
AnimalCrosser
Cameron-Stewart Smart
1901578
Nook Inc.

Contents

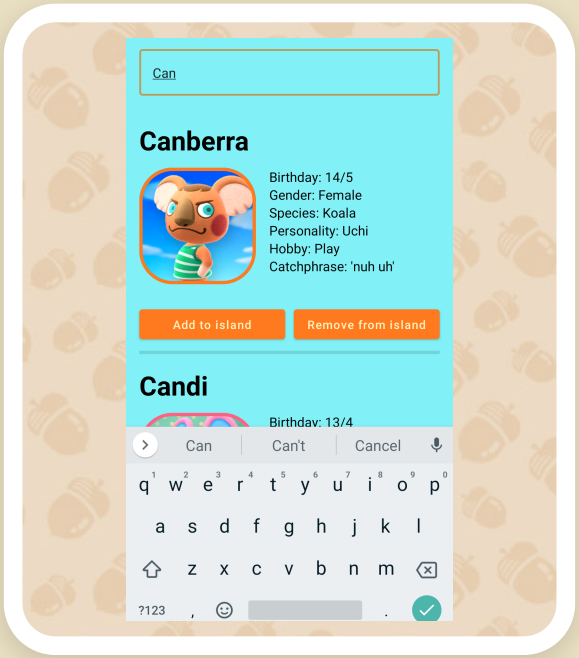
Flow Diagram

Nook Inc.

User Navigation Flow

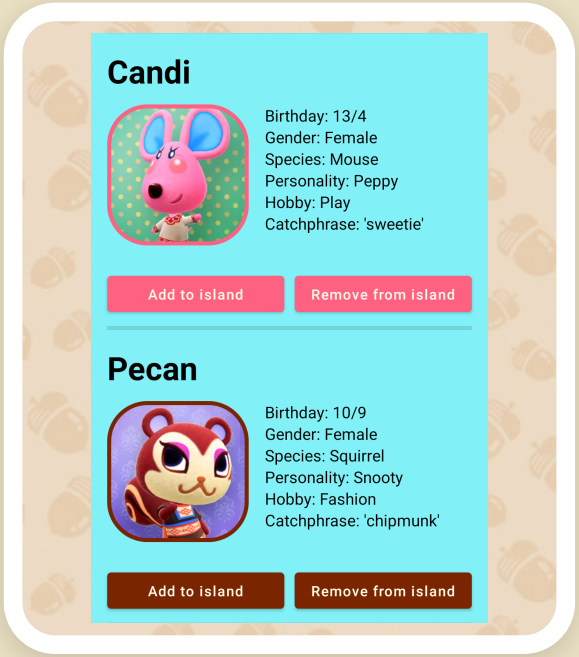

Features
WWW.NINTENDOWIRE.COM
Nook Inc.

Villager Wiki



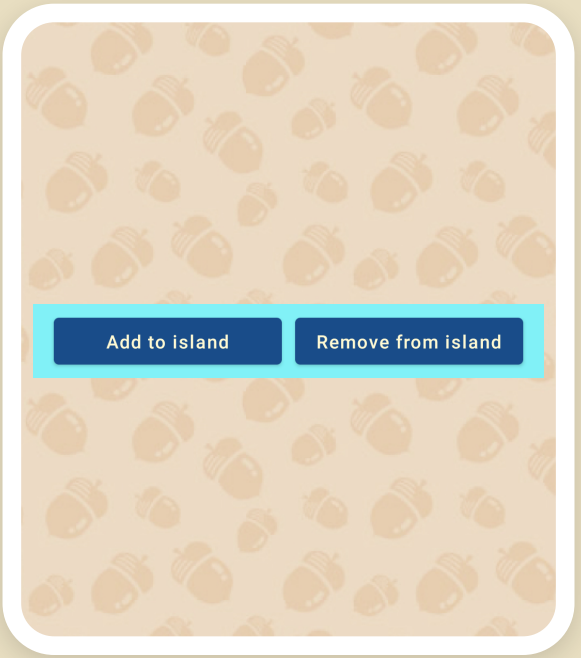
Search bar

- Updates results as typed



Villager Information

- Name
- Personality
- Species
- More



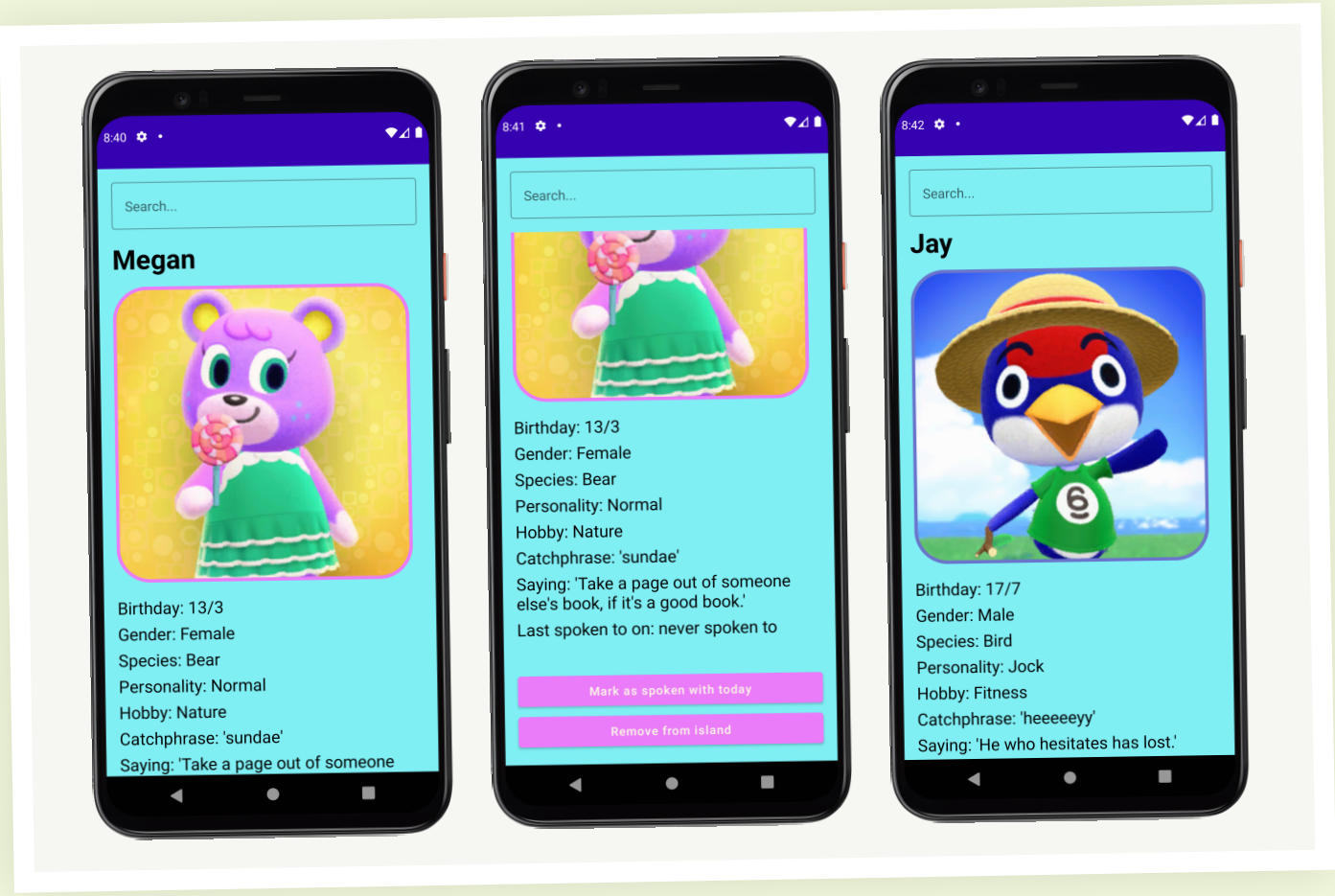
Options

- Add to village
- Remove from village

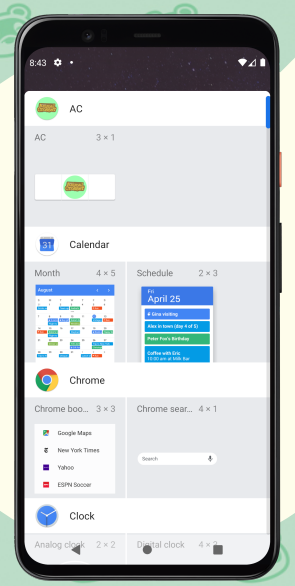
Your Village

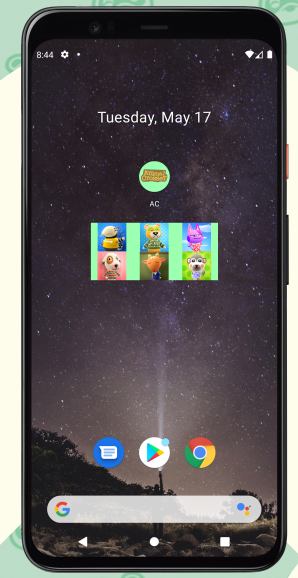
Shows:

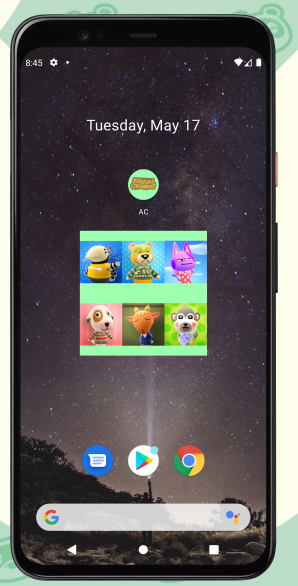
- Larger picture
- Option to remove
- **Date last spoken**



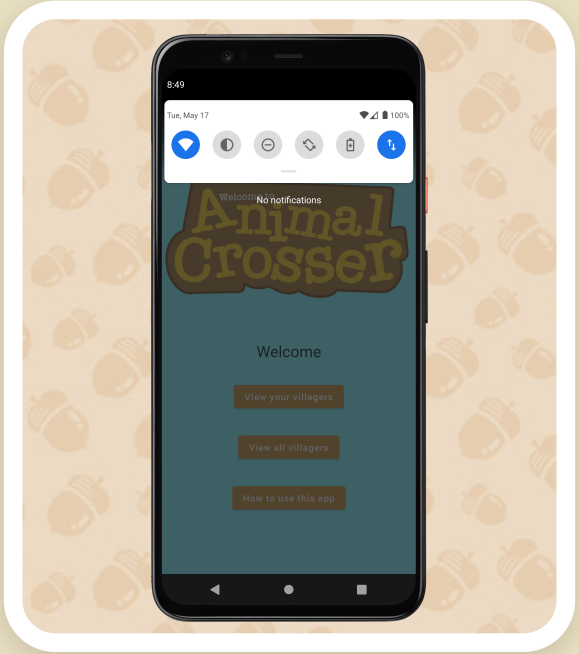
Resizable Widget





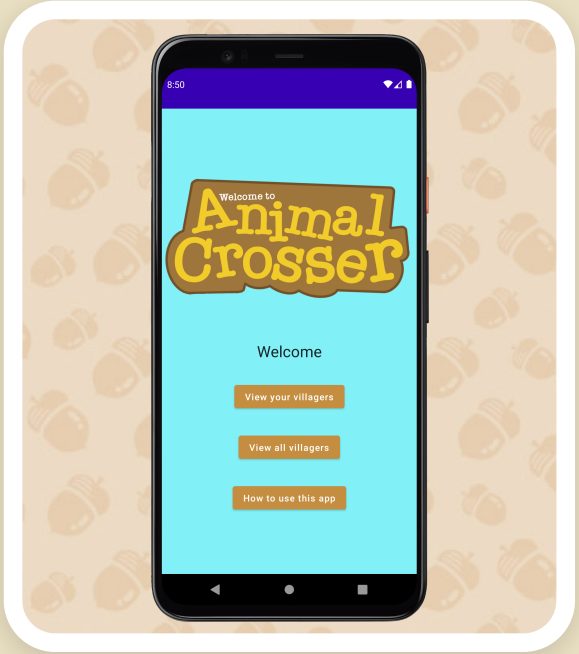


Dark Mode



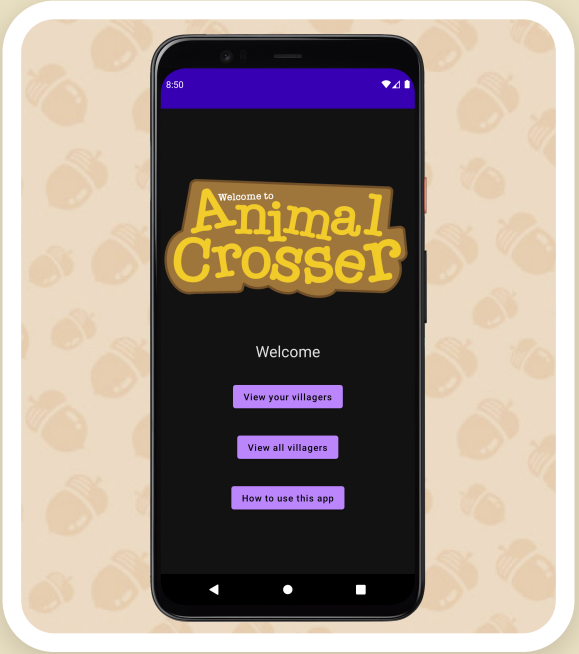
Change with Android

- Automatically adapts with your phones light mode schedule
- Can manually be changed in settings or control centre



Light Mode

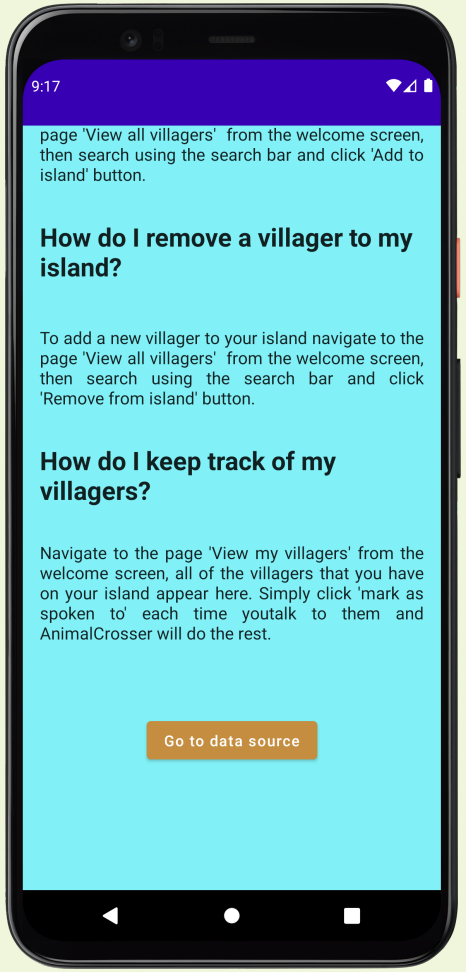
- Animal crossing colour theme

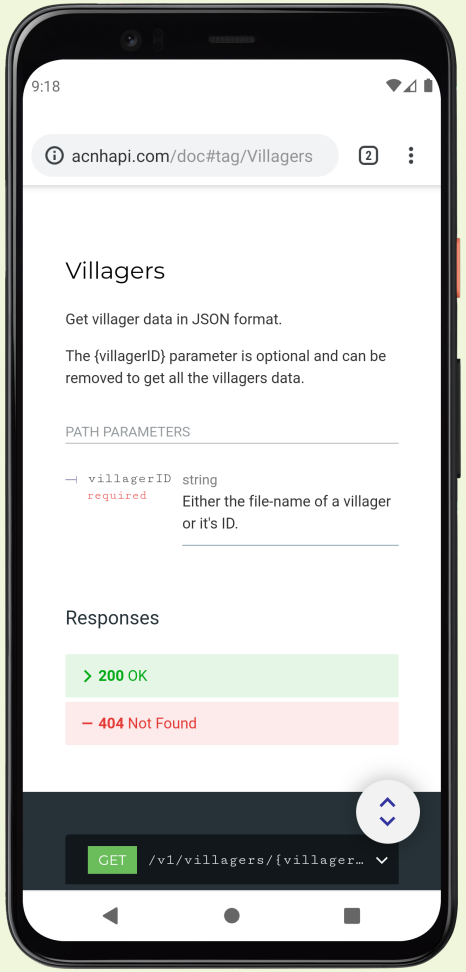


Dark Mode

- Designed for OLEDS by using black and dark grey colours

Built in Guide and API Data Source Browser






Code Snippets
Nook Inc.

Jetpack Compose

- Simplified user interface design
- Navigation implementation that handles creation and destruction of pages
- Reusable composable components
- Works well with Kotlin
- Great documentation available

```
@Composable
@Destination
fun InfoScreen(
    navigator: DestinationsNavigator, // Used for navigation
) {
    val state = rememberLazyListState() // Holds Lazy column position
    val context = LocalContext.current
   // Used to open data source
    val intent = remember { Intent(Intent.ACTION_VIEW, Uri.parse("http://acnhapi.com/doc")) }

    Column(
        modifier = Modifier
            .fillMaxSize(),
    ) {

        LazyColumn( // Allows for scrolling through items
            horizontalAlignment = Alignment.CenterHorizontally,
            state = state
        ) {

            item { // Each object must be put in an object to allow scrolling
                Logo() // Logo.kt composable to display logo image
            }

            item {
                Text("Welcome to AnimalCrosser", // Title
                    modifier = Modifier.fillMaxWidth().padding(horizontal = 16.dp),
                    fontSize = 28.sp,
                    fontWeight = FontWeight.Bold
                )
                Spacer(modifier = Modifier.size(16.dp))
            }
...
```

API Call

```
package art.cameronsma.animalcrosser2.data.remote

import okhttp3.ResponseBody
import retrofit2.http.GET

interface VillagerApi {

    // gets villagers from provided API module
    @GET("villagers")
    suspend fun getListings(): ResponseBody

    companion object {
        const val BASE_URL = "https://mayar.abertay.ac.uk/~1901578/villagers/"
    }

}
```

- Data is initially pulled from an API hosted on Abertay's Mayar server in CSV format
- Retrofit 2 is used to get data and return as response body

Database Interface

- Android room interface was used to interact with an SQLite database stored locally on the phone
- This is used to create a local backup of data called retrieved from API

```
import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query

// Database interface object
// Uses android room to use mySQL statements
@Dao
interface VillagerDao {

    // Insert into database
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertVillagerListings(
        villagerListingEntity: List<VillagerListingEntity>
    )

    // Delete from database
    @Query("DELETE FROM villagerlistingentity")
    suspend fun clearVillagerListings()

    // Select from database where id is equal to input
    @Query(
        """
            SELECT *
            FROM villagerlistingentity
            WHERE LOWER(name) LIKE '%' || LOWER(:query) || '%' OR
                UPPER(:query) == id
        """
    )
    suspend fun searchVillagerListing(query: String): List<VillagerListingEntity>

    // Select from database where id is equal to input
    @Query(
        """
            SELECT *
            FROM villagerlistingentity
            WHERE LOWER(name) LIKE '%' || LOWER(:query) || '%' OR
                UPPER(:query) == id
        """
    )
    suspend fun searchYourVillagerListing(query: String): List<VillagerListingEntity>

    // Mark as in users village where id is equal to input
...
```

Local Data Backup

```
override suspend fun getVillagerListings( // Function to return villager listing of specific id
        fetchFromRemote: Boolean,
        query: String
    ): Flow<Resource<List<VillagerListing>>> { // Returns data flow for list of database objects
        return flow {
            emit(Resource.Loading(true)) // Tells resource object to load data remotely from Api
            val localListing = dao.searchVillagerListing(query)
            emit(Resource.Success( // Tells resource object that data load is successful
                data = localListing.map { it.toVillagerListing() }
            ))

            val isDbEmpty = localListing.isEmpty() && query.isBlank() // Is database empty check
            val shouldJusLoadFromCache = !isDbEmpty && !fetchFromRemote // Id data is not to be loaded
            // remotely and database is empty
            if(shouldJusLoadFromCache) {
                emit(Resource.Loading(false)) // Tells resource object to load data locally from database
                return@flow // Returns data to flow
            }
            val remoteListings = try {
                val response = api.getListings() // Gets data from API
                villagerListingsParser.parse(response.byteStream()) // Parses data from response body
            } catch(e: IOException) { // Exception catcher
                e.printStackTrace()
                emit(Resource.Error("Couldn't load data")) // Tells resource object error occurred
                null // returns null
            } catch (e: HttpException){ // Exception catcher
                e.printStackTrace()
                emit(Resource.Error("Couldn't load data")) // Tells resource object error occurred
                null // returns null
            }

            remoteListings?.let { listings -> // Puts data loaded from remote API into local database
                dao.clearVillagerListings() // Clear entry from database
                dao.insertVillagerListings( // Inserts entry to database
                    listings.map { it.toVillagerListingEntity() }
                )
                emit(Resource.Loading(false)) // Tells resource object to load data locally from database
                emit(Resource.Success( // Tells resource object data load was successful
                    data = dao
                        .searchVillagerListing("")
                        .map { it.toVillagerListing()}
                ))
            }
        }
    }
```

- A local copy of data is kept in the SQLite database and can be used if API is not available and while data is being loaded
- Holds information from API as well as users information such as villagers in village and when last spoken to

Villager List

- Composbable used to show all villagers
- Similar code is used for show your villagers page

```
@Destination
@Composable
fun VillagerListingsScreen(
    navigator: DestinationsNavigator, // Used for navigation
    viewModel: VillagerListingsViewModel = hiltViewModel() // View model for page
) {
...
        SwipeRefresh( // Object that allows for swiping down to refresh that holds rest of page
            state = swipeRefreshState,
            onRefresh = {
                viewModel.onEvent( // Refresh the page when dragged down
                    VillagerListingsEvent.Refresh
                )
            }
        ) {
            LazyColumn( // Column that allows scrolling through items
                modifier = Modifier.fillMaxSize()
            ) {
                items(state.villagers.size) { i -> // Loops through villagers
                    val villager = state.villagers[i] // Villager set to current villager in loop
                    VillagerItem( // Displays current villager using villager item composable
                        villager = villager,
                        modifier = Modifier
                            .fillMaxWidth()
                            .padding(16.dp)
                    )
                    Column (
                       ...
                     )
...
}
```

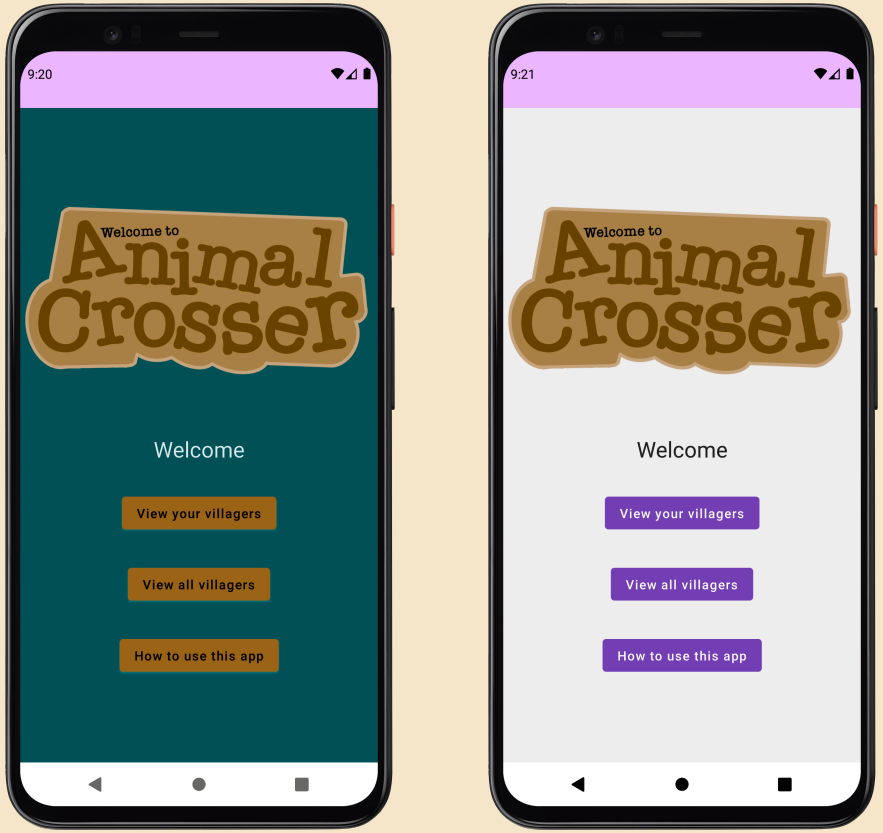
Custom Colour Theme

```
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200,
    surface = graySurface
)

private val LightColorPalette = lightColors(
    primary = darkGoldenrod,
    primaryVariant = darkGoldenrod,
    secondary = icterine,
    surface = electricBlue,
    background = electricBlue,

    )

@Composable
fun AnimalCrosser2Theme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    // Sets as dark theme if system is in dark mode
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

- Two custom colour sets are used to implement a light and dark mode
- Light mode uses animal crossing themed colours
- Dark mode uses more traditional Android dark mode colours such as purple


Usability
Nook Inc.

Usability



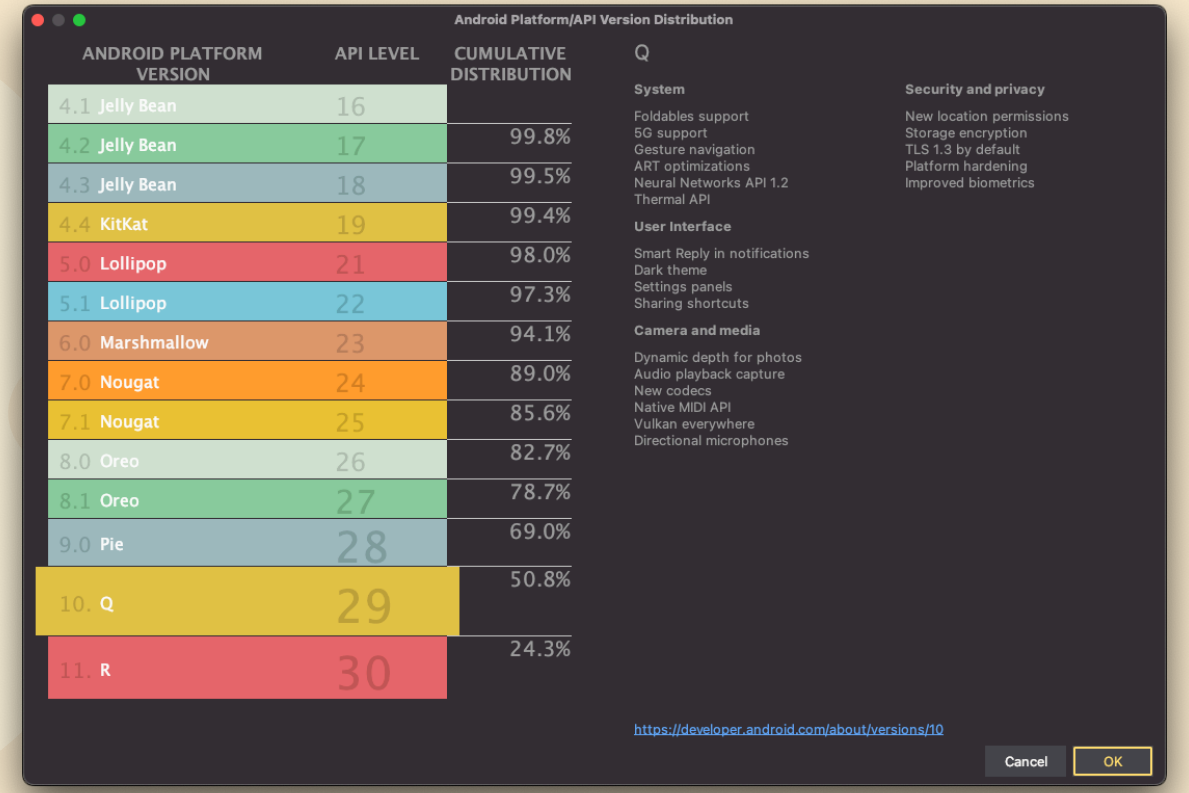
- The app allows the user to change colour themes for visibility
- Also works well with androids high contrast visibility setting for visually impaired users
- No text is displayed as an image so all text can be read by a screen reader

Compatibility

Nook Inc.

Compatibility

- Supports android API 29 so will work with 50.8% of all Android devices




Dodo
Airlines
Security
Nook Inc.

Security

- This app does not store any of the users sensitive personal information
- The app uses the appropriate permissions request cycle to request local data storage and internet access
- All individual information (villagers in village and when last spoken to) are stored locally only


Demonstration


The End
Thank you for watching
Nook Inc.

References

Google Inc. (2022) ‘Compose to Kotlin Compatibility App’. Available at: https://developer.android.com/jetpack/androidx/releases/compose-kotlin (Accessed: 07/05/2022)

Veschi, Gianluca. (2021) ‘Draw two buttons in a row using jetpack compose’. Available at: https://stackoverflow.com/questions/67185495/draw-two-buttons-in-a-row-using-jetpack-compose (Accessed: 08/05/2022)

Asri, Vipul (2020) ‘Colour from hex string in jetpack compose’. Available at: https://stackoverflow.com/questions/60247480/color-from-hex-string-in-jetpack-compose (Accessed: 09/05/2022)

Retool API. (2021) ‘Retool API generator’. Available at: https://retool.com/utilities/generate-api-from-csv (Accessed: 07/05/2022)

Nookipedia. (2021) ‘Animal Crossing API’. Available at: https://api.nookipedia.com/ (Accessed: 07/05/2022)

Lackner, P. (2022) ‘How to build a clean architecture stock market app’. Available at: https://www.youtube.com/watch?v=uLs2FxFSWU4&t=6422s (Accessed: 07/05/2022)

Google Inc. (2022) ‘Use Hilt with other Jetpack libraries’. Available at: https://developer.android.com/training/dependency-injection/hilt-jetpack (Accessed: 09/05/22)

Zeilla. (2021) ‘Animal Crossing New Horizon Icons’. Available at: https://www.spriters-resource.com/nintendo_switch/animalcrossingnewhorizons/sheet/128282/ (Accessed: 11/05/22)

Story, D. (2021) ‘Designing an Animal Crossing Themed Powerpoint’. Available at: https://www.youtube.com/watch?v=1d29CVe1uOw (Accessed: 10/05/22)