Assignment Report

Title of the assignment: Instagram Clone Development

Student's Name: Serzhan Yerasil 23MD0434

Date of Submission: 2024-10-19

Table of Contents

Project Setup

The project was set up using Android Studio, with Kotlin as the main programming language. The Jetpack Compose framework was used to build the app's user interface. Libraries such as Coil were integrated for image loading, and Navigation Compose was used for managing page

```
implementation("androidx.compose.ui:ui:1.5.0")
implementation("androidx.compose.material:material:1.5.0")
implementation("androidx.compose.ui:ui-tooling-preview:1.5.0")
implementation("androidx.navigation:navigation-compose:2.7.1")
implementation("io.coil-kt:coil-compose:2.1.0")
```

androidx.compose.ui:ui:1.5.0

This library is the core part of the Jetpack Compose user interface. It provides the basic components for creating interfaces using a declarative approach. Contains elements such as text, buttons, columns, rows and other basic components.

androidx.compose.material:material:1.5.0

This library provides Material Design components for Jetpack Compose. Includes elements such as app bar, cards, screen structure, bottom navigation, notifications and others that follow Material Design guidelines to create a consistent user experience.

androidx.compose.ui:ui-tooling-preview:1.5.0

This library allows you to use the preview feature in Android Studio. It allows you to see what components will look like without running the application on the device, and includes tools for debugging and analyzing compositions.

androidx.navigation:navigation-compose:2.7.1
This library provides navigation support specifically for Jetpack Compose. It allows you to implement navigation within an application using NavHost and NavController components in a declarative style, simplifying transitions between pages or components.

io.coil-kt:coil-compose:2.1.0
This is an extension of the Coil library (which is used to load images) for Jetpack Compose. It provides convenience components such as rememberImagePainter or rememberAsyncImagePainter for loading and displaying images from URLs and other sources directly in Compose interfaces.

Project Design

Home feed

```
@Composable
fun HomeFeedPage(navController: NavController) {
    val posts = listOf(
        Post( id: 1,  username: "user1",  imageUrl: "https://i.postimg.cc/SKMQyHd7/squid.jpg",  caption: "Hell
        Post( id: 2,  username: "user2",  imageUrl: "https://image2.jpg",  caption: "My second post!",  likes: 90
    )

    LazyColumn {
        items(posts) { post ->
            Card(modifier = Modifier.padding(8.dp)) {
                Column {
                    Text(text = post.username, fontWeight = FontWeight.Bold)
                    Image(
                        painter = rememberImagePainter(post.imageUrl),
                        contentDescription = null,
                        modifier = Modifier.fillMaxWidth().height(200.dp)
                    )
                    Text(text = post.caption)
                    Text(text = "${post.likes} likes", fontWeight = FontWeight.Light)
                }
            }
        }
    }
}
```

This HomeFeedPage feature displays a feed of posts in the application using a LazyColumn. Each post is represented by a Card containing the username, image, description and number of likes. Images are loaded by URL using the Coil library (rememberImagePainter). The function accepts a NavController to navigate between screens, but this example does not use navigation.

Profile page

```
@Composable
fun ProfilePage(navController: NavController) {
    val profile = UserProfile( username: "user1",  profilePictureUrl: "https://profile-pic.jpg",  bio: "Bio here",  postCount: 12)
    val userPosts = listOf(
        Post( id: 1,  username: "user1",  imageUrl: "https://i.postimg.cc/FHqsXhyH/spong-patrick.png",  caption: "Caption 1",  likes: 1200),
        Post( id: 2,  username: "user1",  imageUrl: "https://i.postimg.cc/SKMQyHd7/squid.jpg",  caption: "Caption 2",  likes: 87)
    )


    Column {
        Row(modifier = Modifier.padding(16.dp)) {
            Image(
                painter = rememberAsyncImagePainter(profile.profilePictureUrl),
                contentDescription = null,
                modifier = Modifier.size(80.dp).clip(CircleShape)
            )
            Spacer(modifier = Modifier.width(16.dp))
            Column {
                Text(text = profile.username, fontWeight = FontWeight.Bold)
                Text(text = profile.bio)
                Text(text = "Posts: ${profile.postCount}")
            }
        }
        LazyVerticalGrid(
            columns = GridCells.Fixed( count: 3),
            content = {
                items(List.size) { index ->

                    Image(
                        painter = rememberImagePainter(List[index]),
                        contentDescription = null,
                        modifier = Modifier.size(100.dp)
                    )
                }
            }
        )
    }
}
```

This ProfilePage feature displays a user's profile, including their photo, name, description, and number of posts. At the top of the page there is information about the user, presented in a line with an avatar, name and biography. Under this information there is a grid (LazyVerticalGrid) with images of user posts, where the Coil library (rememberImagePainter) is used for each image. The function accepts a NavController to navigate between screens, but the current code does not implement navigation.

Search page

```
@Composable
fun SearchPage(navController: NavController) {
    var searchQuery by remember { mutableStateOf( value: "") }
    val results = listOf("user1", "user2", "user3")

    Column {
        TextField(
            value = searchQuery,
            onValueChange = { searchQuery = it },
            placeholder = { Text( text: "Search") },
            modifier = Modifier.fillMaxWidth().padding(16.dp)
        )
        LazyColumn {
            items(results.filter { it.contains(searchQuery, ignoreCase = true) }) { result ->
                Text(text = result, modifier = Modifier.padding(16.dp))
            }
        }
    }
}
```

This SearchPage function implements a user search page. At the top is a text field (TextField) for entering a search query. When the text changes, the results list is updated. The results are displayed using a vertical list (LazyColumn), where each element is the name of the found user. The logic for filtering the results list is based on comparing the user's input with the names in the list. The function accepts a NavController for possible navigation between screens.

```
@Composable
fun AddPostPage(navController: NavController) {
    var caption by remember { mutableStateOf( value: "") }

    Column(modifier = Modifier.padding(16.dp)) {
        Button(onClick = { /* Handle Image Picker */ }) {
            Text( text: "Pick an Image")
        }
        Spacer(modifier = Modifier.height(16.dp))
        TextField(
            value = caption,
            onValueChange = { caption = it },
            placeholder = { Text( text: "Add a caption") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = { /* Handle Post Upload */ }) {
```

Add post page

```
@Composable
fun AddPostPage(navController: NavController) {
    var caption by remember { mutableStateOf( value: "") }

    Column(modifier = Modifier.padding(16.dp)) {
        Button(onClick = { /* Handle Image Picker */ }) {
            Text( text: "Pick an Image")
        }
        Spacer(modifier = Modifier.height(16.dp))
        TextField(
            value = caption,
            onValueChange = { caption = it },
            placeholder = { Text( text: "Add a caption") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = { /* Handle Post Upload */ }) {
            Text( text: "Upload Post")
        }
    }
}
```

This AddPostPage function implements the page for adding a new post. The user interface contains a button for selecting an image, which can call the functionality for opening a gallery.
A text input field (TextField) for adding a caption to the image.
A button for uploading a post, which sends the selected image and caption to the server or to local storage.
The interface is built using Column for vertical placement of elements. The function accepts a NavController for possible navigation between screens.

This NotificationsPage function displays a page with notifications. Notifications are presented as a vertical list (LazyColumn). Each list item contains a text description of the notification, such as information about likes or comments on the user's posts. Notification data is passed to the list and displayed in plain text format. The function accepts a NavController for possible navigation between screens, but navigation is not used in the current code.

Notifications Page

```
@Composable
fun NotificationsPage(navController: NavController) {
    val notifications = listOf(
        Notification( id: 1,  content: "User1 liked your post"),
        Notification( id: 2,  content: "User2 commented: Nice post!")
    )

    LazyColumn {
        items(notifications) { notification ->
            Text(text = notification.content, modifier = Modifier.padding(16.dp))
        }
    }
}
```

This NotificationsPage function displays a page with notifications. Notifications are presented as a vertical list (LazyColumn). Each list item contains a text description of the notification, such as information about likes or comments on the user's posts. Notification data is passed to the list and displayed in plain text format. The function accepts a NavController for possible navigation between screens, but navigation is not used in the current code.

Navigation

```
@Composable
fun AppNavigation() {
    val navController = rememberNavController()

    Scaffold(
        bottomBar = { BottomNavigationBar(navController = navController) }
    ) { contentPadding ->
        NavHost(
            navController = navController,
            startDestination = Screen.HomeFeed.route,
            modifier = Modifier.padding(contentPadding)
        ) {
            composable(Screen.HomeFeed.route) { HomeFeedPage(navController) }
            composable(Screen.Search.route) { SearchPage(navController) }
            composable(Screen.AddPost.route) { AddPostPage(navController) }
            composable(Screen.Notifications.route) { NotificationsPage(navController) }
            composable(Screen.Profile.route) { ProfilePage(navController) }
        }
    }
}
```

Navigation in the application is implemented using the NavHost component and the Navigation Compose library. The AppNavigation function manages transitions between the main pages of the application (Home, Profile, Search, Add Post and Notifications). Inside NavHost, the start screen and routes for each page are defined. Transitions are carried out using NavController, which allows switching between screens using unique routes. For user convenience, the application can implement a bottom navigation bar (BottomNavigationBar) for quick switching between pages.

User Interaction

```
@Composable
fun BottomNavigationBar(navController: NavController) {
    val items = listOf(
        Screen.HomeFeed,
        Screen.Search,
        Screen.AddPost,
        Screen.Notifications,
        Screen.Profile
    )

    BottomNavigation {
        items.forEach { screen ->
            BottomNavigationItem(
                icon = {
                    when (screen) {
                        Screen.HomeFeed -> Icon(Icons.Default.Home, contentDescription = "Home")
                        Screen.Search -> Icon(Icons.Default.Search, contentDescription = "Search")
                        Screen.AddPost -> Icon(Icons.Default.Add, contentDescription = "Add Post")
                        Screen.Notifications -> Icon(Icons.Default.Notifications, contentDescription = "Notifications")
                        Screen.Profile -> Icon(Icons.Default.Person, contentDescription = "Profile")
                    }
                },
                selected = false,
                onClick = {
                    navController.navigate(screen.route) {

                        popUpTo(navController.graph.startDestinationId) {
                            saveState = true
                        }
                        launchSingleTop = true
                        restoreState = true
                    }
                }
            )
        }
    }
}
```

User interactions in the app are implemented using click and input handlers. For example:

Item clicks: Each post, profile, or notification can have click actions configured, such as going to the post or profile details. This is achieved using the onClick function in components such as Card or Image.

Input: TextFields allow the user to enter information, such as when adding a caption to a post on the add page. Changes are tracked using the onValueChange function.

Buttons: Buttons, such as selecting an image or submitting a post, are configured using the Button component, where the corresponding actions are called when clicked.

Result

ProfilePage.kt | HomeFeedPage.kt | build.gradle.kts (:app) | SearchPage.kt

```
import androidx.compose.material3.Text
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import coil.compose.rememberImagePainter
import com.example.assignment2.Post


@Composable
fun HomeFeedPage(navController: NavController) {
    val posts = listOf(
        Post( id: 1, username: "Yerasil", imageUrl: "https://i.postimg.cc/FHqsXhyH/spong-patrick.png", captio
        Post( id: 2, username: "Azamat", imageUrl: "https://i.postimg.cc/SKMQyHd7/squid.jpg", caption: "Hello
    )

    LazyColumn {
        items(posts) { post ->
            Card(modifier = Modifier.padding(8.dp)) {
                Column {
                    Text(text = post.username, fontWeight = FontWeight.Bold)
                    Image(
                        painter = rememberImagePainter(post.imageUrl),
                        contentDescription = null,
                        modifier = Modifier.fillMaxWidth().height(200.dp)
                    )
                    Text(text = post.caption)
                    Text(text = "${post.likes} likes", fontWeight = FontWeight.Light)
                }
            }
        }
    }
}
```

Running Devices