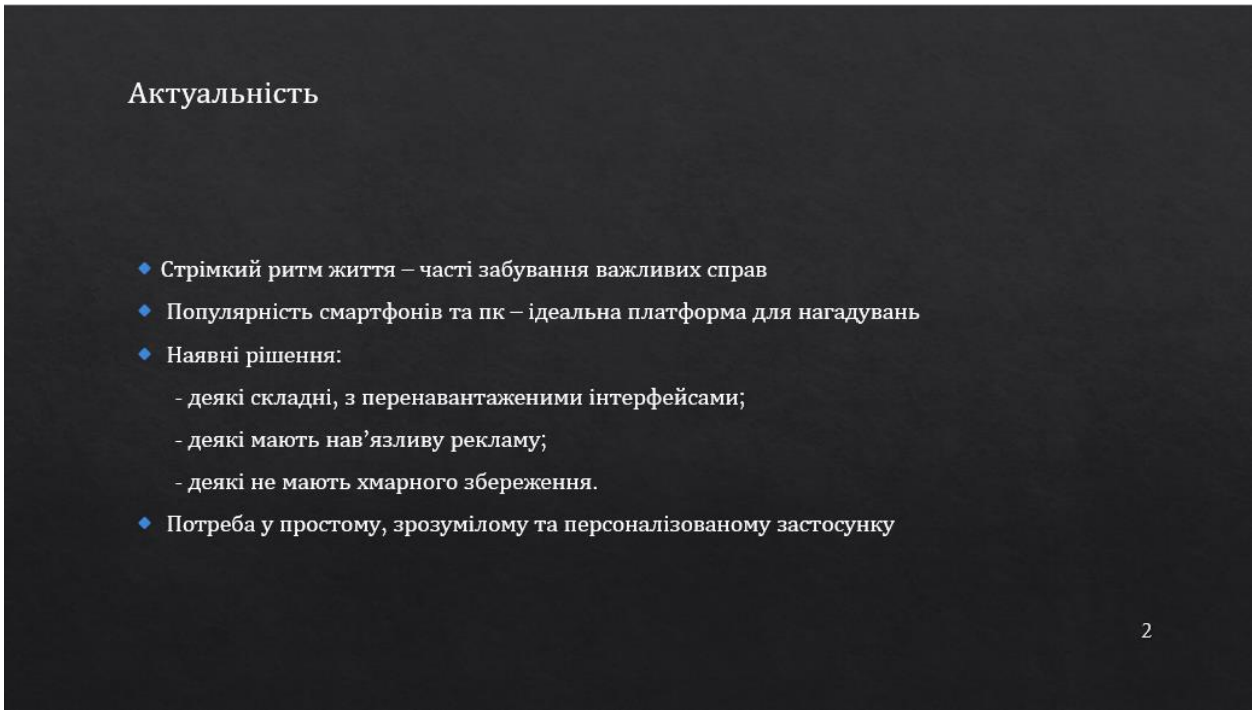
ДОДАТОК А
ГРАФІЧНИЙ МАТЕРІАЛ КВАЛІФІКАЦІЙНОЇ РОБОТИ



Мета та завдання

Мета:
Розробити персоналізований веб та мобільний застосунок нагадувань, який дозволяє користувачам зручно створювати, зберігати та отримувати сповіщення у заданий час.

Завдання:

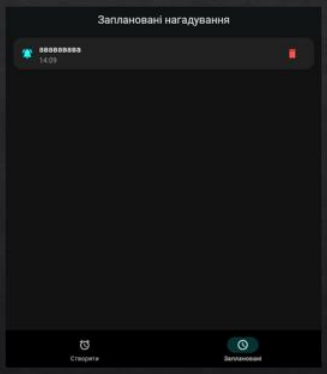
- Реалізувати веб-версію із базовим функціоналом нагадувань
- Створити мобільну версію з розширеними можливостями
- Забезпечити авторизацію користувача (email, Google)
- Використати хмарне збереження даних (Firebase Firestore)
- Додати підтримку повторюваних сповіщень
- Забезпечити інтуїтивно зрозумілий, адаптивний інтерфейс

3

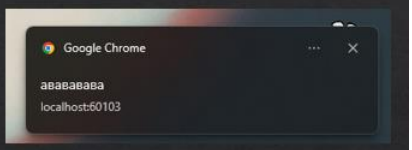
Веб-застосунок

Панель створення нагадувань

Панель запланованих нагадувань



Сповіщення



4

Аналіз наявних рішень

BZ Reminder

+ Має простий інтуїтивно зрозумілий інтерфейс
+ Має систему створення нагадувань на дні народження
- Немає хмарного збереження

5

Аналіз наявних рішень

Alarm & Pill Reminder

+ Великий функціонал
- Перевантажений інтерфейс
- Постійна реклама
- Затримки або відсутність сповіщень

6

Обрані технології

Dart

Flutter

Firebase

Firestore

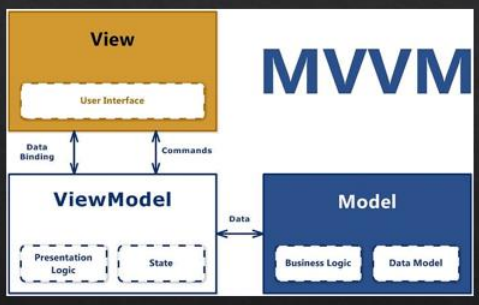
7

Архітектура застосунку

Для розробки було застосовано архітектурний патерн: MVVM

MVVM (Model – View – ViewModel);
Model – структура даних, моделі нагадувань;
View – інтерфейс користувача (екрани, віджети);
ViewModel – логіка обробки введення, збереження, взаємодії з Firebase.

Перевагами є чітке розділення відповідальностей, зручне масштабування та рефакторинг.



8

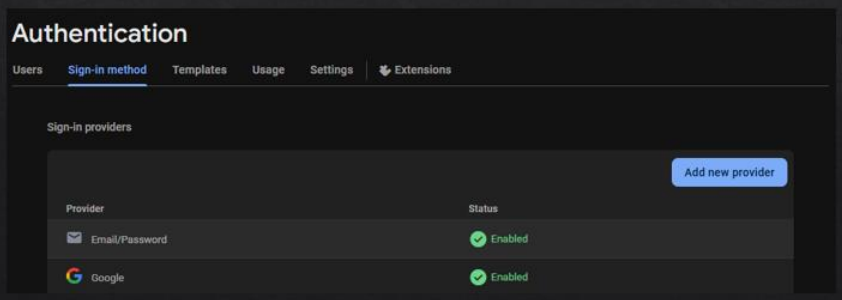
Використання Firebase

Firebase Authentication – авторизація через email чи Google вхід

Cloud Firestore – збереження нагадувань у хмарі

Синхронізація в реальному часі

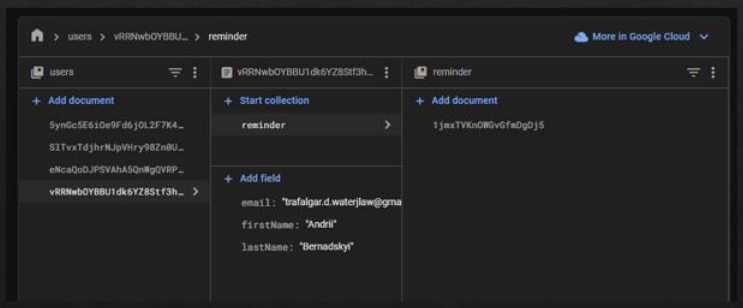
Підтримка офлайн-доступу



9

Модель даних у Firestore

```
class ReminderModel {
  Timestamp? timestamp;
  bool? onOff;
  String? message;
  DateTime? date;
  bool? isRepeating;
```



10

Функціонал застосунку

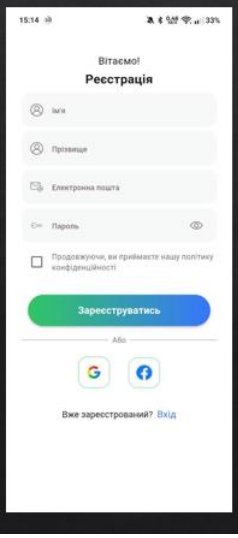
Застосунок включає:

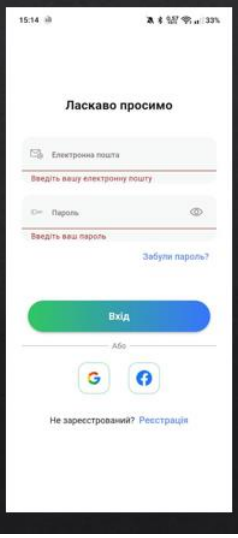
- Створення персональних нагадувань
- Повторювані нагадування (щоденні)
- Увімкнення / вимкнення сповіщень
- Редагування та видалення нагадувань
- Авторизація через Email або Google
- Хмарне збереження у Firestore
- Підтримка свят та щоденних фактів
- Інтуїтивно зрозумілий інтерфейс
- Робота у фоновому режимі
- Вихід з акаунту, зміна пароля, профіль

11

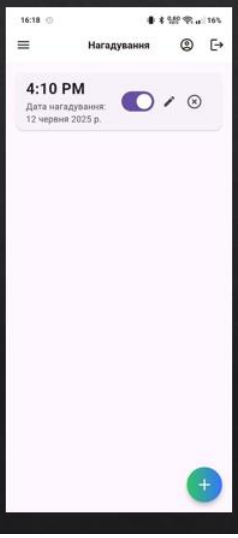
Інтерфейс користувача

Автентифікація

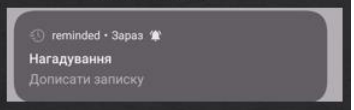




Головна сторінка



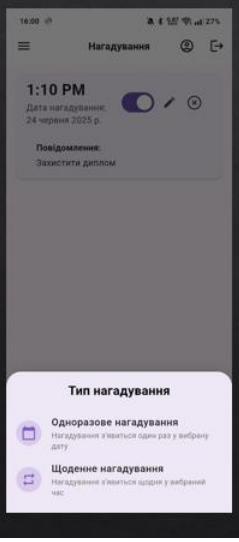
Сповіщення

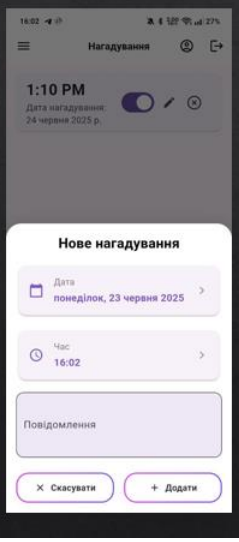


12

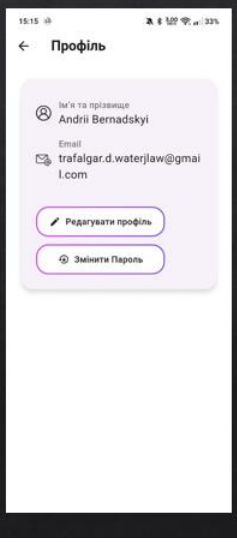
Інтерфейс користувача

Створення нагадувань

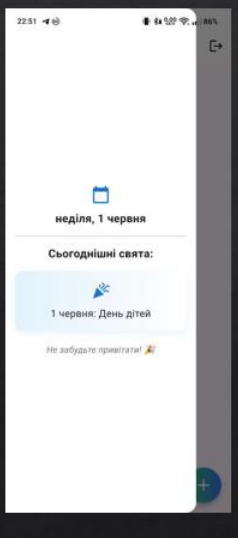




Сторінка профілю



Сторінка святкових подій



13

Висновки

- Реалізовано веб та мобільну версію застосунку нагадувань
- Зроблено акцент на мобільний застосунок — більш зручний у використанні
- Застосовано сучасні технології: Flutter, Firebase
- Здійснено авторизацію та хмарне збереження даних
- Інтерфейс простий, інтуїтивний, адаптований під щоденне використання
- Застосунок вирішує реальні проблеми користувачів

14

ДОДАТОК Б
REMINDER

Б.1 Верифікація користувача

Б.1.1 Код входу користувача через Google

```
Future<void> _signInWithGoogle() async {
  try {
    await GoogleSignIn().signOut();

    final GoogleSignInAccount? googleUser = await
GoogleSignIn().signIn();
    if (googleUser == null) return;

    final GoogleSignInAuthentication googleAuth = await
googleUser.authentication;

    final credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    UserCredential userCredential = await
FirebaseAuth.instance.signInWithCredential(credential);
    User? user = userCredential.user;

    if (user != null) {
      String uid = user.uid;
      String? email = user.email;
      final userDoc =
FirebaseFirestore.instance.collection('users').doc(uid);
      final snapshot = await userDoc.get();

      if (!snapshot.exists) {
        await userDoc.set({
          'email': email ?? '',
          'firstName': '',
          'lastName': '',
        });
      }

      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => HomeScreen()),
      );
```

```
    }
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Помилка входу через Google: $e")),
    );
  }
}
```

Б.1.2 Код входу користувача через пошту та пароль

```
Future<User?> _signIn(BuildContext context, String email, String
password) async {
  try {
    UserCredential userCredential = await
_auth.signInWithEmailAndPassword(
        email: email, password: password);
    User? user = userCredential.user;
    if (user != null && !user.emailVerified) {
      await user.sendEmailVerification();
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Будь ласка, підтвердіть вашу
електронну пошту.")),
      );
      return null;
    }
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Успішний вхід")),
    );
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => HomeScreen()),
    );
    return user;
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Неправильний логін чи пароль")),
    );
    return null;
  }
}
```

Б.1.3 Код реєстрації користувача

```
onPressed: () async{
  if(_formKey.currentState!.validate()){
    if(_isCheck){
      try{
        UserCredential userCredential =
```

```
        await _auth.createUserWithEmailAndPassword(
          email: _emailController.text,
          password: _passController.text,
        );

        await userCredential.user!.sendEmailVerification();

        String uid = userCredential.user!.uid;

        await _users.doc(uid).set({
          'email': _emailController.text,
          'firstName': _firstNameController.text,
          'lastName': _lastNameController.text,
        });

        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text("Перевірте вашу пошту для
підтвердження акаунта")),
        );
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) =>
LoginScreen()),
        );
        ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content: Text("Реєстрація успішна"))
        );
        Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => LoginScreen(),
            ));
      }catch(e){
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text(e.toString()))
        );
      }
    }
  }
}
```

Б.2 Сервіси додатку

Б.2.1 Код витягу свят з API

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:intl/intl.dart';
import 'package:translator/translator.dart';
```

```
Future<String> getHolidaysForToday() async {
  try {
    final now = DateTime.now();
    final formattedDate = DateFormat('yyyy-MM-dd').format(now);

    final url = Uri.parse(

'https://calendarific.com/api/v2/holidays?api_key=apIrJdecmjZ6th
3f7mjSwtZymGjajdwv&country=UA&year=${now.year}&month=${now.month
}&day=${now.day}');
    final response = await http.get(url);

    if (response.statusCode != 200) {
      throw Exception('Не вдалося завантажити свята');
    }

    final data = json.decode(response.body);
    final holidays = data['response']['holidays'] as
List<dynamic>;

    if (holidays.isEmpty) return 'На сьогодні немає свят.';

    final translator = GoogleTranslator();

    List<String> translatedNames = [];
    for (var holiday in holidays) {
      final name = holiday['name'];
      final translation = await translator.translate(name, to:
'uk');
      translatedNames.add(translation.text);
    }

    final holidayNames = translatedNames.join(', ');
    final formattedDateUkr = DateFormat('d MMMM',
'uk').format(now);

    return 'Свята на $formattedDateUkr: $holidayNames';
  } catch (e) {
    print('Помилка при завантаженні свят: $e');
    return 'Сталася помилка при отриманні свят.';
  }
}
```

Б.2.2 Код реалізації нотифікації

```
import 'package:flutter/material.dart';
import
'package:flutter_local_notifications/flutter_local_notifications
.dart';
import 'package:reminded/screens/home_screen.dart';
```

```
import 'package:rxdart/rxdart.dart';
import 'package:timezone/timezone.dart' as tz;
import 'package:timezone/data/latest.dart' as tz;

class NotificationLogic{
  static final _notifications =
FlutterLocalNotificationsPlugin();
  static final onNotifications = BehaviorSubject<String?>();

  static Future _notificationDetails() async {
    return NotificationDetails(
      android: AndroidNotificationDetails(
        "Нагадування!", "Не забувайте пити воду",
        importance: Importance.max, priority: Priority.max),
    );
  }
  static Future cancelNotification(int id) async {
    await _notifications.cancel(id);
  }
  static Future cancelAllNotifications() async {
    await _notifications.cancelAll();
  }
  static Future init(BuildContext context, String uid) async{
    tz.initializeTimeZones();
    final android = AndroidInitializationSettings("timing");
    final settings = InitializationSettings(android: android);
    await _notifications.initialize(settings,
        onDidReceiveNotificationResponse: (payLoad) {
      Navigator.pushReplacement(
          context,
          MaterialPageRoute(
            builder: (context) => HomeScreen(),
          )
      );
      onNotifications.add(payLoad as String?);
    });
  }

  static Future showNotifications({
    int id = 0,
    String? title,
    String? body,
    String? payLoad,
    required DateTime dateTime,
  }) async {
    if(dateTime.isBefore(DateTime.now())){
      dateTime = dateTime.add(Duration(days: 1));
    }
    await _notifications.zonedSchedule(
      id,
      title,
      body,
      tz.TZDateTime.from(dateTime, tz.local),
```

```
      await _notificationDetails(),
      uiLocalNotificationDateInterpretation:
UILocalNotificationDateInterpretation.absoluteTime,
      androidScheduleMode:
AndroidScheduleMode.exactAllowWhileIdle,
      matchDateTimeComponents: null,
      payload: payLoad,
    );
  }
  static Future showDailyNotification({
    int id = 0,
    String? title,
    String? body,
    String? payLoad,
    required TimeOfDay time,
  }) async {
    await _notifications.cancel(id);
    final now = DateTime.now();
    final firstTime = DateTime(now.year, now.month, now.day,
time.hour, time.minute);
    final scheduledDate = firstTime.isBefore(now) ?
firstTime.add(Duration(days: 1)) : firstTime;
    await _notifications.zonedSchedule(
      id,
      title,
      body,
      tz.TZDateTime.from(scheduledDate, tz.local),
      await _notificationDetails(),
      uiLocalNotificationDateInterpretation:
UILocalNotificationDateInterpretation.absoluteTime,
      androidScheduleMode:
AndroidScheduleMode.exactAllowWhileIdle,
      matchDateTimeComponents: DateTimeComponents.time,
      payload: payLoad,
    );
  }

}
```

Б.3 Віджети застосунку

Б.3.1 Код файлу add_reminder

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:intl/intl.dart';
import 'package:reminded/model/reminder_model.dart';
import 'package:reminded/utils/app_colors.dart';
```

```
const Color _primaryColor = Color(0xFF7E57C2);
const Color _surfaceColor = Color(0xFFEDE7F6);

void addReminder(BuildContext context, String uid,
TextEditingController controller) {
  showModalBottomSheet(
    context: context,
    backgroundColor: Colors.transparent,
    isScrollControlled: true,
    builder: (context) {
      return Container(
        decoration: BoxDecoration(
          color: _surfaceColor,
          borderRadius: BorderRadius.vertical(top:
Radius.circular(24)),
          boxShadow: [BoxShadow(color: Colors.black12,
blurRadius: 20)],
        ),
        padding: EdgeInsets.only(
          bottom: MediaQuery.of(context).viewInsets.bottom,
        ),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Padding(
              padding: const EdgeInsets.all(16),
              child: Text(
                'Тип нагадування',
                style: TextStyle(
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                  color: Colors.black,
                ),
              ),
            ),
            _buildReminderTypeTile(
              context: context,
              icon: Icons.calendar_today,
              title: 'Одноразове нагадування',
              subtitle: 'Нагадування з\'явиться один раз у
вибрану дату',
              onTap: () {
                Navigator.pop(context);
                _showReminderForm(context, uid, controller,
isRepeating: false);
              },
            ),
            _buildReminderTypeTile(
              context: context,
              icon: Icons.repeat,
              title: 'Щоденне нагадування',
              subtitle: 'Нагадування з\'явиться щодня у вибраний
```

```
час',
              onTap: () {
                Navigator.pop(context);
                _showReminderForm(context, uid, controller,
isRepeating: true);
              },
            ),
            SizedBox(height: 16),
          ],
        ),
      );
    },
  );
}

Widget _buildReminderTypeTile({
  required BuildContext context,
  required IconData icon,
  required String title,
  required String subtitle,
  required VoidCallback onTap,
}) {
  return ListTile(
    leading: Container(
      width: 40,
      height: 40,
      decoration: BoxDecoration(
        color: _primaryColor.withOpacity(0.2),
        shape: BoxShape.circle,
      ),
      child: Icon(icon, color: _primaryColor),
    ),
    title: Text(title, style: TextStyle(fontWeight:
FontWeight.w500)),
    subtitle: Text(subtitle, style: TextStyle(fontSize: 12,
color: Colors.grey[600])),
    onTap: onTap,
  );
}

void _showReminderForm(
    BuildContext context,
    String uid,
    TextEditingController controller, {
      required bool isRepeating,
    }) {
  TimeOfDay time = TimeOfDay.now();
  DateTime selectedDate = DateTime.now();

  void addReminder(String message) {
    final reminderDateTime = DateTime(
      selectedDate.year,
      selectedDate.month,
```

```
      selectedDate.day,
      time.hour,
      time.minute,
    );

    final reminder = ReminderModel(
      timestamp: Timestamp.fromDate(reminderDateTime),
      onOff: false,
      message: message,
      date: isRepeating ? null : selectedDate,
      isRepeating: isRepeating,
    );

    FirebaseFirestore.instance
        .collection('users')
        .doc(uid)
        .collection('reminder')
        .add(reminder.toMap());

    Fluttertoast.showToast(msg: "Нагадування створено");
  }

  showModalBottomSheet(
    context: context,
    backgroundColor: Colors.transparent,
    isScrollControlled: true,
    builder: (context) {
      return StatefulBuilder(
        builder: (context, setState) {
          return Container(
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.vertical(top:
Radius.circular(24)),
              boxShadow: [BoxShadow(color: Colors.black12,
blurRadius: 20)],
            ),
            padding: EdgeInsets.only(
              bottom: MediaQuery.of(context).viewInsets.bottom,
            ),
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Padding(
                  padding: const EdgeInsets.all(16),
                  child: Text(
                    'Нове нагадування',
                    style: TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: Colors.black,
                    ),
                  ),
```

```
                ),
                Padding(
                  padding: const
EdgeInsets.symmetric(horizontal: 16),
                  child: Column(
                    children: [
                      if (!isRepeating)
                        _buildDateTimePicker(
                          icon: Icons.calendar_today,
                          label: 'Дата',
                          value: DateFormat('EEEE, d MMMM y',
'uk').format(selectedDate),
                          onTap: () async {
                            final picked = await showDatePicker(
                              context: context,
                              initialDate: selectedDate,
                              firstDate: DateTime.now(),
                              lastDate: DateTime(2100),
                              builder: (context, child) =>
Theme(
                                data:
Theme.of(context).copyWith(
                                  colorScheme:
ColorScheme.light(
                                    primary: _primaryColor,
                                  ),
                                ),
                                child: child!,
                              ),
                            );
                            if (picked != null) setState(() =>
selectedDate = picked);
                          },
                        ),
                      SizedBox(height: 16),
                      _buildDateTimePicker(
                        icon: Icons.access_time,
                        label: 'Час',
                        value: time.format(context),
                        onTap: () async {
                          final picked = await showTimePicker(
                            context: context,
                            initialTime: time,
                            builder: (context, child) => Theme(
                              data: Theme.of(context).copyWith(
                                timePickerTheme:
TimePickerThemeData(
                                  backgroundColor: Colors.white,
                                  hourMinuteColor:
_surfaceColor,
                                  hourMinuteTextColor:
_primaryColor,
                                ),
```

```
                                ),
                                child: child!,
                              ),
                            );
                            if (picked != null) setState(() =>
time = picked);
                          },
                        ),
                        SizedBox(height: 16),
                        TextField(
                          controller: controller,
                          maxLines: 3,
                          decoration: InputDecoration(
                            labelText: 'Повідомлення',
                            floatingLabelStyle: TextStyle(color:
_primaryColor),
                            border: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(12),
                              borderSide: BorderSide(color:
Colors.grey[300]!),
                            ),
                            focusedBorder: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(12),
                              borderSide: BorderSide(color:
_primaryColor),
                            ),
                            filled: true,
                            fillColor: _surfaceColor,
                          ),
                        ),
                        SizedBox(height: 24),
                        Row(
                          children: [
                            Expanded(
                              child: GradientBorderButton(
                                onPressed: () =>
Navigator.pop(context),
                                text: 'Скасувати',
                                icon: Icons.close,
                              ),
                            ),
                            SizedBox(width: 16),
                            Expanded(
                              child: GradientBorderButton(
                                onPressed: () {
                                  final message =
controller.text.trim();
                                  if (message.isEmpty) {
                                    Fluttertoast.showToast(msg:
"Введіть повідомлення");
                                    return;
```

```
                                    }
                                    addReminder(message);
                                    Navigator.pop(context);
                                  },
                                  text: 'Додати',
                                  icon: Icons.add,
                                ),
                              ),
                            ],
                          ),
                          SizedBox(height: 16),
                        ],
                      ),
                    ),
                  ],
                ),
              );
            },
          );
        },
      );
    }

    Widget _buildDateTimePicker({
      required IconData icon,
      required String label,
      required String value,
      required VoidCallback onTap,
    }) {
      return Card(
        elevation: 2,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(12),
        ),
        child: InkWell(
          borderRadius: BorderRadius.circular(12),
          onTap: onTap,
          child: Padding(
            padding: const EdgeInsets.all(16),
            child: Row(
              children: [
                Icon(icon, color: _primaryColor),
                SizedBox(width: 16),
                Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      label,
                      style: TextStyle(
                        fontSize: 14,
                        color: Colors.grey[600],
                      ),
                    ),
```

```
                SizedBox(height: 4),
                Text(
                  value,
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.bold,
                    color: _primaryColor,
                  ),
                ),
              ],
            ),
            Spacer(),
            Icon(Icons.chevron_right, color: Colors.grey),
          ],
        ),
      ),
    ),
  );
}
```

Б.3.2 Код файлу delete_reminder

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:reminded/services/notification_logic.dart';

deleteReminder(BuildContext context, String id, String uid)
async {
  return showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(25),
        ),
        title: Text("Видалити нагадування"),
        content: Text("Ви дійсно хочете видалити?"),
        actions: [
          TextButton(
            onPressed: () async {
              try {

                final doc = await FirebaseFirestore.instance
                    .collection("users")
                    .doc(uid)
                    .collection("reminder")
                    .doc(id)
                    .get();
```

```
                await
NotificationLogic.cancelNotification(doc.id.hashCode);


                await doc.reference.delete();

                Fluttertoast.showToast(msg: "Нагадування
видалено");
              } catch (e) {
                Fluttertoast.showToast(msg: e.toString());
              }
              Navigator.pop(context);
            },
            child: Text("Видалити"),
          ),
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: Text("Відміна"),
          ),
        ],
      );
    },
  );
}
```

Б.3.3 Код файлу edit_reminder

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:intl/intl.dart';
import 'package:reminded/utils/app_colors.dart';

const Color _primaryColor = Color(0xFF7E57C2);
const Color _surfaceColor = Color(0xFFEDE7F6);

void editReminder(
    BuildContext context,
    String uid,
    String id,
    Timestamp initialTime,
    String initialMessage,
    bool isRepeating,
    ) {
  TimeOfDay time = TimeOfDay.fromDateTime(initialTime.toDate());
  TextEditingController controller = TextEditingController(text:
initialMessage);
  DateTime selectedDate = initialTime.toDate();

  showModalBottomSheet(
    context: context,
```

```
    backgroundColor: Colors.transparent,
    isScrollControlled: true,
    builder: (context) {
      return StatefulBuilder(
        builder: (context, setState) {
          return Container(
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.vertical(top:
Radius.circular(24)),
              boxShadow: [BoxShadow(color: Colors.black12,
blurRadius: 20)],
            ),
            padding: EdgeInsets.only(
              bottom: MediaQuery.of(context).viewInsets.bottom,
            ),
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                Padding(
                  padding: const EdgeInsets.all(16),
                  child: Text(
                    'Редагування нагадування',
                    style: TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: _primaryColor,
                    ),
                  ),
                ),
                Padding(
                  padding: const
EdgeInsets.symmetric(horizontal: 16),
                  child: Column(
                    children: [
                      if (!isRepeating)
                        _buildDateTimePicker(
                          icon: Icons.calendar_today,
                          label: 'Дата',
                          value: DateFormat('EEEE, d MMMM y',
'uk').format(selectedDate),
                          onTap: () async {
                            final picked = await showDatePicker(
                              context: context,
                              initialDate: selectedDate,
                              firstDate: DateTime.now(),
                              lastDate: DateTime(2100),
                              builder: (context, child) =>
Theme(
                                data:
Theme.of(context).copyWith(
                                  colorScheme:
ColorScheme.light(
```

```
                                    primary: _primaryColor,
                                  ),
                                ),
                                child: child!,
                              ),
                            );
                            if (picked != null) setState(() =>
selectedDate = picked);
                          },
                        ),
                    SizedBox(height: 16),
                    _buildDateTimePicker(
                      icon: Icons.access_time,
                      label: 'Час',
                      value: time.format(context),
                      onTap: () async {
                        final picked = await showTimePicker(
                          context: context,
                          initialTime: time,
                          builder: (context, child) => Theme(
                            data: Theme.of(context).copyWith(
                              timePickerTheme:
TimePickerThemeData(
                                backgroundColor: Colors.white,
                                hourMinuteColor:
_surfaceColor,
                                hourMinuteTextColor:
_primaryColor,
                              ),
                            ),
                            child: child!,
                          ),
                        );
                        if (picked != null) setState(() =>
time = picked);
                      },
                    ),
                    SizedBox(height: 16),
                    TextField(
                      controller: controller,
                      maxLines: 3,
                      decoration: InputDecoration(
                        labelText: 'Повідомлення',
                        floatingLabelStyle: TextStyle(color:
_primaryColor),
                        border: OutlineInputBorder(
                          borderRadius:
BorderRadius.circular(12),
                          borderSide: BorderSide(color:
Colors.grey[300]!),
                        ),
                        focusedBorder: OutlineInputBorder(
                          borderRadius:
```

```
BorderRadius.circular(12),
                              borderSide: BorderSide(color:
_primaryColor),
                            ),
                            filled: true,
                            fillColor: _surfaceColor,
                          ),
                        ),
                        SizedBox(height: 24),
                        Row(
                          children: [
                            Expanded(
                              child: GradientBorderButton(
                                onPressed: () =>
Navigator.pop(context),
                                text: 'Скасувати',
                                icon: Icons.close,
                              ),
                            ),
                            SizedBox(width: 16),
                            Expanded(
                              child: GradientBorderButton(
                                onPressed: () async {
                                  final message =
controller.text.trim();
                                  if (message.isEmpty) {
                                    Fluttertoast.showToast(msg:
"Введіть повідомлення");
                                    return;
                                  }

                                  final newDateTime = DateTime(
                                    selectedDate.year,
                                    selectedDate.month,
                                    selectedDate.day,
                                    time.hour,
                                    time.minute,
                                  );

                                  final Map<String, dynamic>
updatedData = {
                                    'time':
Timestamp.fromDate(newDateTime),
                                    'message': message,
                                  };

                                  if (!isRepeating) {
                                    updatedData['date'] =
Timestamp.fromDate(selectedDate);
                                  }

                                  await FirebaseFirestore.instance
                                      .collection('users')
```

```
                                        .doc(uid)
                                        .collection('reminder')
                                        .doc(id)
                                        .update(updatedData);

                                    Fluttertoast.showToast(msg:
"Нагадування оновлено");
                                    Navigator.pop(context);
                                  },
                                  text: 'Зберегти',
                                  icon: Icons.save,
                                ),
                              ),
                            ],
                          ),
                          SizedBox(height: 16),
                        ],
                      ),
                    ),
                  ],
                ),
              );
            },
          );
        },
      );
}

Widget _buildDateTimePicker({
  required IconData icon,
  required String label,
  required String value,
  required VoidCallback onTap,
}) {
  return Card(
    elevation: 2,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    child: InkWell(
      borderRadius: BorderRadius.circular(12),
      onTap: onTap,
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Row(
          children: [
            Icon(icon, color: _primaryColor),
            SizedBox(width: 16),
            Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  label,
```

```
                    style: TextStyle(
                      fontSize: 14,
                      color: Colors.grey[600],
                    ),
                  ),
                  SizedBox(height: 4),
                  Text(
                    value,
                    style: TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.bold,
                      color: _primaryColor,
                    ),
                  ),
                ],
              ),
              Spacer(),
              Icon(Icons.chevron_right, color: Colors.grey),
            ],
          ),
        ),
      ),
    );
  }
```