ДОДАТОК А

Лістинг коду

Kotlin

MainActivity.kt

```
package com.example.medsky

import android.os.Build
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.viewModels
import androidx.annotation.RequiresApi
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.tooling.preview.Preview
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.example.medsky.ui.theme.MedSkyTheme
import com.example.medsky.ui.theme.cart.CartScreen
import com.example.medsky.ui.theme.cart.CartViewModel
import com.example.medsky.ui.theme.catalog.CatalogScreen
import com.example.medsky.ui.theme.login.SignInScreen
import com.example.medsky.ui.theme.login.SignUpScreen
import com.example.medsky.ui.theme.order.OrderConfirmationScreen
import com.example.medsky.ui.theme.order.OrderStatusScreen
import com.example.medsky.ui.theme.userinfo.UserInfoScreen
```

```
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class MainActivity : ComponentActivity() {
    private val cartViewModel: CartViewModel by viewModels()
    private lateinit var locationService: LocationService

    @RequiresApi(Build.VERSION_CODES.UPSIDE_DOWN_CAKE)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        locationService = LocationService(this)
        setContent {
            MedSkyTheme {
                Surface(color = MaterialTheme.colorScheme.background) {
                    AuthenticationApp(cartViewModel, locationService)
                }
            }
        }
    }
}

@RequiresApi(Build.VERSION_CODES.UPSIDE_DOWN_CAKE)
@Composable
fun AuthenticationApp(cartViewModel: CartViewModel, locationService:
LocationService) {
    val navController = rememberNavController()
    NavHost(navController, startDestination = "signin") {
        composable("signin") { SignInScreen(navController) }
        composable("signup") { SignUpScreen(navController) }
```

```
        composable("catalog") { CatalogScreen(navController, cartViewModel) }
        composable("cart") { CartScreen(navController, cartViewModel) }
        composable("userinfo") { UserInfoScreen(navController, cartViewModel, locationService) }
        composable("orderstatus") { OrderStatusScreen(navController, cartViewModel) }
        composable("orderconfirmation/{orderId}") { backStackEntry ->
            val orderId = backStackEntry.arguments?.getString("orderId") ?: ""
            OrderConfirmationScreen(navController, orderId)
        }
    }
}

@RequiresApi(Build.VERSION_CODES.UPSIDE_DOWN_CAKE)
@Composable
fun DefaultPreview() {
    MedSkyTheme {
        val cartViewModel = CartViewModel() // Використовуйте мок або preview instance для попереднього перегляду
        val context = androidx.compose.ui.platform.LocalContext.current
        val locationService = LocationService(context) // Передайте контекст у LocationService
        AuthenticationApp(cartViewModel, locationService)
    }
}
```

SignInScreen.kt

```
package com.example.medsky.ui.theme.login

import android.util.Log
import androidx.compose.foundation.background
```

```
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.google.firebase.auth.FirebaseAuth
import com.example.medsky.ui.theme.catalog.CatalogScreen

@Composable
fun SignInScreen(navController: NavController) {
    val context = LocalContext.current
    val auth = FirebaseAuth.getInstance()
    val emailState = remember { mutableStateOf("") }
    val passwordState = remember { mutableStateOf("") }
    val showError = remember { mutableStateOf(false) }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(Color(0xFFF0F8FF)),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
```

```
    ) {
        Text(
            text = "MedSky",
            fontSize = 32.sp,
            color = Color(0xFF00796B)
        )
        Spacer(modifier = Modifier.height(20.dp))

        OutlinedTextField(
            value = emailState.value,
            onValueChange = { emailState.value = it },
            label = { Text("Email") },
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Email),
            modifier = Modifier.fillMaxWidth(0.8f)
        )
        Spacer(modifier = Modifier.height(10.dp))

        OutlinedTextField(
            value = passwordState.value,
            onValueChange = { passwordState.value = it },
            label = { Text("Password") },
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Password),
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth(0.8f)
        )
        Spacer(modifier = Modifier.height(20.dp))

        Button(
            onClick = {
```

```
                signIn(auth, emailState.value, passwordState.value, navController, showError)
            },
            colors = ButtonDefaults.buttonColors(containerColor = Color(0xFF00796B)),
            modifier = Modifier
                .fillMaxWidth(0.8f)
                .height(50.dp)
        ) {
            Text(text = "Sign In", color = Color.White)
        }
        Spacer(modifier = Modifier.height(10.dp))

        TextButton(onClick = { navController.navigate("signup") }) {
            Text(text = "Don't have an account? Sign Up")
        }

        if (showError.value) {
            Snackbar(
                modifier = Modifier.padding(16.dp)
            ) {
                Text(text = "Sign In failed. Please try again.")
            }
        }
    }
}

private fun signIn(auth: FirebaseAuth, email: String, password: String, navController:
NavController, showError: MutableState<Boolean>) {
    auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
```

```
                Log.d("MyLog", "Sign In successful!")
                navController.navigate("catalog")
            } else {
                Log.d("MyLog", "Sign In failure: ${task.exception?.message}")
                showError.value = true
            }
        }
}
```

SignUpScreen.kt

```
package com.example.medsky.ui.theme.login

import android.util.Log
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.google.firebase.auth.FirebaseAuth
import com.example.medsky.ui.theme.catalog.CatalogScreen
```

```
@Composable
fun SignUpScreen(navController: NavController) {
    val context = LocalContext.current
    val auth = FirebaseAuth.getInstance()
    val emailState = remember { mutableStateOf("") }
    val passwordState = remember { mutableStateOf("") }
    val showError = remember { mutableStateOf(false) }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(Color(0xFFF0F8FF)),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            text = "MedSky",
            fontSize = 32.sp,
            color = Color(0xFF00796B)
        )
        Spacer(modifier = Modifier.height(20.dp))

        OutlinedTextField(
            value = emailState.value,
            onValueChange = { emailState.value = it },
            label = { Text("Email") },
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Email),
            modifier = Modifier.fillMaxWidth(0.8f)
        )
        Spacer(modifier = Modifier.height(10.dp))
```

```
        OutlinedTextField(
            value = passwordState.value,
            onValueChange = { passwordState.value = it },
            label = { Text("Password") },
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Password),
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth(0.8f)
        )
        Spacer(modifier = Modifier.height(20.dp))

        Button(
            onClick = {
                signUp(auth, emailState.value, passwordState.value, navController, showError)
            },
            colors = ButtonDefaults.buttonColors(containerColor = Color(0xFF00796B)),
            modifier = Modifier
                .fillMaxWidth(0.8f)
                .height(50.dp)
        ) {
            Text(text = "Sign Up", color = Color.White)
        }
        Spacer(modifier = Modifier.height(10.dp))

        TextButton(onClick = { navController.navigate("signin") }) {
            Text(text = "Already have an account? Sign In")
        }
```

```
        if (showError.value) {
            Snackbar(
                modifier = Modifier.padding(16.dp)
            ) {
                Text(text = "Sign Up failed. Please try again.")
            }
        }
    }
}

private fun signUp(auth: FirebaseAuth, email: String, password: String, navController:
NavController, showError: MutableState<Boolean>) {
    auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                Log.d("MyLog", "Sign Up successful!")
                navController.navigate("catalog")
            } else {
                Log.d("MyLog", "Sign Up failure: ${task.exception?.message}")
                showError.value = true
            }
        }
}
```

CartScreen

```
package com.example.medsky.ui.theme.cart

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
```

```
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.example.medsky.ui.theme.userinterface.Medication
import androidx.compose.ui.Alignment.Companion.CenterHorizontally
import androidx.compose.ui.graphics.Color

@Composable
fun CartScreen(navController: NavController, cartViewModel: CartViewModel) {
    val cartItems = cartViewModel.cartItems

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
            .background(MaterialTheme.colorScheme.background),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Top
    ) {
        Text(
            text = "Кошик",
            fontSize = 24.sp,
            color = MaterialTheme.colorScheme.primary,
            modifier = Modifier.padding(10.dp)
        )
```

```
        Spacer(modifier = Modifier.height(10.dp))

        if (cartItems.isEmpty()) {
            Text(text = "Кошик порожній")
        } else {
            LazyColumn {
                items(cartItems) { cartItem ->
                    CartItem(
                        cartItem = cartItem,
                        onRemoveClicked                                =                                {
cartViewModel.removeFromCart(cartItem.medication) }
                    )
                }
            }
        }

        Spacer(modifier = Modifier.height(20.dp))

        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp),
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Button(
                onClick = {
                    cartViewModel.clearCart()
                    navController.popBackStack()
                },
```

```
                colors = ButtonDefaults.buttonColors(
                    containerColor = Color.Blue,
                    contentColor = Color.White)
            ) {
                Text(text = "Очистити кошик")
            }

            Button(
                onClick = {
                    navController.navigate("userinfo")
                },
                colors = ButtonDefaults.buttonColors(
                    containerColor = Color.Blue,
                    contentColor = Color.White)
            ) {
                Text(text = "Продовжити")
            }
        }
    }
}

@Composable
fun CartItem(
    cartItem: CartItem,
    onRemoveClicked: () -> Unit
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 4.dp),
```

```
        colors = CardDefaults.cardColors(
            containerColor = MaterialTheme.colorScheme.surface
        )
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Text(text = cartItem.medication.name, style =
MaterialTheme.typography.titleMedium)
            Text(text = cartItem.medication.description, style =
MaterialTheme.typography.bodySmall)
            Text(text = "Кількість: ${cartItem.quantity}", style =
MaterialTheme.typography.bodySmall)

            Spacer(modifier = Modifier.height(8.dp))

            Button(
                onClick = onRemoveClicked,
                modifier = Modifier.align(Alignment.End),
                colors = ButtonDefaults.buttonColors(
                    containerColor = MaterialTheme.colorScheme.error,
                    contentColor = Color.White
                )
            ) {
                Text(text = "Видалити")
            }
        }
    }
}
```

CartViewModel.kt

```
package com.example.medsky.ui.theme.cart

import android.content.ContentValues.TAG
import android.util.Log
import androidx.compose.runtime.mutableStateListOf
import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel
import com.example.medsky.ui.theme.userinterface.Medication
import com.google.firebase.firestore.FirebaseFirestore
import androidx.compose.runtime.State

class CartViewModel : ViewModel() {
    private val _userInfo = mutableStateOf<Map<String, String>?>(null)
    val userInfo: State<Map<String, String>?> = _userInfo
    val cartItems = mutableStateListOf<CartItem>()

    fun addToCart(medication: Medication) {
        val existingItem = cartItems.find { it.medication.name == medication.name }
        if (existingItem != null) {
            existingItem.quantity += 1
        } else {
            cartItems.add(CartItem(medication, 1))
        }
    }

    fun removeFromCart(medication: Medication) {
        val itemToRemove = cartItems.find { it.medication.name == medication.name }
        if (itemToRemove != null && itemToRemove.quantity > 1) {
            itemToRemove.quantity -= 1
        } else {
```

```
            cartItems.remove(itemToRemove)
        }
    }

    fun clearCart() {
        cartItems.clear()
    }

    private val firestore = FirebaseFirestore.getInstance()
    private val usersCollection = firestore.collection("users")

    fun saveUserInfo(
        name: String,
        address: String,
        phone: String,
        latitude: String,
        longitude: String
    ) {
        val user = hashMapOf(
            "name" to name,
            "address" to address,
            "phone" to phone,
            "latitude" to latitude,
            "longitude" to longitude
        )
        _userInfo.value = user

        usersCollection
            .add(user)
            .addOnSuccessListener { documentReference ->
```

```
            val userId = documentReference.id
            Log.d(TAG, "Інформацію про користувача успішно збережено з ID: $userId")
        }
        .addOnFailureListener { e ->
            Log.e(TAG, "Помилка збереження інформації про користувача", e)
        }
    }
}

data class CartItem(
    val medication: Medication,
    var quantity: Int
)
```

CatalogScreen.kt

```
package com.example.medsky.ui.theme.catalog

import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
```

```
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import coil.compose.rememberImagePainter
import com.example.medsky.ui.theme.userinterface.Medication
import com.example.medsky.ui.theme.userinterface.Screen
import com.example.medsky.ui.theme.cart.CartViewModel
import com.google.firebase.firestore.FirebaseFirestore
import kotlinx.coroutines.tasks.await

@Composable
fun CatalogScreen(navController: NavController, cartViewModel: CartViewModel) {
    val firestore = remember { FirebaseFirestore.getInstance() }
    var medications by remember { mutableStateOf(listOf<Medication>()) }

    LaunchedEffect(Unit) {
        try {
            val snapshot = firestore.collection("medications").get().await()
            val meds = snapshot.documents.map { doc ->
                doc.toObject(Medication::class.java)!!
            }
            medications = meds
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
```

```
        .background(MaterialTheme.colorScheme.background),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Top
) {
    Text(
        text = "Каталог медикаментів",
        style = MaterialTheme.typography.headlineMedium,
        color = MaterialTheme.colorScheme.primary,
        modifier = Modifier.padding(10.dp)
    )
    Spacer(modifier = Modifier.height(10.dp))

    LazyColumn {
        items(medications) { medication ->
            MedicationItem(medication = medication, onAddToCartClicked = {
                cartViewModel.addToCart(medication)
            })
            Spacer(modifier = Modifier.height(16.dp))
        }
    }

    Spacer(modifier = Modifier.height(20.dp))

    Button(
        onClick = { navController.navigate(Screen.Cart.route) },
        modifier = Modifier.align(Alignment.CenterHorizontally),
        colors = ButtonDefaults.buttonColors(
            containerColor = Color.Blue,
            contentColor = Color.White
        )
```

```
        ) {
            Text(text = "Перейти до кошика")
        }
    }
}

@Composable
fun MedicationItem(medication: Medication, onAddToCartClicked: (Medication) ->
Unit) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .height(180.dp)
            .padding(vertical = 8.dp),
        colors = CardDefaults.cardColors(
            containerColor = MaterialTheme.colorScheme.surface
        )
    ) {
        Row(modifier = Modifier.padding(16.dp)) {
            Image(
                painter = rememberImagePainter(
                    data = medication.imageUrl,
                    builder = {
                        crossfade(true)
                    }
                ),
                contentDescription = null,
                modifier = Modifier
                    .size(120.dp)
                    .align(Alignment.CenterVertically)
```

```
                .clip(MaterialTheme.shapes.medium),
            contentScale = ContentScale.Crop
        )
        Spacer(modifier = Modifier.width(16.dp))
        Column(
            modifier = Modifier
                .weight(1f)
                .align(Alignment.CenterVertically)
        ) {
            Text(text = medication.name, style = MaterialTheme.typography.titleMedium)
            Text(text = medication.description, style =
MaterialTheme.typography.bodySmall)
            Spacer(modifier = Modifier.height(8.dp))
            Button(
                onClick = { onAddToCartClicked(medication) },
                modifier = Modifier.align(Alignment.End),
                colors = ButtonDefaults.buttonColors(
                    containerColor = Color.Blue,
                    contentColor = Color.White
                )
            ) {
                Text(text = "Додати до кошика")
            }
        }
    }
}
```

Medication.kt

```
package com.example.medsky.ui.theme.userinterface

data class Medication(
    val id: String = "",
    val name: String = "",
    val description: String = "",
    val quantity: Int = 0,
    val imageUrl: String = ""
) {
    override fun equals(other: Any?): Boolean {
        return if (this === other) {
            true
        } else if (other !is Medication) {
            false
        } else {
            id == other.id
        }
    }

    override fun hashCode(): Int {
        return id.hashCode()
    }
}
```

OrderConfiramtionScreen.kt

```
package com.example.medsky.ui.theme.order

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
```

```
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.google.firebase.firestore.FirebaseFirestore

@Composable
fun OrderConfirmationScreen(navController: NavController, orderId: String) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Дякуємо за ваше замовлення!",
            fontSize = 24.sp,
            color = MaterialTheme.colorScheme.primary,
            modifier = Modifier.padding(10.dp)
        )

        Spacer(modifier = Modifier.height(16.dp))

        Button(
            onClick = {
```

```
                // Видалення замовлення з Firebase
                FirebaseFirestore.getInstance().collection("orders").document(orderId)
                    .delete()
                    .addOnSuccessListener {
                        navController.popBackStack("catalog", false)
                    }
                    .addOnFailureListener { e ->
                        // Обробка помилок під час видалення
                    }
            },
            modifier = Modifier.align(Alignment.CenterHorizontally),
            colors = ButtonDefaults.buttonColors(
                containerColor = Color.Blue,
                contentColor = Color.White
            )
        ) {
            Text(text = "Підтвердити замовлення")
        }
    }
}
```

OrderStatusScreen.kt

```
package com.example.medsky.ui.theme.order

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
```

```
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.google.firebase.firestore.FirebaseFirestore

@Composable
fun OrderConfirmationScreen(navController: NavController, orderId: String) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "Дякуємо за ваше замовлення!",
            fontSize = 24.sp,
            color = MaterialTheme.colorScheme.primary,
            modifier = Modifier.padding(10.dp)
        )

        Spacer(modifier = Modifier.height(16.dp))

        Button(
            onClick = {
                // Видалення замовлення з Firebase
                FirebaseFirestore.getInstance().collection("orders").document(orderId)
                    .delete()
                    .addOnSuccessListener {
```

```
                    navController.popBackStack("catalog", false)
                }
                .addOnFailureListener { e ->
                    // Обробка помилок під час видалення
                }
            },
            modifier = Modifier.align(Alignment.CenterHorizontally),
            colors = ButtonDefaults.buttonColors(
                containerColor = Color.Blue,
                contentColor = Color.White
            )
        ) {
            Text(text = "Підтвердити замовлення")
        }
    }
}
```

Screen.kt

```
package com.example.medsky.ui.theme.userinterface

sealed class Screen(val route: String) {
    object SignIn : Screen("signin")
    object SignUp : Screen("signup")
    object Catalog : Screen("catalog")
    object Cart : Screen("cart")
}
```

UserInfoScreen.kt

```
package com.example.medsky.ui.theme.userinfo

import android.os.Build
import androidx.annotation.RequiresApi
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.example.medsky.LocationService
import com.example.medsky.ui.theme.cart.CartViewModel

@RequiresApi(Build.VERSION_CODES.UPSIDE_DOWN_CAKE)
@Composable
fun UserInfoScreen(navController: NavController, cartViewModel: CartViewModel,
locationService: LocationService) {
    var name by remember { mutableStateOf("") }
    var address by remember { mutableStateOf("") }
    var phone by remember { mutableStateOf("") }
    var latitude by remember { mutableStateOf("") }
    var longitude by remember { mutableStateOf("") }

    Column(
        modifier = Modifier
            .fillMaxSize()
```

```
            .padding(16.dp)
            .background(MaterialTheme.colorScheme.background),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Top
    ) {
        Text(
            text = "Заповніть інформацію про себе",
            fontSize = 24.sp,
            color = MaterialTheme.colorScheme.primary,
            modifier = Modifier.padding(10.dp)
        )

        Spacer(modifier = Modifier.height(10.dp))

        TextField(
            value = name,
            onValueChange = { name = it },
            label = { Text("Ім'я") },
            modifier = Modifier
                .fillMaxWidth()
                .padding(vertical = 8.dp)
        )

        Spacer(modifier = Modifier.height(10.dp))

        TextField(
            value = address,
            onValueChange = { address = it },
            label = { Text("Адреса") },
            modifier = Modifier
```

```
        .fillMaxWidth()
        .padding(vertical = 8.dp)
)

Spacer(modifier = Modifier.height(10.dp))

TextField(
    value = phone,
    onValueChange = { phone = it },
    label = { Text("Телефон") },
    modifier = Modifier
        .fillMaxWidth()
        .padding(vertical = 8.dp)
)

Spacer(modifier = Modifier.height(10.dp))

Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceBetween
) {
    TextField(
        value = latitude,
        onValueChange = { latitude = it },
        label = { Text("Широта") },
        modifier = Modifier
            .fillMaxWidth(0.5f)
            .padding(vertical = 8.dp)
    )
```

```
            TextField(
                value = longitude,
                onValueChange = { longitude = it },
                label = { Text("Довгота") },
                modifier = Modifier
                    .fillMaxWidth(0.5f)
                    .padding(vertical = 8.dp)
            )
        }

        Spacer(modifier = Modifier.height(20.dp))

        Button(
            onClick = {
                // Оновлюємо дані користувача у ViewModel
                cartViewModel.saveUserInfo(name, address, phone, latitude, longitude)

                // Перехід на екран статусу замовлення
                navController.navigate("orderstatus")
            },
            modifier = Modifier.align(Alignment.CenterHorizontally),
            colors = ButtonDefaults.buttonColors(
                containerColor = Color.Blue,
                contentColor = Color.White)
        ) {
            Text(text = "Продовжити")
        }
    }
}
```

LocationService.kt

```
package com.example.medsky

import android.Manifest
import android.content.Context
import android.content.pm.PackageManager
import android.location.Location
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices

class LocationService(private val context: Context) {
    private val fusedLocationClient: FusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(context)

    fun getLastLocation(onSuccess: (Location) -> Unit, onFailure: () -> Unit) {
        if (ContextCompat.checkSelfPermission(context,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            fusedLocationClient.lastLocation
                .addOnSuccessListener { location ->
                    if (location != null) {
                        onSuccess(location)
                    } else {
                        onFailure()
                    }
                }
                .addOnFailureListener { exception ->
                    onFailure()
                    // Обробка помилок
```

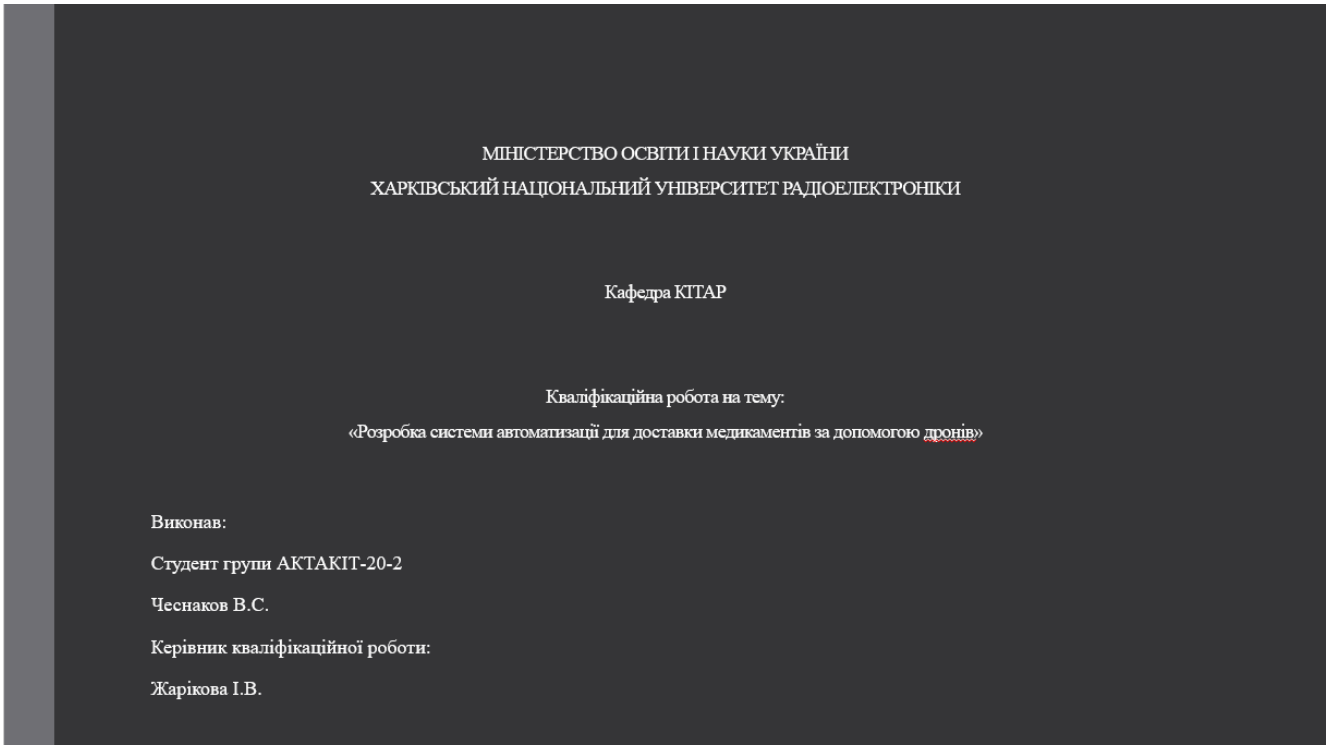
```
                exception.printStackTrace()
            }
        } else {
            onFailure()
        }
    }
}
```

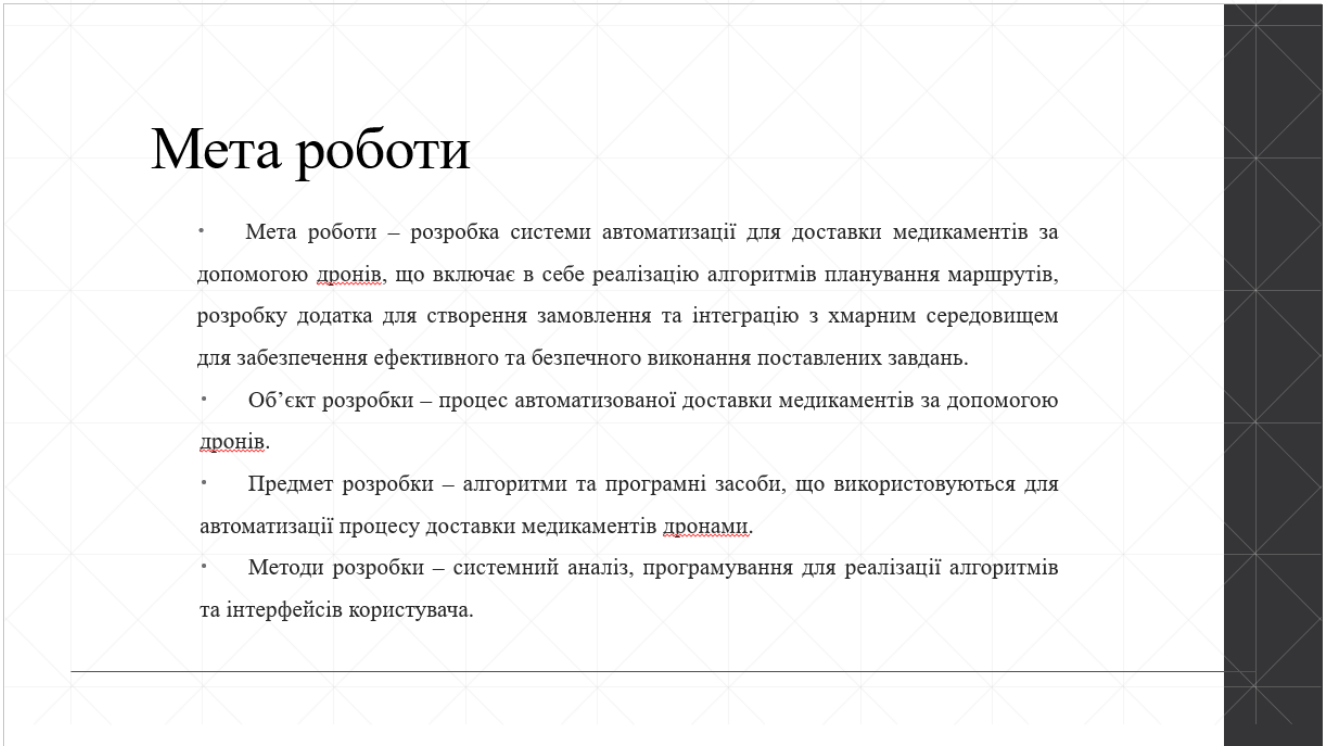
MyApp.kt

```
package com.example.medsky

import android.app.Application
import dagger.hilt.android.HiltAndroidApp

@HiltAndroidApp
class MyApp : Application()
```

ДОДАТОК Б

Демонстраційний матеріал





ПОСТАВЛЕНІ ЗАДАЧІ РОБОТИ

Для досягнення поставленої мети необхідно вирішити такі завдання:

- аналіз вимог та функціональних можливостей системи автоматизованої доставки медикаментів;
- аналіз алгоритмів автоматичного планування маршрутів для дронів;
- розробка додатку для автоматизованої системи доставки;
- вибір платформи та реалізація алгоритму планування маршруту для автоматизованої системи доставки;

АНАЛІЗ ВИМОГ ТА ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ АВТОМАТИЗОВАНОЇ ДОСТАВКИ МЕДИКАМЕНТІВ

Робота в умовах обмеженого доступу

Стійкість до впливу факторів бойової обстановки

Умови забезпечення швидкої реакції та доставки

Шляхи забезпечення конфіденційності системи доставки медикаментів

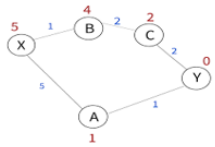
Робота в умовах екстремальних температур та кліматичних умовах

Запас енергії та тривалість польоту

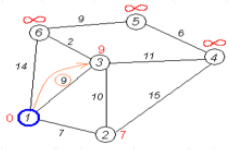
Можливість використання відео- та тепловізійної зйомки

АНАЛІЗ АЛГОРИТМІВ АВТОМАТИЧНОГО ПЛАНУВАННЯ МАРШРУТІВ

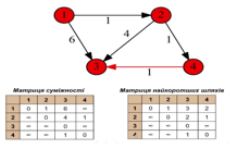
Алгоритм A* (A star)



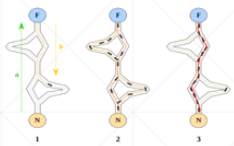
Алгоритм Дейкстри



Алгоритм Флойда-Уоршелла



Мурашиний алгоритм



РОЗРОБКА ДОДАТКУ ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДОСТАВКИ

Мова програмування

Kotlin

Платформа

android studio

Хмарне середовище

Firebase

ВИБІР ПЛАТФОРМИ ТА РЕАЛІЗАЦІЯ АЛГОРИТМУ ПЛАНУВАННЯ МАРШРУТУ ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДОСТАВКИ

Огляд технічних характеристик

- дальність його польоту може досягати 15 км, вона дозволяє робити доставку в межах міста та в передмістя;
- максимальне навантаження – до 2,7 кг, цього цілком вистачить для доставки медикаментів;
- автономність до 55 хв польоту, чого цілком вистачає для більшості завдань доставки в межах міста;
- система навігації має високу точність;
- наявність сенсорів уникнення перешкод, функції повернення додому. Наявні сенсори забезпечують безпеку польоту;
- легка система управління, інтелектуальне планування маршрутів;
- стійкість до погодних умов.

Реалізація алгоритму Дейкстри

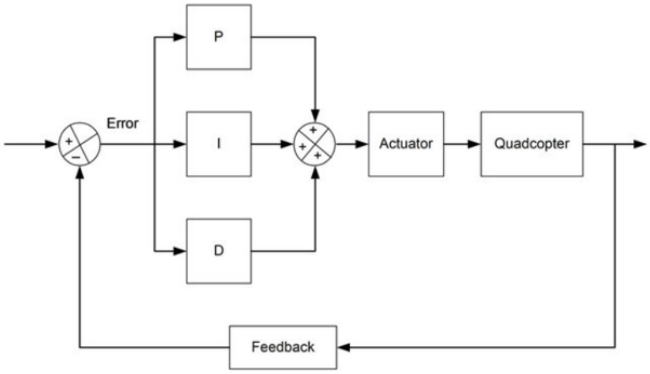
Приклад роботи алгоритму

```
Граф після ініціалізації:
{'A': {'B': 2, 'C': 3, 'G': 12}, 'B': {'A': 2, 'D': 4, 'E': 5, 'H': 13}, 'C': {'A': 3, 'D': 6, 'I': 14}, 'D': {'B': 4, '
C': 6, 'E': 7}, 'E': {'B': 5, 'D': 7, 'F': 8}, 'F': {'E': 8, 'I': 11, 'G': 15}, 'G': {'H': 9, 'A': 12, 'F': 15}, 'H': {'
G': 9, 'I': 10, 'B': 13}, 'I': {'H': 10, 'F': 11, 'C': 14}}
Введіть початкову точку: A
Введіть кінцеву точку: I
Дрон рухається з A до A.
Маршрут: C -> I
Дрон рухається з A до C.
Маршрут: I
Дрон рухається з C до I.
Маршрут: завершено.
Граф після переміщення дрона:
{'A': {'B': 2, 'C': 3, 'G': 12}, 'B': {'A': 2, 'D': 4, 'E': 5, 'H': 13}, 'C': {'A': 3, 'D': 6, 'I': 14}, 'D': {'B': 4, '
C': 6, 'E': 7}, 'E': {'B': 5, 'D': 7, 'F': 8}, 'F': {'E': 8, 'I': 11, 'G': 15}, 'G': {'H': 9, 'A': 12, 'F': 15}, 'H': {'
G': 9, 'I': 10, 'B': 13}, 'I': {'H': 10, 'F': 11, 'C': 14}}
```

Система автоматизованого управління для дрона DJI Matrice 300 RTK

- сенсори
- керуючі елементи
- алгоритми керування
- алгоритми планування маршруту та уникнення перешкод

Опис пропорційно-інтегрально-диференціального (ПІД) регулятора



ВИСНОВКИ

У результаті аналізу різних джерел інформації за темою кваліфікаційної роботи було розглянуто основні вимоги та функціональні можливості до автоматизованої системи доставки медикаментів. Система повинна забезпечувати стабільну та ефективну роботу у важкодоступних місцях. Дрони повинні мати гарну автономність, а також мати вбудовану систему навігацію, а також протидіяти радіоелектронним та іншим видам перешкод. Повинні мати конструкцію, яка буде стійкою до фізичних пошкоджень. Мати достатній запас енергії для забезпечення довготривалого польоту. І на останок бути готовими працювати в різних кліматичних умовах.

Проведено аналіз існуючих алгоритмів автоматичного планування маршруту, було проаналізовано такі алгоритми, як: A* (A-star), алгоритм Дейкстри, алгоритм Флойда-Уоршела та мурашиний алгоритм. Було прийняте рішення, щодо використання алгоритму Дейкстри в системі. Цей алгоритм має гарну швидкість та дозволяє оптимально планувати маршрути для дронів, завдяки можливості знаходити найкоротший шлях. Його інтеграція в систему забезпечить ефективну доставку до місця призначення, враховуючи оптимізацію маршрутів.

Створено додаток для користувача та оператора, для зручності взаємодії із системою. Для створення було використано середовище Android Studio, МП Kotlin, адже ця мова надає змогу писати код компактніше та виразніше, що прискорює процес розробки. Також він сприяє збільшенню продуктивності додатків, завдяки статичній типізації та функціональним можливостям. Також було обране хмарне середовище Firebase, воно легко інтегрується в проекти та дозволяє розширити їх функціонал.

Здійснено вибір платформи для системи. Обраний дрон DJI Matrice 300 RTK, він має вбудовану систему навігації, що дозволяє інтегрувати алгоритм автоматичного планування маршруту. Описано систему автоматизованого управління. А також було здійснено опис ПІД-регулятора, який використовується для стабілізації польоту.

Розроблений додаток та інтегрований алгоритм забезпечать ефективну та надійну роботу системи автоматизованої доставки, що є дуже важливим для медичних потреб.

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК.505500.021	Пояснювальна записка	А4, 56с.
		Текст програми	31 с.
		Додаткові матеріали	
2		Демонстраційний матеріал	ppt,12 с.
		Відео роботи програми	1 шт

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК.505500.021		
Розроб.		Чеснаков В. С.			Розробка системи автоматизації для доставки медикаментів за допомогою дронів	Літера	Аркуш	Аркушів
Перевір.		Жарікова І. В.				Н	1	1
						Кафедра КІТАМ ХНУРЕ		
Н.контр.		Стародубцев М.Г						
Затв.		Невлюдов І. Ш.						