ElevatedButton.styleFrom(backgroundColor:
Colors.lightBlueAccent),
                child: Text('Обрати потрібний файл', style:
TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color:
Colors.black)),
              ),
              SizedBox(height: 20),
                Expanded(
                  child: SingleChildScrollView(
                      physics: const
AlwaysScrollableScrollPhysics(),
                    child: Text(fileContent ?? 'Текст файлу буде
відображатись тут'),
                  ),
                ),
            ],
          ),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          if (picked) {
            validateAndOpen(fileContent!);
            //print("ПЕРЕДАНИЙ КОНТЕНТ:");
            //print(fileContent);
            picked = false;
          }
          else {showDialog(context: context, builder: (context)
=> AlertDialog(title: Text('Ви ще не обрали потрібний
файл!')));};
          setState(() {});
        },
        tooltip: '''Let's play''',
        backgroundColor: (picked ? (Colors.green) :
(Colors.grey)),
        child: const Icon(Icons.play_arrow, color:
Color(0xffffffff),),
      ),
    );
```

```
  }
}
```

Б.3 questions.dart

```
import 'package:flutter/material.dart';
import 'package:auto_size_text/auto_size_text.dart';
import 'package:flutter/services.dart';
String ans = '';
class QuestionsScreen extends StatefulWidget {
  final String fileContent;

  const QuestionsScreen({super.key, required this.fileContent});

  @override
  State<QuestionsScreen> createState() =>
_QuestionsScreenState();
}

class _QuestionsScreenState extends State<QuestionsScreen> {
  bool isLandscape = false;
  List<ThemeBlock> themes = [];
  int maxQuestions = 0;
  int maxScore = 0;
  int currentScore = 0;
  int completeCount = 0;
  int answeredCount = 0;
  int totalQuestions = 0;
  bool isParsed = false;


  final ScrollController verticalController =
ScrollController();
  final ScrollController verticalTextController =
ScrollController();

  @override
  void initState() {
    super.initState();
    parseAndPrepare();
    verticalController.addListener(() {
      if (verticalTextController.hasClients &&
          verticalController.offset !=
verticalTextController.offset) {

verticalTextController.jumpTo(verticalController.offset);
      }
    });
    verticalTextController.addListener(() {
      if (verticalController.hasClients &&
          verticalTextController.offset !=
verticalController.offset) {
```

```
verticalController.jumpTo(verticalTextController.offset);
      }
    });
  }

  void parseAndPrepare() {
    try {
      themes = parseSimpleQuiz(widget.fileContent);
      maxQuestions = themes.map((t) =>
t.questions.length).fold(0, (a, b) => a > b ? a : b);
      maxScore = themes
          .expand((t) => t.questions)
          .fold(0, (sum, q) => sum + q.price);
      totalQuestions = themes.fold(0, (sum, t) => sum +
t.questions.length);
      setState(() {
        isParsed = true;
      });
    } catch (e) {
      print('Помилка при парсингу: $e');
    }
  }

  @override
  void dispose() {
    verticalController.dispose();
    verticalTextController.dispose();
    super.dispose();
  }

  void handleAnswer(Question question) {
    if (!question.answered) {
      setState(() {
        question.answered = true;
        if (ans ==
question.correctAnswer.toLowerCase().replaceAll(' ', '')){
          showDialog(context: context, barrierDismissible:
false, builder: (context) => AlertDialog(
              title: Text('Правильна відповідь!'),
              content: Center(heightFactor: 0.4, child:
Text('+${question.price}', style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.green, fontSize: 20))),
          )
          );
          currentScore += question.price;
          answeredCount++;
        }
        else{
          showDialog(context: context, barrierDismissible:
false, builder: (context) => AlertDialog(
              title: Text('Правильна
відповідь:\n\n${question.correctAnswer}'),
              content: Center(heightFactor: 0.4, child: Text('-
```

```
${question.price}', style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.red, fontSize: 20)))
          )
          );
          currentScore -= question.price;
          }
        Future.delayed(Duration(seconds: 4), () {

          Navigator.pop(context);
          Navigator.pop(context);
          setState(() {});
        });
        print(ans);
        print(question.correctAnswer.toLowerCase());
        completeCount += 1;

        if (completeCount >= totalQuestions) {
          Future.delayed(Duration(seconds: 5),(){
            showDialog(
              barrierDismissible: false,
              context: context,
              builder: (_) => AlertDialog(
                title: const Text('Гру завершено'),
                content: Text('Ви заробили $currentScore з
$maxScore очок!\nПравильних відповідей:
$answeredCount/$totalQuestions'),
                actions: [
                  TextButton(
                    onPressed: () {
                      Navigator.of(context).popUntil((route) =>
route.isFirst);

SystemChrome.setPreferredOrientations([DeviceOrientation.portrai
tUp]);
                    },
                    child: const Text('OK'),
                  ),
                ],
              ),
            );
          });
        }
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    if (!isParsed) {
      return Scaffold(
        appBar: AppBar(title: const Text('Зачекайте')),
        body: const Center(child: CircularProgressIndicator()),
      );
```

```
    }

    return Scaffold(
      appBar: AppBar(
        title: Text('Бали: $currentScore / $maxScore',
            style: const TextStyle(fontSize: 22, fontWeight:
FontWeight.bold)),
        flexibleSpace: Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(
                begin: Alignment.topCenter,
                end: Alignment.bottomCenter,
                colors: <Color>[Color(0xff7ac972),
Color(0xff9ad594)]),
          ),
        ),
        leading: IconButton(
            icon: const Icon(Icons.house, color: Colors.black),
            onPressed: () {

SystemChrome.setPreferredOrientations([DeviceOrientation.portrai
tUp]);
              Navigator.of(context).popUntil((route) =>
route.isFirst);
              },
            ),
        actions: [
          IconButton(
            tooltip: 'Змінити орієнтацію',
            icon: Icon(isLandscape
                ? Icons.screen_lock_landscape
                : Icons.screen_rotation),
            onPressed: () {
              isLandscape
                  ? SystemChrome.setPreferredOrientations(
                  [DeviceOrientation.landscapeLeft])
                  : SystemChrome.setPreferredOrientations(
                  [DeviceOrientation.landscapeLeft,
DeviceOrientation.landscapeRight,
DeviceOrientation.portraitUp]);
              setState(() {
                isLandscape = !isLandscape;
              });
            },
          )
        ],
      ),
      body: Container(
        margin: const EdgeInsets.all(10),
        child: Row(
          children: [
            SizedBox(
              width: 150,
```

```
              child: ListView.builder(
                controller: verticalTextController,
                itemCount: themes.length,
                itemBuilder: (context, i) {
                  return ThemeTile(title: themes[i].title);
                },
              ),
            ),
            Expanded(
              child: SingleChildScrollView(
                scrollDirection: Axis.horizontal,
                child: SizedBox(
                  width: maxQuestions * 124,
                  child: ListView.builder(
                    controller: verticalController,
                    itemCount: themes.length,
                    itemBuilder: (context, i) {
                      final theme = themes[i];
                      return Row(
                        children: List.generate(maxQuestions,
(j) {
                          if (j < theme.questions.length) {
                            final q = theme.questions[j];
                            return QuestionCell(
                              question: q,
                              onAnswered: () => handleAnswer(q),
                            );
                          } else {
                            return const EmptyCell();
                          }
                        }),
                      );
                    },
                  ),
                ),
              ),
            ],
          ),
        ),
      );
  }
}

//
// МОДЕЛІ
//

class ThemeBlock {
  final String title;
  final List<Question> questions;

  ThemeBlock({required this.title, required this.questions});
```

```
}

class Question {
  final String text;
  final String correctAnswer;
  final int price;
  bool answered = false;

  Question({
    required this.text,
    required this.correctAnswer,
    required this.price,
  });
}

//
// ПАРСЕР
//

List<ThemeBlock> parseSimpleQuiz(String content) {
  final lines = content.split('\n').map((e) =>
e.trim()).toList();

  if (lines.isEmpty || !lines[0].startsWith('#QUIZ_FILE_V1')) {
    throw Exception('Файл не підтримується.');
  }

  final themeCountLine = lines.firstWhere((l) =>
l.startsWith('themes='));
  final questionCountLine = lines.firstWhere((l) =>
l.startsWith('questions='));

  int themeCount = int.tryParse(themeCountLine.split('=')[1]) ??
0;
  int questionCount =
int.tryParse(questionCountLine.split('=')[1]) ?? 0;

  List<ThemeBlock> themes = [];
  int i = 0;

  while (i < lines.length) {
    final line = lines[i];

    // Розпізнавання теми
    if (line.startsWith('[Тема:')) {
      if (!line.endsWith(']')) {
        throw Exception('Помилка: тема на рядку ${i + 1} не має
закритих дужок.');
      }
      final themeTitle = line.substring(6, line.length -
1).trim(); // без [Тема: ...]
      List<Question> questions = [];
      i++;
```

```
      while (i < lines.length && !lines[i].startsWith('[Тема:'))
{
        // Питання
        if (lines[i].startsWith('{Питання:')) {
          final questionLine = lines[i];
          if (!questionLine.endsWith('}')) {
            throw Exception('Помилка: питання на рядку ${i + 1}
не має закритих дужок.');
          }
          final questionText = questionLine.substring(9,
questionLine.length - 1).trim();
          i++;

          // Відповідь
          if (i >= lines.length || !lines[i].startsWith('=')) {
            throw Exception('Відсутня відповідь після питання:
$questionText');
          }
          String correctAnswer = lines[i].substring(1).trim();
          i++;

          // Ціна
          int price = 0;
          if (i < lines.length && lines[i].startsWith('\$')) {
            price = int.tryParse(lines[i].substring(1)) ?? 0;
            i++;
          }

          questions.add(Question(
            text: questionText,
            correctAnswer: correctAnswer,
            price: price,
          ));
        } else {
          i++;
        }
      }

      themes.add(ThemeBlock(title: themeTitle, questions:
questions));
    } else {
      i++;
    }
  }

  // Перевірка відповідностей
  if (themes.length != themeCount) {
    throw Exception('Кількість тем не відповідає заявленій.
Знайдено ${themes.length}, очікувалось $themeCount.');
  }

  int actualQuestions = themes.fold(0, (sum, t) => sum +
```

```
t.questions.length);
  if (actualQuestions != questionCount) {
    throw Exception('Кількість питань не відповідає заявленій.
Знайдено $actualQuestions, очікувалось $questionCount.');
  }

  return themes;
}



//
// ВІДЖЕТИ
//

class ThemeTile extends StatelessWidget {
  final String title;

  const ThemeTile({super.key, required this.title});

  @override
  Widget build(BuildContext context) {
    return Container(
      height: 84,
      alignment: Alignment.centerLeft,
      padding: const EdgeInsets.all(8),
      decoration: BoxDecoration(
        border: Border(
          bottom: BorderSide(color: Colors.grey.shade300),
        ),
      ),
      child: AutoSizeText(
        title,
        style: const TextStyle(fontWeight: FontWeight.bold,
fontSize: 16),
        maxLines: 9,
        minFontSize: 6,
        overflow: TextOverflow.ellipsis,
      ),
    );
  }
}

class QuestionCell extends StatelessWidget {
  final Question question;
  final VoidCallback onAnswered;
  final TextEditingController _controller =
TextEditingController();

  QuestionCell({super.key, required this.question, required
this.onAnswered});

  @override
```

```
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: question.answered
          ? null
          : () {
        showDialog(
          barrierDismissible: false,
          context: context,
          builder: (_) => AlertDialog(
            title: const Text('Питання'),
            content: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Text(question.text),
                const SizedBox(height: 12),
                // Text('Правильна відповідь:
${question.correctAnswer}'),
                // const SizedBox(height: 8),
                // Text('Ціна: ${question.price}'),
                Builder(builder: (context){
                  // var height =
MediaQuery.of(context).size.height;
                  var width = MediaQuery.of(context).size.width;

                  return SizedBox(
                    // height: height + 100,
                    width: width,
                    child:
                    TextField(
                      controller: _controller,
                      onChanged: (text){ans =
text.toLowerCase().replaceAll(' ', '');},
                      onSubmitted: (text){ans =
text.toLowerCase().replaceAll(' ', '');},
                      decoration: InputDecoration(hintText:
question.correctAnswer, hintStyle:TextStyle(color:
Color(0x55000000)) ),
                    ),
                  );
                }),
              ],
            ),
            actions: [
              TextButton(
                onPressed: () {
                  onAnswered();
                },
                child: const Text('Відповісти'),
                style: ButtonStyle(backgroundColor:
WidgetStatePropertyAll<Color>(Colors.green), foregroundColor:
WidgetStatePropertyAll<Color>(Colors.white)),
              ),
```

```
            ],
          ),
        );
      },
      child: Container(
        width: 120,
        height: 80,
        margin: const EdgeInsets.all(2),
        alignment: Alignment.center,
        decoration: BoxDecoration(
          color: question.answered ? Colors.grey :
Colors.lightBlueAccent,
          borderRadius: BorderRadius.circular(5.0),
        ),
        child: Text(
          '${question.price}',
          style: TextStyle(fontWeight: FontWeight.bold,
fontSize: 16, color: question.answered ? Colors.white24 :
Colors.black),
        ),
      ),
    );
  }
}

class EmptyCell extends StatelessWidget {
  const EmptyCell({super.key});

  @override
  Widget build(BuildContext context) {
    return Container(
      width: 120,
      height: 80,
      margin: const EdgeInsets.all(2),
      decoration: BoxDecoration(
        color: const Color(0x11000000),
        borderRadius: BorderRadius.circular(5.0),
      ),
    );
  }
}
```

Б.4 question_editor_screen.dart

```
import 'dart:io';

import 'package:auto_size_text/auto_size_text.dart';
import 'package:file_picker/file_picker.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

class QuestionEditorScreen extends StatefulWidget {
  final int themeCount;
```

```
  final int maxQuestionsPerTheme;

  const QuestionEditorScreen({
    super.key,
    required this.themeCount,
    required this.maxQuestionsPerTheme,
  });

  @override
  State<QuestionEditorScreen> createState() =>
_QuestionEditorScreenState();
}

class _QuestionEditorScreenState extends
State<QuestionEditorScreen> {
  late List<EditableTheme> themes;
  bool isLandscape = false;
  final ScrollController verticalController =
ScrollController();
  final ScrollController verticalTextController =
ScrollController();
  String nameFile = 'default';

  @override
  void initState() {
    super.initState();
    themes = List.generate(widget.themeCount, (i) {
      return EditableTheme(
        title: 'Тема ${i + 1}',
        questions: List.generate(widget.maxQuestionsPerTheme,
(_) => EditableQuestion()),
      );
    });
    verticalController.addListener(() {
      if (verticalTextController.hasClients &&
          verticalController.offset !=
verticalTextController.offset) {

verticalTextController.jumpTo(verticalController.offset);
      }
    });
    verticalTextController.addListener(() {
      if (verticalController.hasClients &&
          verticalTextController.offset !=
verticalController.offset) {

verticalController.jumpTo(verticalTextController.offset);
      }
    });
  }

  @override
  void dispose() {
```

```
    verticalController.dispose();
    verticalTextController.dispose();
    super.dispose();
  }

  Future<void> _exportToTxt() async {
    final buffer = StringBuffer();
    int totalQuestions = themes.fold(0, (sum, t) => sum +
t.questions.where((q) => q.text.isNotEmpty).length);

    buffer.writeln('#QUIZ_FILE_V1');
    buffer.writeln('themes=${themes.length}');
    buffer.writeln('questions=$totalQuestions');
    buffer.writeln('');

    for (var theme in themes) {
      buffer.writeln('[Тема: ${theme.title}]');
      for (var q in theme.questions) {
        if (q.text.trim().isNotEmpty) {
          buffer.writeln('{Питання: ${q.text}}');
          buffer.writeln('= ${q.answer}');
          buffer.writeln('\$${q.price}');
          buffer.writeln('');
        }
      }
    }

    // Clipboard.setData(ClipboardData(text:
buffer.toString()));
    // ScaffoldMessenger.of(context).showSnackBar(
    //   const SnackBar(content: Text('Текст скопійовано в
буфер')),
    // );
    String? outputPath = await
FilePicker.platform.getDirectoryPath();

    if (outputPath != null) {
      final file = File('$outputPath/$nameFile.txt');
      await file.writeAsString(buffer.toString());
      //print('Файл збережено у: ${file.path}');
      showDialog(context: context, barrierDismissible: false,
builder: (context) => AlertDialog(
        title: Text('По шляху: $outputPath/$nameFile.txt'),
        content: Center(heightFactor: 0.4, child: Text('Файл
було успішно збережено!', style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.green, fontSize: 20))),
      )
      );
      Future.delayed(Duration(seconds: 10), () {

        Navigator.pop(context);
        Navigator.pop(context);
        setState(() {});
```

```
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: const Icon(Icons.house, color: Colors.black),
          onPressed: () {

SystemChrome.setPreferredOrientations([DeviceOrientation.portrai
tUp]);
            Navigator.of(context).popUntil((route) =>
route.isFirst);
          },
        ),
        title: const Text('Редактор набору питань'),
        flexibleSpace: Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(
                begin: Alignment.topCenter,
                end: Alignment.bottomCenter,
                colors: <Color>[Color(0xff7ac972),
Color(0xff9ad594)]),
          ),
        ),
        actions: [
          IconButton(
            icon: const Icon(Icons.save),
            tooltip: 'Зберег як TXT',
            onPressed: (){
              showDialog(context: context, builder: (context) =>
AlertDialog(
                title: Text('Введіть назву файлу:'),
                content:
                    TextField(
                      decoration: const
InputDecoration(labelText: 'Назва:'),
                      onChanged: (val) => nameFile = val,
                    ),
                actions: [
                  TextButton(
                    onPressed: _exportToTxt,
                    child: Text('Куди зберігати?'),
                  )
                ],
              ));
            }
          ),
          IconButton(
            tooltip: 'Змінити орієнтацію',
```

```
            icon: Icon(isLandscape
                ? Icons.screen_lock_landscape
                : Icons.screen_rotation),
            onPressed: () {
              isLandscape
                  ? SystemChrome.setPreferredOrientations(
                  [DeviceOrientation.landscapeLeft])
                  : SystemChrome.setPreferredOrientations(
                  [DeviceOrientation.landscapeLeft,
DeviceOrientation.landscapeRight,
DeviceOrientation.portraitUp]);
              setState(() {
                isLandscape = !isLandscape;
              });
            },
          )
        ],
      ),
      body: Row(
        children: [
          SizedBox(
            width: 150,
            child: ListView.builder(
              controller: verticalTextController,
              itemCount: themes.length,
              itemBuilder: (context, i) => ListTile(
                title: Container(
                  height: 184,
                  //color: Color(0x11000000),
                  child: Column(
                    children: [
                      TextField(
                        decoration: const
InputDecoration(labelText: 'Назва теми'),
                        controller: TextEditingController(text:
themes[i].title),
                        onChanged: (val) => themes[i].title =
val,
                        onSubmitted: (val) => setState(() {}),
                      ),
                      AutoSizeText(

                        themes[i].title,
                        style: const TextStyle(fontWeight:
FontWeight.bold, fontSize: 16),
                        maxLines: 15,
                        minFontSize: 6,
                        overflow: TextOverflow.ellipsis,
                      ),
                    ],
                  ),
                ),
              ),
```

```
            ),
          ),
          Expanded(
            child: SingleChildScrollView(
              scrollDirection: Axis.horizontal,
              child: SizedBox(
                width: widget.maxQuestionsPerTheme * 250,
                child: ListView.builder(
                  controller: verticalController,
                  itemCount: themes.length,
                  itemBuilder: (context, i) {
                    return Row(
                      children:
List.generate(widget.maxQuestionsPerTheme, (j) {
                        final question = themes[i].questions[j];
                        return SizedBox(
                          width: 240,
                          height: 200,
                          child: Card(
                            margin: const EdgeInsets.all(6),
                            child: Padding(
                              padding: const EdgeInsets.all(8),
                              child: Column(
                                children: [
                                  Text('Питання ${j + 1}'),
                                  TextField(
                                    decoration: const
InputDecoration(labelText: 'Текст питання'),
                                    onChanged: (val) =>
question.text = val,
                                  ),
                                  TextField(
                                    decoration: const
InputDecoration(labelText: 'Правильна відповідь'),
                                    onChanged: (val) =>
question.answer = val,
                                  ),
                                  TextField(
                                    decoration: const
InputDecoration(labelText: 'Ціна'),
                                    keyboardType:
TextInputType.number,
                                    onChanged: (val) =>
question.price = int.tryParse(val) ?? 0,
                                  ),
                                ],
                              ),
                            ),
                          ),
                        );
                      }),
                    );
                  },
```

```
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}

class EditableTheme {
  String title;
  List<EditableQuestion> questions;

  EditableTheme({required this.title, required this.questions});
}

class EditableQuestion {
  String text = '';
  String answer = '';
  int price = 0;
}
```