

# MenuAnNam - Mobile Application

## Source Code Documentation

Nguyen Thien Nguyen - 10422059

January 20, 2026

# Contents

<b>1 Overview</b>	<b>2</b>
1.1 Project Description . . . . .	2
1.2 Technology Stack . . . . .	2
1.3 AWS Lambda Integration . . . . .	2
<b>2 Application Entry Point</b>	<b>3</b>
2.1 MainActivity.kt . . . . .	3
2.2 Utils.kt . . . . .	3
<b>3 Data Layer</b>	<b>4</b>
3.1 Database . . . . .	4
3.1.1 FlashCardDao.kt . . . . .	4
3.1.2 MenuDatabase.kt . . . . .	4
3.2 Entity . . . . .	5
3.2.1 FlashCard.kt . . . . .	5
3.3 Network . . . . .	5
3.3.1 DataTypes.kt . . . . .	5
3.3.2 NetworkService.kt . . . . .	5
<b>4 Presentation Layer</b>	<b>6</b>
4.1 Navigation . . . . .	6
4.1.1 Navigator.kt . . . . .	6
4.1.2 Routes.kt . . . . .	8
4.2 Components . . . . .	8
4.2.1 TopBarComponent.kt . . . . .	8
4.2.2 BottomBarComponent.kt . . . . .	8
4.3 Screens . . . . .	9
4.3.1 MenuScreen.kt . . . . .	9
4.3.2 LoginScreen.kt . . . . .	9
4.3.3 TokenScreen.kt . . . . .	10
4.3.4 AddScreen.kt . . . . .	11
4.3.5 FilterScreen.kt . . . . .	12
4.3.6 SearchScreen.kt . . . . .	13
4.3.7 EditScreen.kt . . . . .	14
4.3.8 StudyScreen.kt . . . . .	16
<b>5 UI Theme</b>	<b>19</b>
5.1 Color.kt . . . . .	19
5.2 Theme.kt . . . . .	19
5.3 Type.kt . . . . .	19
<b>6 Build Configuration</b>	<b>20</b>
6.1 build.gradle.kts (App Module) . . . . .	20
6.2 libs.versions.toml . . . . .	21
<b>7 Mobile Application Use Cases &amp; Implementation</b>	<b>23</b>
7.1 Use Cases Implementation Status . . . . .	23
7.1.1 1. Request an Authentication Token . . . . .	23
7.1.2 2. Save the Pair Email Address/Authentication Token in the App . . . . .	23
7.1.3 3. Add a Flashcard . . . . .	23
7.1.4 4. Search Flashcards . . . . .	24
7.1.5 5. Edit a Flashcard . . . . .	24
7.1.6 6. Study Flashcards . . . . .	25

# Chapter 1

## Overview

### 1.1 Project Description

MenuAnNam is an Android mobile application built with Jetpack Compose for learning Vietnamese vocabulary through flashcards. The app integrates with AWS Lambda services for authentication and audio synthesis, uses Room database for local storage, and implements modern Android architecture patterns.

### 1.2 Technology Stack

- **UI Framework:** Jetpack Compose with Material 3
- **Language:** Kotlin
- **Database:** Room (SQLite)
- **Networking:** Retrofit 2 with OkHttp
- **Data Storage:** DataStore Preferences
- **Audio Playback:** ExoPlayer (Media3)
- **Navigation:** Type-safe Compose Navigation
- **Backend:** AWS Lambda Functions

### 1.3 AWS Lambda Integration

- **Token Generation:** <https://egsbwqh7kildllpkijk6nt4soq0wlge.lambda-url.ap-southeast-1.on.aws/>
- **Audio Synthesis:** <https://ityqww3rx5vifjpyufgnpkv5te0ibrcx.lambda-url.ap-southeast-1.on.aws/>

# Chapter 2

## Application Entry Point

### 2.1 MainActivity.kt

```
1 package com.example.menuannam
2
3 // Main entry point: initializes database, network,
4 // navigation, and DataStore for app
5 import android.content.Context
6 import android.os.Bundle
7 import androidx.activity.ComponentActivity
8 import androidx.activity.compose.setContent
9 import androidx.compose.runtime.rememberCoroutineScope
10 import androidx.datastore.preferences.core.
11     stringPreferencesKey
12 import androidx.datastore.preferences.preferencesDataStore
13 import androidx.navigation.compose.rememberNavController
14 import com.example.menuannam.data.database.FlashCardDatabase
15 import com.example.menuannam.data.network.NetworkService
16 import com.example.menuannam.presentation.navigation.
17     AppNavigation
18 import com.example.menuannam.ui.theme.MenuAnNamTheme
19 import okhttp3.OkHttpClient
20 import retrofit2.Retrofit
21 import retrofit2.converter.gson.GsonConverterFactory
22 import java.util.concurrent.TimeUnit
23
24 // DataStore for persistent user credentials (email/token)
25 val Context.dataStore by preferencesDataStore(name =
26     "user_credentials")
27
28 TOKEN = stringPreferencesKey("token")
29 EMAIL = stringPreferencesKey("email")
30
31 class MainActivity : ComponentActivity() {
32     override fun onCreate(savedInstanceState: Bundle?) {
33         super.onCreate(savedInstanceState)
34         enableEdgeToEdge()
35         setContent {
36             MenuAnNamTheme {
37                 val navController = rememberNavController()
38                 val appContext = applicationContext
39                 val scope = rememberCoroutineScope()
40
41                 // Database singleton for flashcard CRUD
42                 operations
43                     val db = FlashCardDatabase.getDatabase(
44                         appContext)
45                     val flashCardDao = db.flashCardDao()
46
47                     // HTTP client with 30s timeouts to prevent
48                     Lambda timeout issues
49                     val sharedOkHttpClient = OkHttpClient.
50                         Builder()
51                             .connectTimeout(30, TimeUnit.SECONDS)
52                             .readTimeout(30, TimeUnit.SECONDS)
53                             .build()
54
55                     // Retrofit for Lambda API calls (token
56                     generation, audio synthesis)
57                     val retrofit: Retrofit = Retrofit.Builder()
58                         .baseUrl("https://placeholder.com") //
59                         URL overridden per endpoint
60                         .client(sharedOkHttpClient)
61                         .addConverterFactory(
62                             GsonConverterFactory.create())
63                         .build()
64
65                     val networkService = retrofit.create(
66                         NetworkService::class.java)
67
68                     // Pass all services to navigation system
69                     AppNavigation(
70                         navController,
71                         flashCardDao,
72                         scope,
73                         networkService
74                     )
75                 }
76             }
77         }
78     }
79
80     ./app/src/main/java/com/example/menuannam/MainActivity.kt
```

### 2.2 Utils.kt

```
1 package com.example.menuannam
2
3 import android.content.Context
4 import java.io.File
5 import java.io.FileOutputStream
6 import java.security.MessageDigest
7
8 // Convert string to MD5 hash for unique audio filenames (e.
9 // g., "hello" → "5d41402abc...")
10 fun String.toMd5(): String {
11     val bytes = MessageDigest.getInstance("MD5").digest(this
12         .toByteArray())
13     return bytes.joinToString("") { "%02x".format(it) }
14 }
15
16 // Save Base64-decoded audio bytes to app's private storage
17 // as MP3
18 fun saveAudioToInternalStorage(context: Context, audioData:
19     ByteArray, filename: String): File {
20     val file = File(context.filesDir, filename) // Private
21     // app directory
22     FileOutputStream(file).use { fos ->
23         fos.write(audioData)
24     }
25     return file // Return File for ExoPlayer playback
26 }
27
28 // Check if audio file exists in cache before making API
29 // call
30 fun getCachedAudioFile(context: Context, word: String): File
31     ? {
32     val file = File(context.filesDir, "${word.toMd5()}.mp3")
33     return if (file.exists()) file else null
34 }
35
36     ./app/src/main/java/com/example/menuannam/Utils.kt
```

# Chapter 3

## Data Layer

### 3.1 Database

#### 3.1.1 FlashCardDao.kt

```
1 package com.example.menuannam.data.database
2
3 import androidx.room.Dao
4 import androidx.room.Delete
5 import androidx.room.Insert
6 import androidx.room.OnConflictStrategy
7 import androidx.room.Query
8 import androidx.room.RawQuery
9 import androidx.sqlite.db.SupportSQLiteQuery
10 import com.example.menuannam.data.entity.FlashCard
11
12 // DAO for FlashCard CRUD operations; all suspend functions
13 // run on background thread
13 @Dao
14 interface FlashCardDao {
15     // Raw SQL for database checkpoint operations
16     @RawQuery
17     fun checkpoint(supportSQLQuery: SupportSQLiteQuery): Int
18
19     @Query("SELECT * FROM FlashCards")
20     suspend fun getAll(): List<FlashCard>
21
22     // Random shuffle for study sessions
23     @Query("SELECT * FROM FlashCards ORDER BY RANDOM() LIMIT :size")
24     suspend fun getLesson(size: Int): List<FlashCard>
25
26     // Duplicate detection by matching both fields
27     @Query(
28         "SELECT * FROM FlashCards WHERE english_card LIKE :english AND "
29         "    vietnamese_card LIKE :vietnamese LIMIT 1"
30     )
31     suspend fun findByCards(english: String, vietnamese: String): FlashCard?
32
33     // IGNORE strategy: silently skip if duplicate exists (returns -1)
34     @Insert(onConflict = OnConflictStrategy.IGNORE)
35     suspend fun insert(flashCard: FlashCard): Long
36
37     @Insert(onConflict = OnConflictStrategy.IGNORE)
38     suspend fun insertAll(vararg flashCard: FlashCard)
39
40     // Update by old English/Vietnamese pair (search key) to
41     // new values
41     @Query(
42         "UPDATE FlashCards SET english_card = :englishNew "
43         + ", vietnamese_card =:vietnameseNew " +
44             "WHERE english_card = :englishOld " +
45             "AND vietnamese_card = :vietnameseOld"
46     )
47     suspend fun updateFlashCard(
48         englishOld: String,
49         vietnameseOld: String,
50         englishNew: String,
51         vietnameseNew: String
52     )
53
54     @Query(
55         "DELETE FROM FlashCards WHERE english_card = :english "
56         + "AND vietnamese_card =:vietnamese"
57     )
58     suspend fun deleteFlashCard(english: String, vietnamese: String)
59
60     @Query("SELECT * FROM FlashCards WHERE uid = :id")
61     suspend fun getById(id: Int): FlashCard?
62
63     @Query("SELECT COUNT(*) FROM FlashCards")
64     suspend fun getCount(): Int
65
66     @Query("UPDATE FlashCards SET english_card = :english,
67             vietnamese_card = :vietnamese WHERE uid = :id")
68     suspend fun update(id: Int, english: String, vietnamese: String)
69
70     @Delete
71     suspend fun delete(flashCard: FlashCard)
72
73     // CASE WHEN for exact vs partial match: exactEn=1 →
74     // LIKE :en, exactEn=0 → LIKE '%' || :en || '%'
75     @Query(
76         "SELECT * FROM FlashCards WHERE " +
77             "(CASE WHEN :exactEn THEN english_card LIKE "
78             ":en " +
79                 "WHEN NOT :exactEn THEN english_card LIKE "
80                 "'%' || :en || '%' END) " +
81                 "AND " +
82                     "(CASE WHEN :exactVn THEN vietnamese_card "
83                     "LIKE :vn " +
84                         "WHEN NOT :exactVn THEN vietnamese_card LIKE "
85                         "'%' || :vn || '%' END)"
86     )
87     suspend fun getFilteredFlashCards(en: String, exactEn: Int, vn: String, exactVn: Int): List<FlashCard>
88 }
```

./app/src/main/java/com/example/menuannam/data/database/FlashCardDatabase.kt

#### 3.1.2 MenuDatabase.kt

```
1 package com.example.menuannam.data.database
2
3 import android.content.Context
4 import androidx.room.Database
5 import androidx.room.Room
6 import androidx.room.RoomDatabase
7 import com.example.menuannam.data.entity.FlashCard
8 import kotlin.jvm.Volatile
9
10 // Room database with singleton pattern for thread-safe
11 // FlashCard persistence
11 @Database(entities = [FlashCard::class], version = 1)
12 abstract class FlashCardDatabase : RoomDatabase() {
13     abstract fun flashCardDao(): FlashCardDao
14
15     companion object {
16         @Volatile // All threads see latest value
17         immediately
18         private var INSTANCE: FlashCardDatabase? = null
19
19         // Double-checked locking: fast path check, then
19         synchronized creation to prevent race conditions
20     }
21 }
```

```
20     fun getDatabase(context: Context): FlashCardDatabase    27           INSTANCE = instance
21     {                                                 28           instance
22         return INSTANCE ?: synchronized(this) {          29               }
23             val instance = Room.databaseBuilder(
24                 context.applicationContext, // Use app 30               )
25                     context to prevent memory leaks   31               }
26                     FlashCardDatabase::class.java,      32           }
27                     "FlashCardDatabase"                .app/src/main/java/com/example/menuannam/data/database/M
28             ).build()                                     I
```

## 3.2 Entity

### 3.2.1 FlashCard.kt

```
1 package com.example.menuannam.data.entity
2
3 import androidx.room.ColumnInfo
4 import androidx.room.Entity
5 import androidx.room.Index
6 import androidx.room.PrimaryKey
7 import kotlinx.serialization.Serializable
8
9 // FlashCard entity with unique index on (english_card,
10 // vietnamese_card) to prevent duplicates
11 @Entity(tableName = "FlashCards", indices = [Index(
12     value = ["english_card", "vietnamese_card"],
13     unique = true // Database-level constraint enforced by
```

```
    Room
13  }])
14  @Serializable
15  data class FlashCard(
16      @PrimaryKey(autoGenerate = true) val uid: Int, // Auto-
           increment from 1
17      @ColumnInfo(name = "english_card") val englishCard:
           String?,
18      @ColumnInfo(name = "vietnamesecard") val vietnamesecard
           : String?
19  )
./app/src/main/java/com/example/menuannam/data/entity/Flash
```

### 3.3 Network

### 3.3.1 DataTypes.kt

```
1 package com.example.menuannam.data.network
2
3 import kotlinx.serialization.Serializable
4
5 // Request payloads and responses for Lambda API calls
6
7 @Serializable
8 data class UserCredential(val email: String) // Token
9     generation request
10
11 @Serializable
12 data class TokenResponse(
13     val code: Int, // HTTP status (200 = success)
14     val message: String // Token string or error message
15 )
16
17 @Serializable
```

```
17 data class AudioRequest(
18     val word: String, // Vietnamese word for audio synthesis
19     val email: String, // User tracking
20     val token: String // Auth token (expires after some time
21 )
22
23 @Serializable
24 data class AudioResponse(
25     val code: Int, // HTTP status (200 = success, 500 =
26     token invalid/expired)
27     val message: String // Base64-encoded MP3 or error
28     message
29 )
```

### 3.3.2 NetworkService.kt

```
1 package com.example.menuannam.data.network
2
3 import retrofit2.http.Body
4 import retrofit2.http.PUT
5 import retrofit2.http.Url
6
7 // Retrofit interface for AWS Lambda endpoints; default URLs
8 // overridden per call
9 interface NetworkService {
10     // Token generation: sends email, returns token or error
11     @PUT
12     suspend fun generateToken(
13         @Url url: String = "https://
14         egsbwqh7kildllpkijk6nt4soq0wlge.lambda-url.ap-
15         southeast-1.on.aws/"
```

```
13     @Body email: UserCredential
14 ): TokenResponse
15
16 // Audio synthesis: sends Vietnamese word + token,
17 // returns Base64 MP3 or 500 if token expired
17 @PUT
18 suspend fun generateAudio(
19     @Url url: String = "https://
20 ityqwv3rx5vijpyufgnpkv5te0ibrx.lambda-url.ap-
21 southeast-1.on.aws/",
20     @Body request: AudioRequest
21 ): AudioResponse
22 }
```

./app/src/main/java/com/example/menuannam/data/network/Net

# Chapter 4

# Presentation Layer

## 4.1 Navigation

### 4.1.1 Navigator.kt

```

1 package com.example.menuannam.presentation.navigation
2
3 import androidx.compose.foundation.layout.padding
4 import androidx.compose.material3.Scaffold
5 import androidx.compose.runtime.*
6 import androidx.navigation.NavHostController
7 import androidx.navigation.compose.NavHost
8 import androidx.navigation.compose.composable
9 import androidx.navigation.compose.
10     currentBackStackEntryAsState
11 import androidx.navigation.toRoute
12 import kotlinx.coroutines.CoroutineScope
13 import com.example.menuannam.data.database.FlashCardDao
14 import com.example.menuannam.presentation.screens.MenuScreen
15 import com.example.menuannam.presentation.screens.AddScreen
16 import com.example.menuannam.presentation.screens.
17     StudyScreen
18 import com.example.menuannam.presentation.screens.
19     CardViewMode
20 import com.example.menuannam.presentation.screens.
21     SearchScreen
22 import com.example.menuannam.presentation.screens.
23     FilterScreen
24 import com.example.menuannam.presentation.screens.EditScreen
25 import com.example.menuannam.presentation.screens.
26     LoginScreen
27 import com.example.menuannam.presentation.screens.
28     TokenScreen
29 import com.example.menuannam.data.network.NetworkService
30 import com.example.menuannam.presentation.components.
31     TopBarComponent
32 import com.example.menuannam.presentation.components.
33     BottomBarComponent
34
35 // Central Navigation Hub: Controls all screen transitions
36     and manages app state
37 // Uses Compose Navigation with type-safe routes for compile
38     -time error checking
39 @Composable
40 fun AppNavigation(
41     navigation: NavHostController, // NavController from
42     MainActivity
43     flashCardDao: FlashCardDao, // Database access
44     coroutineScope: CoroutineScope, // For async operations
45     (launched from MainActivity)
46     networkService: NetworkService // Retrofit service for
47     API calls
48 ) {
49     // Shared message across all screens - updated by each
50     // screen, displayed in BottomBar
51     var message by remember { mutableStateOf("Welcome!") }
52     val changeMessage: (String) -> Unit = { message = it }
53
54     // Get current route name from back stack to determine
55     what screen is currently displayed
56     val backStackEntry by navigation.
57         currentBackStackEntryAsState()
58     val routeName = backStackEntry?.destination?.route
59
60     // Map route names to user-friendly titles displayed in
61     // TopBar
62     val title = when (routeName) {
63         HomeRoute::class.qualifiedName -> "Menu An Nam"
64         StudyCardsRoute::class.qualifiedName -> "Study Cards"
65         "AddCardRoute::class.qualifiedName -> "Add Card"
66         SearchCardsRoute::class.qualifiedName -> "Search
67             Results"
68     }
69
70     FilterRoute::class.qualifiedName -> "Search Cards"
71     LoginRoute::class.qualifiedName -> "Login"
72     ShowCardRoute::class.qualifiedName -> "Flash Card"
73     EditCardRoute::class.qualifiedName -> "Edit Card"
74     else -> "Menu An Nam"
75 }
76
77 /**
78 * Show back button on all screens EXCEPT home
79 * Only home doesn't have a back button (it's the start
80 destination)
81 */
82 val showBack = routeName != HomeRoute::class.
83     qualifiedName
84 val navigateBack: () -> Unit = { navigation.navigateUp()
85 }
86
87 // ===== SCAFFOLD LAYOUT =====
88 /**
89 * Scaffold provides standard app structure:
90 * - topBar: Title and back button
91 * - content: NavHost with screen composition
92 * - bottomBar: Status message display
93 */
94 Scaffold(
95     topBar = {
96         TopBarComponent(
97             title = title,
98             showBack = if (showBack) navigateBack else
99                 null
100     )
101 },
102     bottomBar = {
103         BottomBarComponent(message = message)
104     }
105 ) { innerPadding ->
106
107     // ===== NAVIGATION HOST =====
108     /**
109      * NavHost manages all composable screens
110      * startDestination = HomeRoute (app always starts
111 at home)
112      * Padding prevents overlap with TopBar/BottomBar
113      */
114     NavHost(
115         navController = navigation,
116         startDestination = HomeRoute,
117         modifier = androidx.compose.ui.Modifier.padding(
118             innerPadding)
119     ) {
120
121         // ===== HOME ROUTE - Main menu
122         // =====
123
124         composable<HomeRoute> {
125             MenuScreen(
126                 onStudy = {
127                     navigation.navigate(StudyCardsRoute)
128                 },
129                 onAdd = { navigation.navigate(
130                     AddCardRoute) },
131                 onSearch = { navigation.navigate(
132                     FilterRoute) },
133                 onLogin = { navigation.navigate(
134                     LoginRoute) }.
135             }
136         }
137     }
138 }

```

```

106         changeMessage = changeMessage
107     }
108     //=====
109     // LOGIN ROUTE - Get email and request token
110     //=====
111     composable<LoginRoute> {
112         LoginScreen(
113             changeMessage = changeMessage,
114             networkService = networkService,
115             navigateToToken = { enteredEmail ->
116                 navigation.navigate(TokenRoute(
117                     enteredEmail))
118             }
119         )
120     }
121     //=====
122     // STUDY CARDS ROUTE - Interactive flashcard
123     study session
124     //=====
125     composable<StudyCardsRoute> {
126         StudyScreen(
127             changeMessage = changeMessage,
128             flashCardDao = flashCardDao,
129             networkService = networkService,
130             mode = CardViewMode.STUDY_SESSION,
131             coroutineScope = coroutineScope
132         )
133     }
134     //=====
135     // ADD CARD ROUTE - Create new flashcard
136     //=====
137     composable<AddCardRoute> {
138         AddScreen(
139             changeMessage = changeMessage,
140             insertFlashCard = { card ->
141                 coroutineScope.launch {
142                     // Insert card (Room ignores if
143                     duplicate)
144                     flashCardDao.insertAll(card)
145                     changeMessage("Added: ${card.
146                     englishCard}")
147                 }
148             }
149         )
150     }
151     //=====
152     // SEARCH CARDS ROUTE - Display filtered search
153     results
154     //=====
155     composable<SearchCardsRoute> { backStackEntry ->
156         val route = backStackEntry.toRoute<
157             SearchCardsRoute>()
158         SearchScreen(
159             changeMessage = changeMessage,
160             flashCardDao = flashCardDao,
161             onEdit = { cardId ->
162                 navigation.navigate(EditCardRoute(
163                     cardId))
164             },
165             englishText = route.englishText,
166             exactEnglish = route.exactEnglish,
167             vietnameseText = route.vietnameseText,
168             exactVietnamese = route.exactVietnamese
169         )
170     }
171     //=====
172     // FILTER ROUTE - Search input form
173     //=====
174     composable<FilterRoute> {
175         FilterScreen(
176             changeMessage = changeMessage,
177             onSearch = { en, exactEn, vn, exactVn ->
178                 navigation.navigate(
179                     SearchCardsRoute(
180                         englishText = en,
181                         exactEnglish = exactEn,
182                         vietnameseText = vn,
183                         exactVietnamese = exactVn
184                     )
185                 )
186             }
187         )
188     }
189     //=====
190     // EDIT CARD ROUTE - Edit existing flashcard
191     with audio management
192     //=====
193     composable<EditCardRoute> { backStackEntry ->
194         val route = backStackEntry.toRoute<
195             EditCardRoute>()
196         EditScreen(
197             cardId = route.id,
198             flashCardDao = flashCardDao,
199             networkService = networkService,
200             changeMessage = changeMessage,
201             onCardUpdated = { navigation.navigateUp
202                 () }
203         )
204     }
205     //=====
206     // SHOW CARD ROUTE - View single card and delete
207     if desired
208     //=====
209     /**
210      * Receives cardId from SearchScreen
211      * Displays full card details with delete and
212      * play audio options
213      */
214     composable<ShowCardRoute> { backStackEntry ->
215         val route = backStackEntry.toRoute<
216             ShowCardRoute>()
217         StudyScreen(
218             changeMessage = changeMessage,
219             flashCardDao = flashCardDao,
220             networkService = networkService,
221             mode = CardViewMode.SINGLE_CARD,
222             cardId = route.id,
223             onCardDeleted = { navigation.navigateUp
224                 () }
225         )
226     }
227     //=====
228     // TOKEN ROUTE - Store token from login
229     //=====
230     /**
231      * Receives email parameter from LoginRoute
232      * User enters token string received via email
233      * Saves token to DataStore for later audio API
234      */
235     composable<TokenRoute> { backStackEntry ->
236         val route = backStackEntry.toRoute<
237             TokenRoute>()
238         TokenScreen(
239             email = route.email,
240             changeMessage = changeMessage,
241             navigateToHome = {
242                 navigation.navigate(HomeRoute) {
243                     popUpTo(HomeRoute) { inclusive =
244                         true }
245                 }
246             }
247         )
248     }
249     //=====
250     // APP ROUTE - Main application entry point
251     //=====
252     composable<AppRoute> {
253         AppScreen(
254             changeMessage = changeMessage,
255             onSearch = { en, exactEn, vn, exactVn ->
256                 navigation.navigate(
257                     SearchCardsRoute(
258                         englishText = en,
259                         exactEnglish = exactEn,
260                         vietnameseText = vn,
261                         exactVietnamese = exactVn
262                     )
263                 )
264             }
265         )
266     }
267     //=====
268     // HOME ROUTE - Welcome screen
269     //=====
270     composable<HomeRoute> {
271         HomeScreen(
272             changeMessage = changeMessage,
273             onSearch = { en, exactEn, vn, exactVn ->
274                 navigation.navigate(
275                     SearchCardsRoute(
276                         englishText = en,
277                         exactEnglish = exactEn,
278                         vietnameseText = vn,
279                         exactVietnamese = exactVn
280                     )
281                 )
282             }
283         )
284     }
285     //=====
286     // FILTER ROUTE - Search input form
287     //=====
288     composable<FilterRoute> {
289         FilterScreen(
290             changeMessage = changeMessage,
291             onSearch = { en, exactEn, vn, exactVn ->
292                 navigation.navigate(
293                     SearchCardsRoute(
294                         englishText = en,
295                         exactEnglish = exactEn,
296                         vietnameseText = vn,
297                         exactVietnamese = exactVn
298                     )
299                 )
300             }
301         )
302     }
303     //=====
304     // EDIT CARD ROUTE - Edit existing flashcard
305     with audio management
306     //=====
307     composable<EditCardRoute> { backStackEntry ->
308         val route = backStackEntry.toRoute<
309             EditCardRoute>()
310         EditScreen(
311             cardId = route.id,
312             flashCardDao = flashCardDao,
313             networkService = networkService,
314             changeMessage = changeMessage,
315             onCardUpdated = { navigation.navigateUp
316                 () }
317         )
318     }
319     //=====
320     // SHOW CARD ROUTE - View single card and delete
321     if desired
322     //=====
323     /**
324      * Receives cardId from SearchScreen
325      * Displays full card details with delete and
326      * play audio options
327      */
328     composable<ShowCardRoute> { backStackEntry ->
329         val route = backStackEntry.toRoute<
330             ShowCardRoute>()
331         StudyScreen(
332             changeMessage = changeMessage,
333             flashCardDao = flashCardDao,
334             networkService = networkService,
335             mode = CardViewMode.SINGLE_CARD,
336             cardId = route.id,
337             onCardDeleted = { navigation.navigateUp
338                 () }
339         )
340     }
341     //=====
342     // TOKEN ROUTE - Store token from login
343     //=====
344     /**
345      * Receives email parameter from LoginRoute
346      * User enters token string received via email
347      * Saves token to DataStore for later audio API
348      */
349     composable<TokenRoute> { backStackEntry ->
350         val route = backStackEntry.toRoute<
351             TokenRoute>()
352         TokenScreen(
353             email = route.email,
354             changeMessage = changeMessage,
355             navigateToHome = {
356                 navigation.navigate(HomeRoute) {
357                     popUpTo(HomeRoute) { inclusive =
358                         true }
359                 }
360             }
361         )
362     }
363     //=====
364     // APP ROUTE - Main application entry point
365     //=====
366     composable<AppRoute> {
367         AppScreen(
368             changeMessage = changeMessage,
369             onSearch = { en, exactEn, vn, exactVn ->
370                 navigation.navigate(
371                     SearchCardsRoute(
372                         englishText = en,
373                         exactEnglish = exactEn,
374                         vietnameseText = vn,
375                         exactVietnamese = exactVn
376                     )
377                 )
378             }
379         )
380     }
381     //=====
382     // HOME ROUTE - Welcome screen
383     //=====
384     composable<HomeRoute> {
385         HomeScreen(
386             changeMessage = changeMessage,
387             onSearch = { en, exactEn, vn, exactVn ->
388                 navigation.navigate(
389                     SearchCardsRoute(
390                         englishText = en,
391                         exactEnglish = exactEn,
392                         vietnameseText = vn,
393                         exactVietnamese = exactVn
394                     )
395                 )
396             }
397         )
398     }
399     //=====
400     // FILTER ROUTE - Search input form
401     //=====
402     composable<FilterRoute> {
403         FilterScreen(
404             changeMessage = changeMessage,
405             onSearch = { en, exactEn, vn, exactVn ->
406                 navigation.navigate(
407                     SearchCardsRoute(
408                         englishText = en,
409                         exactEnglish = exactEn,
410                         vietnameseText = vn,
411                         exactVietnamese = exactVn
412                     )
413                 )
414             }
415         )
416     }
417     //=====
418     // EDIT CARD ROUTE - Edit existing flashcard
419     with audio management
420     //=====
421     composable<EditCardRoute> { backStackEntry ->
422         val route = backStackEntry.toRoute<
423             EditCardRoute>()
424         EditScreen(
425             cardId = route.id,
426             flashCardDao = flashCardDao,
427             networkService = networkService,
428             changeMessage = changeMessage,
429             onCardUpdated = { navigation.navigateUp
430                 () }
431         )
432     }
433     //=====
434     // SHOW CARD ROUTE - View single card and delete
435     if desired
436     //=====
437     /**
438      * Receives cardId from SearchScreen
439      * Displays full card details with delete and
440      * play audio options
441      */
442     composable<ShowCardRoute> { backStackEntry ->
443         val route = backStackEntry.toRoute<
444             ShowCardRoute>()
445         StudyScreen(
446             changeMessage = changeMessage,
447             flashCardDao = flashCardDao,
448             networkService = networkService,
449             mode = CardViewMode.SINGLE_CARD,
450             cardId = route.id,
451             onCardDeleted = { navigation.navigateUp
452                 () }
453         )
454     }
455     //=====
456     // TOKEN ROUTE - Store token from login
457     //=====
458     /**
459      * Receives email parameter from LoginRoute
460      * User enters token string received via email
461      * Saves token to DataStore for later audio API
462      */
463     composable<TokenRoute> { backStackEntry ->
464         val route = backStackEntry.toRoute<
465             TokenRoute>()
466         TokenScreen(
467             email = route.email,
468             changeMessage = changeMessage,
469             navigateToHome = {
470                 navigation.navigate(HomeRoute) {
471                     popUpTo(HomeRoute) { inclusive =
472                         true }
473                 }
474             }
475         )
476     }
477     //=====
478     // APP ROUTE - Main application entry point
479     //=====
480     composable<AppRoute> {
481         AppScreen(
482             changeMessage = changeMessage,
483             onSearch = { en, exactEn, vn, exactVn ->
484                 navigation.navigate(
485                     SearchCardsRoute(
486                         englishText = en,
487                         exactEnglish = exactEn,
488                         vietnameseText = vn,
489                         exactVietnamese = exactVn
490                     )
491                 )
492             }
493         )
494     }
495     //=====
496     // HOME ROUTE - Welcome screen
497     //=====
498     composable<HomeRoute> {
499         HomeScreen(
500             changeMessage = changeMessage,
501             onSearch = { en, exactEn, vn, exactVn ->
502                 navigation.navigate(
503                     SearchCardsRoute(
504                         englishText = en,
505                         exactEnglish = exactEn,
506                         vietnameseText = vn,
507                         exactVietnamese = exactVn
508                     )
509                 )
510             }
511         )
512     }
513     //=====
514     // FILTER ROUTE - Search input form
515     //=====
516     composable<FilterRoute> {
517         FilterScreen(
518             changeMessage = changeMessage,
519             onSearch = { en, exactEn, vn, exactVn ->
520                 navigation.navigate(
521                     SearchCardsRoute(
522                         englishText = en,
523                         exactEnglish = exactEn,
524                         vietnameseText = vn,
525                         exactVietnamese = exactVn
526                     )
527                 )
528             }
529         )
530     }
531     //=====
532     // EDIT CARD ROUTE - Edit existing flashcard
533     with audio management
534     //=====
535     composable<EditCardRoute> { backStackEntry ->
536         val route = backStackEntry.toRoute<
537             EditCardRoute>()
538         EditScreen(
539             cardId = route.id,
540             flashCardDao = flashCardDao,
541             networkService = networkService,
542             changeMessage = changeMessage,
543             onCardUpdated = { navigation.navigateUp
544                 () }
545         )
546     }
547     //=====
548     // SHOW CARD ROUTE - View single card and delete
549     if desired
550     //=====
551     /**
552      * Receives cardId from SearchScreen
553      * Displays full card details with delete and
554      * play audio options
555      */
556     composable<ShowCardRoute> { backStackEntry ->
557         val route = backStackEntry.toRoute<
558             ShowCardRoute>()
559         StudyScreen(
560             changeMessage = changeMessage,
561             flashCardDao = flashCardDao,
562             networkService = networkService,
563             mode = CardViewMode.SINGLE_CARD,
564             cardId = route.id,
565             onCardDeleted = { navigation.navigateUp
566                 () }
567         )
568     }
569     //=====
570     // TOKEN ROUTE - Store token from login
571     //=====
572     /**
573      * Receives email parameter from LoginRoute
574      * User enters token string received via email
575      * Saves token to DataStore for later audio API
576      */
577     composable<TokenRoute> { backStackEntry ->
578         val route = backStackEntry.toRoute<
579             TokenRoute>()
580         TokenScreen(
581             email = route.email,
582             changeMessage = changeMessage,
583             navigateToHome = {
584                 navigation.navigate(HomeRoute) {
585                     popUpTo(HomeRoute) { inclusive =
586                         true }
587                 }
588             }
589         )
590     }
591     //=====
592     // APP ROUTE - Main application entry point
593     //=====
594     composable<AppRoute> {
595         AppScreen(
596             changeMessage = changeMessage,
597             onSearch = { en, exactEn, vn, exactVn ->
598                 navigation.navigate(
599                     SearchCardsRoute(
600                         englishText = en,
601                         exactEnglish = exactEn,
602                         vietnameseText = vn,
603                         exactVietnamese = exactVn
604                     )
605                 )
606             }
607         )
608     }
609     //=====
610     // HOME ROUTE - Welcome screen
611     //=====
612     composable<HomeRoute> {
613         HomeScreen(
614             changeMessage = changeMessage,
615             onSearch = { en, exactEn, vn, exactVn ->
616                 navigation.navigate(
617                     SearchCardsRoute(
618                         englishText = en,
619                         exactEnglish = exactEn,
620                         vietnameseText = vn,
621                         exactVietnamese = exactVn
622                     )
623                 )
624             }
625         )
626     }
627     //=====
628     // FILTER ROUTE - Search input form
629     //=====
630     composable<FilterRoute> {
631         FilterScreen(
632             changeMessage = changeMessage,
633             onSearch = { en, exactEn, vn, exactVn ->
634                 navigation.navigate(
635                     SearchCardsRoute(
636                         englishText = en,
637                         exactEnglish = exactEn,
638                         vietnameseText = vn,
639                         exactVietnamese = exactVn
640                     )
641                 )
642             }
643         )
644     }
645     //=====
646     // EDIT CARD ROUTE - Edit existing flashcard
647     with audio management
648     //=====
649     composable<EditCardRoute> { backStackEntry ->
650         val route = backStackEntry.toRoute<
651             EditCardRoute>()
652         EditScreen(
653             cardId = route.id,
654             flashCardDao = flashCardDao,
655             networkService = networkService,
656             changeMessage = changeMessage,
657             onCardUpdated = { navigation.navigateUp
658                 () }
659         )
660     }
661     //=====
662     // SHOW CARD ROUTE - View single card and delete
663     if desired
664     //=====
665     /**
666      * Receives cardId from SearchScreen
667      * Displays full card details with delete and
668      * play audio options
669      */
670     composable<ShowCardRoute> { backStackEntry ->
671         val route = backStackEntry.toRoute<
672             ShowCardRoute>()
673         StudyScreen(
674             changeMessage = changeMessage,
675             flashCardDao = flashCardDao,
676             networkService = networkService,
677             mode = CardViewMode.SINGLE_CARD,
678             cardId = route.id,
679             onCardDeleted = { navigation.navigateUp
680                 () }
681         )
682     }
683     //=====
684     // TOKEN ROUTE - Store token from login
685     //=====
686     /**
687      * Receives email parameter from LoginRoute
688      * User enters token string received via email
689      * Saves token to DataStore for later audio API
690      */
691     composable<TokenRoute> { backStackEntry ->
692         val route = backStackEntry.toRoute<
693             TokenRoute>()
694         TokenScreen(
695             email = route.email,
696             changeMessage = changeMessage,
697             navigateToHome = {
698                 navigation.navigate(HomeRoute) {
699                     popUpTo(HomeRoute) { inclusive =
700                         true }
701                 }
702             }
703         )
704     }
705     //=====
706     // APP ROUTE - Main application entry point
707     //=====
708     composable<AppRoute> {
709         AppScreen(
710             changeMessage = changeMessage,
711             onSearch = { en, exactEn, vn, exactVn ->
712                 navigation.navigate(
713                     SearchCardsRoute(
714                         englishText = en,
715                         exactEnglish = exactEn,
716                         vietnameseText = vn,
717                         exactVietnamese = exactVn
718                     )
719                 )
720             }
721         )
722     }
723     //=====
724     // HOME ROUTE - Welcome screen
725     //=====
726     composable<HomeRoute> {
727         HomeScreen(
728             changeMessage = changeMessage,
729             onSearch = { en, exactEn, vn, exactVn ->
730                 navigation.navigate(
731                     SearchCardsRoute(
732                         englishText = en,
733                         exactEnglish = exactEn,
734                         vietnameseText = vn,
735                         exactVietnamese = exactVn
736                     )
737                 )
738             }
739         )
740     }
741     //=====
742     // FILTER ROUTE - Search input form
743     //=====
744     composable<FilterRoute> {
745         FilterScreen(
746             changeMessage = changeMessage,
747             onSearch = { en, exactEn, vn, exactVn ->
748                 navigation.navigate(
749                     SearchCardsRoute(
750                         englishText = en,
751                         exactEnglish = exactEn,
752                         vietnameseText = vn,
753                         exactVietnamese = exactVn
754                     )
755                 )
756             }
757         )
758     }
759     //=====
760     // EDIT CARD ROUTE - Edit existing flashcard
761     with audio management
762     //=====
763     composable<EditCardRoute> { backStackEntry ->
764         val route = backStackEntry.toRoute<
765             EditCardRoute>()
766         EditScreen(
767             cardId = route.id,
768             flashCardDao = flashCardDao,
769             networkService = networkService,
770             changeMessage = changeMessage,
771             onCardUpdated = { navigation.navigateUp
772                 () }
773         )
774     }
775     //=====
776     // SHOW CARD ROUTE - View single card and delete
777     if desired
778     //=====
779     /**
780      * Receives cardId from SearchScreen
781      * Displays full card details with delete and
782      * play audio options
783      */
784     composable<ShowCardRoute> { backStackEntry ->
785         val route = backStackEntry.toRoute<
786             ShowCardRoute>()
787         StudyScreen(
788             changeMessage = changeMessage,
789             flashCardDao = flashCardDao,
790             networkService = networkService,
791             mode = CardViewMode.SINGLE_CARD,
792             cardId = route.id,
793             onCardDeleted = { navigation.navigateUp
794                 () }
795         )
796     }
797     //=====
798     // TOKEN ROUTE - Store token from login
799     //=====
800     /**
801      * Receives email parameter from LoginRoute
802      * User enters token string received via email
803      * Saves token to DataStore for later audio API
804      */
805     composable<TokenRoute> { backStackEntry ->
806         val route = backStackEntry.toRoute<
807             TokenRoute>()
808         TokenScreen(
809             email = route.email,
810             changeMessage = changeMessage,
811             navigateToHome = {
812                 navigation.navigate(HomeRoute) {
813                     popUpTo(HomeRoute) { inclusive =
814                         true }
815                 }
816             }
817         )
818     }
819     //=====
820     // APP ROUTE - Main application entry point
821     //=====
822     composable<AppRoute> {
823         AppScreen(
824             changeMessage = changeMessage,
825             onSearch = { en, exactEn, vn, exactVn ->
826                 navigation.navigate(
827                     SearchCardsRoute(
828                         englishText = en,
829                         exactEnglish = exactEn,
830                         vietnameseText = vn,
831                         exactVietnamese = exactVn
832                     )
833                 )
834             }
835         )
836     }
837     //=====
838     // HOME ROUTE - Welcome screen
839     //=====
840     composable<HomeRoute> {
841         HomeScreen(
842             changeMessage = changeMessage,
843             onSearch = { en, exactEn, vn, exactVn ->
844                 navigation.navigate(
845                     SearchCardsRoute(
846                         englishText = en,
847                         exactEnglish = exactEn,
848                         vietnameseText = vn,
849                         exactVietnamese = exactVn
850                     )
851                 )
852             }
853         )
854     }
855     //=====
856     // FILTER ROUTE - Search input form
857     //=====
858     composable<FilterRoute> {
859         FilterScreen(
860             changeMessage = changeMessage,
861             onSearch = { en, exactEn, vn, exactVn ->
862                 navigation.navigate(
863                     SearchCardsRoute(
864                         englishText = en,
865                         exactEnglish = exactEn,
866                         vietnameseText = vn,
867                         exactVietnamese = exactVn
868                     )
869                 )
870             }
871         )
872     }
873     //=====
874     // EDIT CARD ROUTE - Edit existing flashcard
875     with audio management
876     //=====
877     composable<EditCardRoute> { backStackEntry ->
878         val route = backStackEntry.toRoute<
879             EditCardRoute>()
880         EditScreen(
881             cardId = route.id,
882             flashCardDao = flashCardDao,
883             networkService = networkService,
884             changeMessage = changeMessage,
885             onCardUpdated = { navigation.navigateUp
886                 () }
887         )
888     }
889     //=====
890     // SHOW CARD ROUTE - View single card and delete
891     if desired
892     //=====
893     /**
894      * Receives cardId from SearchScreen
895      * Displays full card details with delete and
896      * play audio options
897      */
898     composable<ShowCardRoute> { backStackEntry ->
899         val route = backStackEntry.toRoute<
900             ShowCardRoute>()
901         StudyScreen(
902             changeMessage = changeMessage,
903             flashCardDao = flashCardDao,
904             networkService = networkService,
905             mode = CardViewMode.SINGLE_CARD,
906             cardId = route.id,
907             onCardDeleted = { navigation.navigateUp
908                 () }
909         )
910     }
911     //=====
912     // TOKEN ROUTE - Store token from login
913     //=====
914     /**
915      * Receives email parameter from LoginRoute
916      * User enters token string received via email
917      * Saves token to DataStore for later audio API
918      */
919     composable<TokenRoute> { backStackEntry ->
920         val route = backStackEntry.toRoute<
921             TokenRoute>()
922         TokenScreen(
923             email = route.email,
924             changeMessage = changeMessage,
925             navigateToHome = {
926                 navigation.navigate(HomeRoute) {
927                     popUpTo(HomeRoute) { inclusive =
928                         true }
929                 }
930             }
931         )
932     }
933     //=====
934     // APP ROUTE - Main application entry point
935     //=====
936     composable<AppRoute> {
937         AppScreen(
938             changeMessage = changeMessage,
939             onSearch = { en, exactEn, vn, exactVn ->
940                 navigation.navigate(
941                     SearchCardsRoute(
942                         englishText = en,
943                         exactEnglish = exactEn,
944                         vietnameseText = vn,
945                         exactVietnamese = exactVn
946                     )
947                 )
948             }
949         )
950     }
951     //=====
952     // HOME ROUTE - Welcome screen
953     //=====
954     composable<HomeRoute> {
955         HomeScreen(
956             changeMessage = changeMessage,
957             onSearch = { en, exactEn, vn, exactVn ->
958                 navigation.navigate(
959                     SearchCardsRoute(
960                         englishText = en,
961                         exactEnglish = exactEn,
962                         vietnameseText = vn,
963                         exactVietnamese = exactVn
964                     )
965                 )
966             }
967         )
968     }
969     //=====
970     // FILTER ROUTE - Search input form
971     //=====
972     composable<FilterRoute> {
973         FilterScreen(
974             changeMessage = changeMessage,
975             onSearch = { en, exactEn, vn, exactVn ->
976                 navigation.navigate(
977                     SearchCardsRoute(
978                         englishText = en,
979                         exactEnglish = exactEn,
980                         vietnameseText = vn,
981                         exactVietnamese = exactVn
982                     )
983                 )
984             }
985         )
986     }
987     //=====
988     // EDIT CARD ROUTE - Edit existing flashcard
989     with audio management
990     //=====
991     composable<EditCardRoute> { backStackEntry ->
992         val route = backStackEntry.toRoute<
993             EditCardRoute>()
994         EditScreen(
995             cardId = route.id,
996             flashCardDao = flashCardDao,
997             networkService = networkService,
998             changeMessage = changeMessage,
999             onCardUpdated = { navigation.navigateUp
1000                () }
1001            )
1002        }
1003    }
1004}

```

## 4.1.2 Routes.kt

```
1 package com.example.menuannam.presentation.navigation
2
3 import kotlinx.serialization.Serializable
4
5 // TYPE-SAFE NAVIGATION ROUTES: @Serializable objects for
6 // compile-time verification
7 // Kotlinx.serialization handles automatic type conversion
8 // for route parameters
9
10 @Serializable
11 object HomeRoute // Main menu: Study, Add Cards, Search,
12   Login
13
14 @Serializable
15 object AddCardRoute // Create new flashcard with English/
16   Vietnamese input
17
18 @Serializable
19 data class SearchCardsRoute( // Display filtered search
20   results with edit/delete options
21   val englishText: String = "", // English search term
22   val exactEnglish: Int = 0, // 1 for exact match, 0 for
23   partial match
24
25 @Serializable
26 object LoginRoute // User enters email to get token for
27   audio synthesis
28
29 @Serializable
30 data class ShowCardRoute(val id: Int) // View single
31   flashcard with delete and audio playback
32
33 @Serializable
34 data class EditCardRoute(val id: Int) // Edit existing
35   flashcard by ID
36
37 @Serializable
38 data class TokenRoute(val email: String) // Token input
39   screen - email passed from LoginRoute for context
40
41 @Serializable
42 object FilterRoute // Filter search screen with English/
43   Vietnamese terms and exact/partial match options
44
45 ./app/src/main/java/com/example/menuannam/presentation/navi
```

## 4.2 Components

### 4.2.1 TopBarComponent.kt

```
1 package com.example.menuannam.presentation.components
2
3 import androidx.compose.ui.semantics.contentDescription
4 import androidx.compose.material3.ButtonDefaults
5 import androidx.compose.material3.CenterAlignedTopAppBar
6 import androidx.compose.material3.ExperimentalMaterial3Api
7 import androidx.compose.material3.MaterialTheme
8 import androidx.compose.material3.Text
9 import androidx.compose.material3.TextButton
10 import androidx.compose.material3.TopAppBarDefaults
11 import androidx.compose.runtime.Composable
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.semantics.semantics
14
15 @OptIn(ExperimentalMaterial3Api::class)
16 @Composable
17 fun TopBarComponent(
18   title: String, // Screen name to display (e.g., "Study
19   Cards", "Add Card")
20   showBack: (() -> Unit)? = null, // Optional callback to
21   navigate up - if null, no back button shown
22 ) {
23   CenterAlignedTopAppBar(
24     title = { Text(title) },
25     navigationIcon = {
26       if (showBack != null) { // Conditional back
27         TextButton(
28           onClick = showBack,
```

### 4.2.2 BottomBarComponent.kt

```
1 package com.example.menuannam.presentation.components
2
3 import androidx.compose.foundation.layout.fillMaxWidth
4 import androidx.compose.material3.BottomAppBar
5 import androidx.compose.material3.Text
6 import androidx.compose.runtime.Composable
7 import androidx.compose.ui.Modifier
8 import androidx.compose.ui.semantics.contentDescription
9 import androidx.compose.ui.semantics.semantics
10 import androidx.compose.ui.text.style.TextAlign
11
12 @Composable
13 fun BottomBarComponent(
14   message: String // Current status message (updated by
15   Navigator's changeMessage callback)
```

```
21   val vietnameseText: String = "", // Vietnamese search
22   term
23   val exactVietnamese: Int = 0 // 1 for exact match, 0 for
24   partial match
25 )
26
27 @Serializable
28 object LoginRoute // User enters email to get token for
29   audio synthesis
30
31 @Serializable
32 data class ShowCardRoute(val id: Int) // View single
33   flashcard with delete and audio playback
34
35 @Serializable
36 data class EditCardRoute(val id: Int) // Edit existing
37   flashcard by ID
38
39 @Serializable
40 data class TokenRoute(val email: String) // Token input
41   screen - email passed from LoginRoute for context
42
43 @Serializable
44 object FilterRoute // Filter search screen with English/
45   Vietnamese terms and exact/partial match options
46
47 ./app/src/main/java/com/example/menuannam/presentation/navi
```

```
27
28   modifier = Modifier.semantics { // Enables testing framework to find elements
29     contentDescription = "navigateBack"
30   },
31   colors = ButtonDefaults.textButtonColors(
32     contentColor = MaterialTheme.
33       colorScheme.onPrimary
34   )
35   ) {
36     Text("Back")
37   }
38   colors = TopAppBarDefaults.topAppBarColors(
39     containerColor = MaterialTheme.colorScheme.
40       primary,
41     titleContentColor = MaterialTheme.colorScheme.
42       onPrimary,
43     navigationIconContentColor = MaterialTheme.
44       colorScheme.onPrimary,
45     actionIconContentColor = MaterialTheme.
46       colorScheme.onPrimary
47   )
48 }
49
50 ./app/src/main/java/com/example/menuannam/presentation/com
```

```
15 ) {
16   BottomAppBar(){
17     Text( // Shows real-time feedback: "Card 1 of 5", "
18       Logged out", "Error loading flashcards: ..."
19     modifier = Modifier
20       .fillMaxWidth()
21       .semantics { contentDescription = "Message" }
22     ),
23     textAlign = TextAlign.Center,
24     text = message
25   }
26
27 ./app/src/main/java/com/example/menuannam/presentation/com
```

## 4.3 Screens

### 4.3.1 MenuScreen.kt

```
1 package com.example.menuannam.presentation.screens
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.fillMaxWidth
6 import androidx.compose.foundation.layout.fillMaxWidth
7 import androidx.compose.foundation.layout.padding
8 import androidx.compose.material3.Button
9 import androidx.compose.material3.ExperimentalMaterial3Api
10 import androidx.compose.material3.Text
11 import androidx.compose.runtime.Composable
12 import androidx.compose.runtime.LunchedEffect
13 import androidx.compose.runtime.getValue
14 import androidx.compose.runtime.mutableStateOf
15 import androidx.compose.runtime.remember
16 import androidx.compose.runtime.rememberCoroutineScope
17 import androidx.compose.runtime.setValue
18 import androidx.compose.ui.Alignment
19 import androidx.compose.ui.Modifier
20 import androidx.compose.ui.platform.LocalContext
21 import androidx.compose.ui.semantics.contentDescription
22 import androidx.compose.ui.semantics.semantics
23 import androidx.compose.ui.unit.dp
24 import androidx.datastore.preferences.core.edit
25 import com.example.menuannam.EMAIL
26 import com.example.menuannam.TOKEN
27 import com.example.menuannam.dataStore
28 import kotlinx.coroutines.flow.first
29 import kotlinx.coroutines.launch
30
31 @OptIn(ExperimentalMaterial3Api::class)
32 @Composable
33 fun MenuScreen(
34     onStudy: () -> Unit, // Navigate to StudyCardsRoute
35     onAdd: () -> Unit, // Navigate to AddCardRoute
36     onSearch: () -> Unit, // Navigate to SearchCardsRoute
37     onLogin: () -> Unit, // Navigate to LoginRoute
38     changeMessage: (String) -> Unit = {} // Update bottom
39 ) {
40     val context = LocalContext.current
41     val appContext = context.applicationContext
42     val scope = rememberCoroutineScope()
43
44     var email by remember { mutableStateOf("") }
45
46     // Load saved email from DataStore on screen load -
47     // shows user they are logged in if email stored
48     LaunchedEffect(Unit) {
49         val prefs = appContext.dataStore.data.first()
50         email = prefs[EMAIL] ?: ""
51         changeMessage("Email loaded: $email")
52     }
53
54     Column(
55         modifier = Modifier
56             .fillMaxSize()
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102 }
```

```
        .padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(16.dp,
        Alignment.CenterVertically),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Button(
            onClick = onStudy,
            modifier = Modifier.fillMaxWidth()
                .semantics { contentDescription = "navigateToStudyCards" }
        ) { Text("Study Cards") }

        Button(
            onClick = onAdd,
            modifier = Modifier.fillMaxWidth()
                .semantics { contentDescription = "navigateToAddCard" }
        ) { Text("Add Cards") }

        Button(
            onClick = onSearch,
            modifier = Modifier.fillMaxWidth()
                .semantics { contentDescription = "navigateToSearchCards" }
        ) { Text("Search Cards") }

        Button(
            onClick = onLogin,
            modifier = Modifier.fillMaxWidth()
                .semantics { contentDescription = "navigateToLoginScreen" }
        ) { Text("Login") }

        Button(
            onClick = {
                scope.launch {
                    appContext.dataStore.edit {
                        it.remove(EMAIL)
                        it.remove(TOKEN)
                    }
                    email = ""
                    changeMessage("Logged out")
                }
            },
            modifier = Modifier.fillMaxWidth()
                .semantics { contentDescription = "ExecuteLogout" }
        ) {
            Text("Log out", modifier = Modifier.semantics {
                contentDescription = "Logout"
            })
        }
    }
}
./app/src/main/java/com/example/menuannam/presentation/scre
```

### 4.3.2 LoginScreen.kt

```
1 package com.example.menuannam.presentation.screens
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.fillMaxWidth
6 import androidx.compose.foundation.layout.padding
7 import androidx.compose.material3.Button
8 import androidx.compose.material3.OutlinedTextField
9 import androidx.compose.material3.Text
10 import androidx.compose.runtime.Composable
11 import androidx.compose.runtime.LunchedEffect
12 import androidx.compose.runtime.getValue
13 import androidx.compose.runtime.mutableStateOf
14 import androidx.compose.runtime.rememberCoroutineScope
15 import androidx.compose.runtime.saveable.rememberSaveable
16 import androidx.compose.runtime.setValue
17 import androidx.compose.ui.Modifier
18 import androidx.compose.ui.semantics.contentDescription
19 import androidx.compose.ui.semantics.semantics
20 import androidx.compose.ui.unit.dp
21 import com.example.menuannam.data.network.UserCredential
22 import com.example.menuannam.data.network.NetworkService
23 import kotlinx.coroutines.Dispatchers
24 import kotlinx.coroutines.launch
25 import kotlinx.coroutines.withContext
26
27 @Composable
28 fun LoginScreen(
29     changeMessage: (String) -> Unit, // Updates status bar
```

```
30     with feedback
31     networkService: NetworkService, // Retrofit interface
32     for Lambda API
33     navigateToToken: (String) -> Unit // Callback to show
34     TokenScreen with email
35 ) {
36     var email by rememberSaveable { mutableStateOf("") }
37     val scope = rememberCoroutineScope()
38
39     LaunchedEffect(Unit) {
40         changeMessage("Please, introduce your email.")
41     }
42
43     Column(
44         modifier = Modifier.padding(16.dp),
45         verticalArrangement = Arrangement.spacedBy(16.dp)
46     ) {
47         OutlinedTextField(
48             value = email,
49             onValueChange = { email = it },
50             modifier = Modifier
51                 .fillMaxWidth()
52                 .semantics { contentDescription = "emailTextField" },
53             label = { Text("email") }
54         )
55
56         Button(
57             modifier = Modifier
```

```
55         .fillMaxWidth()
56         .semantics { contentDescription = "Enter" },
57     onClick = {
58         scope.launch {
59             try { // Send email to AWS Lambda for
60                 token generation
61                 val result = withContext(Dispatchers
62                 .IO) {
63                     networkService.generateToken(
64                         email = UserCredential(email))
65                     if (result.code == 200) { // Response code 200 means success
66                         changeMessage("Token sent to
67                         email: ${result.message}")
68                         navigateToToken(email)
69                     } else {
70                         changeMessage("Error: ${result.
71                         message}")
72                     }
73                 } catch (e: Exception) {
74                     changeMessage("There was an error in
75                     the token request.")
76                 }
77             }
78         }
79     }
80 }
```

### 4.3.3 TokenScreen.kt

```
1 package com.example.menuannam.presentation.screens
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.fillMaxSize
6 import androidx.compose.foundation.layout.fillMaxWidth
7 import androidx.compose.foundation.layout.padding
8 import androidx.compose.material3.Button
9 import androidx.compose.material3.OutlinedTextField
10 import androidx.compose.material3.Text
11 import androidx.compose.runtime.Composable
12 import androidx.compose.runtime.LaunchedEffect
13 import androidx.compose.runtime.getValue
14 import androidx.compose.runtime.mutableStateOf
15 import androidx.compose.runtime.remember
16 import androidx.compose.runtime.rememberCoroutineScope
17 import androidx.compose.runtime.setValue
18 import androidx.compose.ui.Modifier
19 import androidx.compose.ui.platform.LocalContext
20 import androidx.compose.ui.semantics.contentDescription
21 import androidx.compose.ui.semantics.semantics
22 import androidx.compose.ui.unit.dp
23 import androidx.datastore.preferences.core.edit
24 import com.example.menuannam.EMAIL
25 import com.example.menuannam.TOKEN
26 import com.example.menuannam.dataStore
27 import kotlinx.coroutines.Dispatchers
28 import kotlinx.coroutines.launch
29 import kotlinx.coroutines.withContext
30
31 @Composable
32 fun TokenScreen(
33     email: String, // User email from LoginScreen, displayed
34     // as context
35     changeMessage: (String) -> Unit, // Updates status bar
36     // with feedback
37     navigateToHome: (String) -> Unit // Return to main menu
38     // after save
39 ) {
40     val scope = rememberCoroutineScope()
41     val context = LocalContext.current
42     val applicationContext = context.applicationContext
43     var token by remember { mutableStateOf("") }
44
45     LaunchedEffect(Unit) {
46         changeMessage("Please, introduce your token.")
47     }
48
49     Column(
```

```
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    ) {
        OutlinedTextField(
            value = token,
            onValueChange = { token = it },
            modifier = Modifier
                .fillMaxWidth()
                .semantics { contentDescription = "tokenTextField" },
            label = { Text("token") }
        )
        OutlinedTextField(
            value = email,
            onValueChange = {},
            modifier = Modifier
                .fillMaxWidth()
                .semantics { contentDescription = "emailTextField" },
            label = { Text("email") },
            readOnly = true
        )
        Button(
            modifier = Modifier
                .fillMaxWidth()
                .semantics { contentDescription = "Enter" },
            onClick = {
                scope.launch {
                    withContext(Dispatchers.IO) { // Save
                        token and email to DataStore for future audio requests
                        applicationContext.dataStore.edit {
                            preferences ->
                                preferences[EMAIL] = email
                                preferences[TOKEN] = token
                            }
                        }
                    navigateToHome(token)
                }
            }
        ) {
            Text("Enter")
        }
    }
}
```

#### 4.3.4 AddScreen.kt

```

1 package com.example.menuannam.presentation.screens
2
3 import androidx.compose.foundation.layout.*
4 import androidx.compose.material3.*
5 import androidx.compose.runtime.*
6 import androidx.compose.runtime.saveable.rememberSaveable
7 import androidx.compose.ui.Alignment
8 import androidx.compose.ui.Modifier
9 import androidx.compose.ui.semantics.contentDescription
10 import androidx.compose.ui.semantics.editableText
11 import androidx.compose.ui.semantics.semantics
12 import androidx.compose.ui.semantics.text
13 import androidx.compose.ui.text.AnnotatedString
14 import androidx.compose.ui.unit.dp
15 import com.example.menuannam.data.entity.FlashCard
16
17 @OptIn(ExperimentalMaterial3Api::class)
18 @Composable
19 fun AddScreen(
20     changeMessage: (String) -> Unit, // Updates status bar
21     with feedback
22     insertFlashCard: (FlashCard) -> Unit // Callback to save
23     card (receives FlashCard)
24 ) {
25     var english by rememberSaveable { mutableStateOf("") }
26     // TextFields persist state across recomposition
27     var vietnamese by rememberSaveable { mutableStateOf("") }
28
29     LaunchedEffect(Unit) {
30         changeMessage("Add a new flashcard")
31     }
32
33     Column(
34         modifier = Modifier
35             .fillMaxSize()
36             .padding(16.dp),
37         verticalArrangement = Arrangement.spacedBy(16.dp,
38             Alignment.CenterVertically),
39         horizontalAlignment = Alignment.CenterHorizontally
40     ) {
41
42         TextField(
43             value = english,
44             onValueChange = { english = it },
45             label = { Text("English") },
46             modifier = Modifier
47                 .fillMaxWidth()
48                 .semantics { contentDescription = "enTextField" }
49
50         )
51
52         TextField(
53             value = vietnamese,
54             onValueChange = { vietnamese = it },
55             label = { Text("Vietnamese") },
56             modifier = Modifier
57                 .fillMaxWidth()
58                 .semantics { contentDescription = "viTextField" }
59
60         )
61
62         Row(
63             modifier = Modifier.fillMaxWidth(),
64             horizontalArrangement = Arrangement.spacedBy(16.dp,
65                 Alignment.CenterHorizontally),
66             verticalAlignment = Alignment.CenterVertically
67         ) {
68             Button(
69                 onClick = {
70                     try { // Room handles OnConflictStrategy.IGNORE (silently ignores duplicates)
71                         insertFlashCard(FlashCard(0, english
72                             , vietnamese))
73                         changeMessage("Flash card
74                             successfully added to your database.")
75                         english = ""
76                         vietnamese = ""
77                     } catch (e: Exception) { // Duplicate detection
78                         changeMessage("Flash card already
79                             exists in your database.")
80                     }
81                 },
82                 modifier = Modifier.semantics {
83                     contentDescription = "Add"
84                 }
85                 Text("Add")
86
87             )
88             Button(
89                 onClick = {
90                     english = ""
91                     vietnamese = ""
92                 },
93                 modifier = Modifier.semantics {
94                     contentDescription = "Clear"
95                 }
96                 Text("Clear")
97             )
98         }
99     }
100 }
101
102 ./app/src/main/java/com/example/menuannam/presentation/screen

```

### 4.3.5 FilterScreen.kt

```

1 package com.example.menuannam.presentation.screens
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.Spacer
7 import androidx.compose.foundation.layout.fillMaxWidth
8 import androidx.compose.foundation.layout.padding
9 import androidx.compose.foundation.layout.size
10 import androidx.compose.material3.Button
11 import androidx.compose.material3.Checkbox
12 import androidx.compose.material3.OutlinedTextField
13 import androidx.compose.material3.Text
14 import androidx.compose.runtime.Composable
15 import androidx.compose.runtime.LunchedEffect
16 import androidx.compose.runtime.getValue
17 import androidx.compose.runtime.mutableStateOf
18 import androidx.compose.runtime.remember
19 import androidx.compose.runtime.setValue
20 import androidx.compose.ui.Alignment
21 import androidx.compose.ui.Modifier
22 import androidx.compose.ui.semantics.contentDescription
23 import androidx.compose.ui.semantics.semantics
24 import androidx.compose.ui.unit.dp
25
26 @Composable
27 fun FilterScreen(
28     changeMessage: (String) -> Unit = {}, // Updates status
29     bar with feedback
30     onSearch: (en: String, exactEn: Int, vn: String, exactVn
31     : Int) -> Unit // Navigate to SearchCardsRoute with
32     parameters
33 ) {
34     var englishText by remember { mutableStateOf("") } // English search term
35     var vietnameseText by remember { mutableStateOf("") } // Vietnamese search term
36     var exactEnglish by remember { mutableStateOf(false) } // Exact match checkbox for English
37     var exactVietnamese by remember { mutableStateOf(false) } // Exact match checkbox for Vietnamese
38
39     LaunchedEffect(Unit) {
40         changeMessage("Enter search criteria and click
41             Search")
42     }
43
44     Column(
45         modifier = Modifier
46             .fillMaxWidth()
47             .padding(16.dp),
48         verticalArrangement = Arrangement.spacedBy(16.dp),
49         horizontalAlignment = Alignment.CenterHorizontally
50     ) {
51         // English search field with checkbox
52         Row(
53             modifier = Modifier.fillMaxWidth(),
54             verticalAlignment = Alignment.CenterVertically,
55             horizontalArrangement = Arrangement.spacedBy(8.
56             dp)
57         ) {
58             OutlinedTextField(
59                 value = englishText,
60                 onValueChange = { englishText = it },
61                 label = { Text("English") },
62                 modifier = Modifier
63                     .weight(1f)
64                     .semantics { contentDescription = "
65             enSearchField" },
66                 singleLine = true
67             )
68             Checkbox(
69                 checked = exactEnglish,
70                 onCheckedChange = { exactEnglish = it },
71                 modifier = Modifier.semantics {
72                     contentDescription = "enExactCheckbox"
73                 }
74             )
75
76             // Vietnamese search field with checkbox
77             Row(
78                 modifier = Modifier.fillMaxWidth(),
79                 verticalAlignment = Alignment.CenterVertically,
80                 horizontalArrangement = Arrangement.spacedBy(8.
81             dp)
82         ) {
83             OutlinedTextField(
84                 value = vietnameseText,
85                 onValueChange = { vietnameseText = it },
86                 label = { Text("Vietnamese") },
87                 modifier = Modifier
88                     .weight(1f)
89                     .semantics { contentDescription = "
90             vnSearchField" },
91                 singleLine = true
92             )
93             Checkbox(
94                 checked = exactVietnamese,
95                 onCheckedChange = { exactVietnamese = it },
96                 modifier = Modifier.semantics {
97                     contentDescription = "vnExactCheckbox"
98                 }
99             )
100
101             Spacer(modifier = Modifier.size(16.dp))
102
103             // Search button
104             Button(
105                 onClick = {
106                     val en = englishText.trim().ifEmpty { "" }
107                     val vn = vietnameseText.trim().ifEmpty { "" }
108
109                     val exactEn = if (exactEnglish) 1 else 0
110                     val exactVn = if (exactVietnamese) 1 else 0
111
112                     changeMessage("Searching...")
113                     onSearch(en, exactEn, vn, exactVn)
114                 },
115                 modifier = Modifier
116                     .fillMaxWidth()
117                     .semantics { contentDescription = "
118             searchButton" }
119             ) {
120                 Text("Search")
121             }
122
123             // Clear button
124             Button(
125                 onClick = {
126                     englishText = ""
127                     vietnameseText = ""
128                     exactEnglish = false
129                     exactVietnamese = false
130                     changeMessage("Search criteria cleared")
131                 },
132                 modifier = Modifier
133                     .fillMaxWidth()
134                     .semantics { contentDescription = "
135             clearColorButton" }
136             ) {
137                 Text("Clear")
138             }
139         }
140     }
141
142     .app/src/main/java/com/example/menuannam/presentation/scre

```

#### 4.3.6 SearchScreen.kt

```

1 package com.example.menuannam.presentation.screens
2
3 import androidx.compose.foundation.border
4 import androidx.compose.foundation.clickable
5 import androidx.compose.foundation.layout.Arrangement
6 import androidx.compose.foundation.layout.Column
7 import androidx.compose.foundation.layout.Row
8 import androidx.compose.foundation.layout.Spacer
9 import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.padding
11 import androidx.compose.foundation.layout.size
12 import androidx.compose.foundation.layout.width
13 import androidx.compose.foundation.lazy.LazyColumn
14 import androidx.compose.foundation.lazy.items
15 import androidx.compose.material3.Button
16 import androidx.compose.material3.ButtonDefaults
17 import androidx.compose.material3.MaterialTheme
18 import androidx.compose.material3.OutlinedTextField
19 import androidx.compose.material3.Text
20 import androidx.compose.runtime.Composable
21 import androidx.compose.runtime.LaunchedEffect
22 import androidx.compose.runtime.getValue
23 import androidx.compose.runtime.mutableStateOf
24 import androidx.compose.runtime.remember
25 import androidx.compose.runtime.rememberCoroutineScope
26 import androidx.compose.runtime.setValue
27 import androidx.compose.ui.Alignment
28 import androidx.compose.ui.Modifier
29 import androidx.compose.ui.graphics.Color
30 import androidx.compose.ui.semantics.contentDescription
31 import androidx.compose.ui.semantics.semantics
32 import androidx.compose.ui.unit.dp
33 import androidx.compose.ui.unit.sp
34 import kotlinx.coroutines.launch
35 import com.example.menuannam.data.database.FlashCardDao
36 import com.example.menuannam.data.entity.FlashCard
37
38 // FlashCardList component - LazyColumn displaying
39 // flashcards with edit/delete buttons
40 @Composable
41 fun FlashCardList(
42     flashCards: List<FlashCard>,
43     onEdit: (Int) -> Unit = {}, // Navigate to EditCardRoute
44     onDelete: (FlashCard) -> Unit = {} // Delete card from
45     database
46 ) {
47     LazyColumn(
48         modifier = Modifier.padding(16.dp)
49     ) {
50         items(
51             items = flashCards,
52             key = { flashCard ->
53                 flashCard.uid
54             }
55         ) { flashCard ->
56             Row(
57                 modifier = Modifier
58                     .fillMaxWidth()
59                     .border(width = 1.dp, color = Color.
60 LightGray)
61                     .padding(8.dp),
62                     verticalAlignment = Alignment.
63 CenterVertically
64             ) {
65                 Column(
66                     modifier = Modifier
67                         .weight(1f)
68                         .padding(6.dp)
69                 ) {
70                     Text("${flashCard.englishCard ?: ""} = $"
71 ${flashCard.vietnameseCard ?: ""}")
72                     Spacer(modifier = Modifier.width(8.dp))
73                     Row(
74                         horizontalArrangement = Arrangement.
75 spacedBy(4.dp)
76                     ) {
77                         Button(
78                             onClick = { onEdit(flashCard.uid) },
79                             colors = ButtonDefaults.buttonColors
80 (containerColor = MaterialTheme.colorScheme.primary)
81                     ) {
82                         Text("Edit", fontSize = 12.sp)
83                     }
84                     Button(
85                         onClick = { onDelete(flashCard) },
86                         colors = ButtonDefaults.buttonColors
87 (containerColor = MaterialTheme.colorScheme.error)
88                     ) {
89                         Text("Delete", fontSize = 12.sp)
90                     }
91                 }
92             }
93         }
94     }
95     @Composable
96     fun SearchScreen(
97         changeMessage: (String) -> Unit = {},
98         flashCardDao: FlashCardDao,
99         onEdit: (Int) -> Unit = {},
100        englishText: String = "",
101        exactEnglish: Int = 0,
102        vietnameseText: String = "",
103        exactVietnamese: Int = 0
104    ) {
105        var filteredCards by remember { mutableStateOf<List<
106 FlashCard>>(emptyList()) }
107        var isLoading by remember { mutableStateOf(false) }
108        val coroutineScope = rememberCoroutineScope()
109
110        val performSearch = {
111            coroutineScope.launch {
112                try {
113                    isLoading = true
114                    filteredCards = flashCardDao.
115 getFilteredFlashCards(englishText, exactEnglish,
116 vietnameseText, exactVietnamese)
117                    changeMessage("Found ${filteredCards.size}
118 cards")
119                    isLoading = false
120                } catch (e: Exception) {
121                    changeMessage("Error: ${e.message}")
122                    isLoading = false
123                }
124            }
125
126            LaunchedEffect(Unit) {
127                performSearch()
128            }
129
130            Column(
131                modifier = Modifier
132                    .fillMaxWidth()
133                    .padding(16.dp),
134                    horizontalAlignment = Alignment.CenterHorizontally,
135                    verticalArrangement = Arrangement.spacedBy(12.dp),
136            ) {
137                // Results display
138                if (isLoading) {
139                    Text("Searching...")
140                } else if (filteredCards.isEmpty()) {
141                    Text("No cards found. Try different search terms
142 .")
143                } else {
144                    FlashCardList(
145                        flashCards = filteredCards,
146                        onEdit = onEdit,
147                        onDelete = { cardToDelete ->
148                            coroutineScope.launch {
149                                try {
150                                    flashCardDao.delete(cardToDelete)
151                                } catch (e: Exception) {
152                                    changeMessage("Card deleted
153 successfully!")
154                                }
155                            }
156                        )
157                    }
158                }
159            }
160        }
161
162        .app/src/main/java/com/example/menuannam/presentation/scre

```

#### 4.3.7 EditScreen.kt

```

1 package com.example.menuannam.presentation.screens
2
3 import android.util.Base64
4 import androidx.compose.foundation.layout.Arrangement
5 import androidx.compose.foundation.layout.Column
6 import androidx.compose.foundation.layout.Spacer
7 import androidx.compose.foundation.layout.fillMaxWidth
8 import androidx.compose.foundation.layout.padding
9 import androidx.compose.foundation.layout.size
10 import androidx.compose.material3.Button
11 import androidx.compose.material3.ButtonDefaults
12 import androidx.compose.material3.MaterialTheme
13 import androidx.compose.material3.OutlinedTextField
14 import androidx.compose.material3.Text
15 import androidx.compose.runtime.Composable
16 import androidx.compose.runtime.LaunchedEffect
17 import androidx.compose.runtime.getValue
18 import androidx.compose.runtime.mutableStateOf
19 import androidx.compose.runtime.remember
20 import androidx.compose.runtime.rememberCoroutineScope
21 import androidx.compose.runtime.setValue
22 import androidx.compose.ui.Alignment
23 import androidx.compose.ui.Modifier
24 import androidx.compose.ui.platform.LocalContext
25 import androidx.compose.ui.semantics.contentDescription
26 import androidx.compose.ui.semantics.semantics
27 import androidx.compose.ui.unit.dp
28 import android.net.Uri
29 import androidx.media3.common.MediaItem
30 import androidx.media3.exoplayer.ExoPlayer
31 import com.example.menuannam.EMAIL
32 import com.example.menuannam.TOKEN
33 import com.example.menuannam.data.database.FlashCardDao
34 import com.example.menuannam.data.entity.FlashCard
35 import com.example.menuannam.data.network.AudioRequest
36 import com.example.menuannam.data.network.NetworkService
37 import com.example.menuannam.dataStore
38 import com.example.menuannam.saveAudioToInternalStorage
39 import com.example.menuannam.toMd5
40 import kotlinx.coroutines.flow.first
41 import kotlinx.coroutines.launch
42 import java.io.File
43
44 @Composable
45 fun EditScreen(
46     cardId: Int, // Flashcard ID to edit
47     flashCardDao: FlashCardDao, // Database access
48     networkService: NetworkService, // API for audio
49     generation
50     changeMessage: (String) -> Unit, // Status message
51     callback
52     onCardUpdated: () -> Unit // Navigate back after
53     successful update
54 ) {
55     val context = LocalContext.current
56     val appContext = context.applicationContext
57     val coroutineScope = rememberCoroutineScope()
58
59     var flashCard by remember { mutableStateOf<FlashCard?>(
60         null
61     ) }
62     var englishText by remember { mutableStateOf("") }
63     var vietnameseText by remember { mutableStateOf("") }
64     var audioExists by remember { mutableStateOf(false) }
65     var audioFilePath by remember { mutableStateOf("") }
66     var isLoading by remember { mutableStateOf(true) }
67
68     // Load flashcard and check audio existence on screen
69     load
70     LaunchedEffect(cardId) {
71         try {
72             val card = flashCardDao.getById(cardId)
73             if (card != null) {
74                 flashCard = card
75                 englishText = card.englishCard ?: ""
76                 vietnameseText = card.vietnameseCard ?: ""
77
78                 // Check if audio file exists for current
79                 Vietnamese word
80                 val vn = card.vietnameseCard ?: ""
81                 if (vn.isNotBlank()) {
82                     val fileName = "${vn.toMd5()}.mp3"
83                     val file = File(appContext.filesDir,
84                         fileName)
85                     audioExists = file.exists()
86                     audioFilePath = if (audioExists) file.
87                     absolutePath else ""
88
89                     changeMessage("Please, edit the flashcard.")
90                 } else {
91                     changeMessage("Flash card not found")
92                 }
93             } catch (e: Exception) {
94                 changeMessage("Error loading flash card: ${e.
95                 message}")
96             }
97         } finally {
98             isLoading = false
99         }
100    }
101
102    if (isLoading) {
103        Column(
104            modifier = Modifier
105                .fillMaxWidth()
106                .padding(16.dp),
107                horizontalAlignment = Alignment.
108                CenterHorizontally
109        ) {
110            Text("Loading...")
111        }
112    } else if (flashCard == null) {
113        Column(
114            modifier = Modifier
115                .fillMaxWidth()
116                .padding(16.dp),
117                horizontalAlignment = Alignment.
118                CenterHorizontally
119        ) {
120            Text(
121                text = "Flash card not found",
122                color = MaterialTheme.colorScheme.error
123            )
124        }
125    } else {
126        Column(
127            modifier = Modifier
128                .fillMaxWidth()
129                .padding(16.dp),
130                verticalArrangement = Arrangement.spacedBy(12.dp)
131        ),
132        horizontalAlignment = Alignment.
133        CenterHorizontally
134    ) {
135        // English text field
136        OutlinedTextField(
137            value = englishText,
138            onValueChange = { englishText = it },
139            label = { Text("en") },
140            modifier = Modifier
141                .fillMaxWidth()
142                .semantics { contentDescription = "enTextField" }
143        )
144
145        // Vietnamese text field
146        OutlinedTextField(
147            value = vietnameseText,
148            onValueChange = { vietnameseText = it },
149            label = { Text("vn") },
150            modifier = Modifier
151                .fillMaxWidth()
152                .semantics { contentDescription = "vnTextField" }
153
154        )
155
156        // Audio file path display (if exists)
157        if (audioExists) {
158            OutlinedTextField(
159                value = audioFilePath,
160                onValueChange = {},
161                label = { Text("audio") },
162                readOnly = true,
163                modifier = Modifier.fillMaxWidth(),
164                singleLine = false,
165                maxLines = 3
166            )
167        }
168
169        Spacer(modifier = Modifier.size(4.dp))
170
171        // Update flashcard button - saves changes to
172        database
173        Button(
174            onClick = {
175                coroutineScope.launch {
176                    try {
177                        flashCardDao.update(
178                            id = cardId,
179                            english = englishText,
180                            vietnamese = vietnameseText
181                        )
182                        changeMessage("Flash card
183                        updated successfully")
184                    } catch (e: Exception) {
185                        changeMessage("Error updating
186                        flash card: ${e.message}")
187                    }
188                }
189            }
190        ),
```

```

173         modifier = Modifier
174             .fillMaxWidth()
175             .semantics { contentDescription = "updateButton" }
176     ) {
177         Text("Update flashcard")
178     }
179
180     // Conditional audio buttons based on file
181     existence
182     if (audioExists) { // If audio exists: "Clean
183     audio" and "Play audio"
184     // Clean audio button - deletes audio file
185     from storage
186     Button(
187         onClick = {
188             coroutineScope.launch {
189                 try {
190                     val file = File(
191                         audioFilePath)
192                     if (file.exists() && file.
193                         delete()) {
194                         audioExists = false
195                         audioFilePath = ""
196                         changeMessage("Audio
197                         deleted successfully")
198                     } else {
199                         changeMessage("Failed to
200                         delete audio")
201                     }
202                 } catch (e: Exception) {
203                     changeMessage("Error
204                         deleting audio: ${e.message}")
205                 }
206             },
207             modifier = Modifier
208                 .fillMaxWidth()
209                 .semantics { contentDescription = "cleanAudioButton" },
210                 colors = ButtonDefaults.buttonColors(
211                     containerColor = MaterialTheme.
212                     colorScheme.error
213                 )
214             ) {
215                 Text("Clean audio")
216             }
217
218             // Play audio button
219             Button(
220                 onClick = {
221                     coroutineScope.launch {
222                         try {
223                             changeMessage("Playing audio
224                             ...")
225                         val mediaItem = MediaItem.
226                             fromUri(audioFilePath.toUri())
227                         val player = ExoPlayer.
228                             Builder(appContext).build()
229                             player.addListener(object :
230                                 androidx.media3.common.Player.Listener {
231                                     override fun
232                                         onPlaybackStateChanged(playbackState: Int) {
233                                             when (playbackState) {
234                                                 common.Player.STATE_BUFFERING ->
235                                                 androidx.media3.
236                                                 common.Player.STATE_READY ->
237                                                 androidx.media3.
238                                                 common.Player.STATE_PLAYING ->
239                                                 androidx.media3.
240                                                 common.Player.STATE_ENDED -> {
241                                                     player.
242                                                     release()
243
244                                                 changeMessage("Finished")
245
246                                                 else -> {}
247
248                                             }
249                                         }
250                                         player.setMediaItem(
251                                             mediaItem)
252                                         player.prepare()
253                                         player.play()
254
255                         }
256                     } catch (e: Exception) {
257                         changeMessage("Error
258                             playing
259                             audio: ${e.message}"))
260
261             },
262             modifier = Modifier
263                 .fillMaxWidth()
264                 .semantics { contentDescription = "playAudioButton" }
265         ) {
266             Text("Play audio")
267         }
268     } else { // If audio doesn't exist: "Generate
269     audio"
270     // Generate audio button - calls API to
271     synthesize audio
272     Button(
273         onClick = {
274             coroutineScope.launch {
275                 try { // Load token and email
276                     from DataStore for audio API request
277                     val prefs = appContext.
278                         dataStore.data.first()
279                     ""
280                     val email = prefs[EMAIL] ?: ""
281                     val token = prefs[TOKEN] ?: ""
282                     val word = vietnameseText.
283                         trim()
284
285                     if (word.isBlank() || email.
286                         isBlank() || token.isBlank()) { // Missing required
287                         data
288                         changeMessage("Missing
289                         data to request audio")
290                     }
291                     return@launch
292                 }
293
294                 changeMessage("Generating
295                 audio...")
296
297                 val resp = networkService.
298                     generateAudio( // Send Vietnamese word + token to AWS
299                         Lambda
300                         request = AudioRequest(
301                             word, email, token)
302
303                         if (resp.code == 200) { // /
304                             Response code 200 = success
305                             val audioBytes = Base64.
306                             decode(resp.message, Base64.DEFAULT) // Decode Base64-
307                             encoded MP3
308                             val fileName = "${word.
309                             toMd5()}.mp3"
310                             val file =
311                             saveAudioToInternalStorage(appContext, audioBytes,
312                             fileName)
313                             audioExists = true
314                             audioFilePath = file.
315                             absolutePath
316                             changeMessage("Audio
317                             generated successfully")
318                         } else { // 500 = token
319                             invalid/expired
320                             changeMessage("Audio
321                             generation failed (${resp.code}): ${resp.message}")
322                         }
323                     } catch (e: Exception) {
324                         changeMessage("Error
325                         generating audio: ${e.message}")
326                     }
327
328                 },
329                 modifier = Modifier
330                     .fillMaxWidth()
331                     .semantics { contentDescription = "generateAudioButton" }
332             ) {
333                 Text("Generate audio")
334             }
335         }
336     }
337
338     }
339
340     }
341
342     }
343
344     }
345
346     }
347
348     }
349
350     }
351
352     }
353
354     }
355
356     }
357
358     }
359
360     }
361
362     }
363
364     }
365
366     }
367
368     }
369
370     }
371
372     }
373
374     }
375
376     }
377
378     }
379
380     }
381
382     }
383
384     }
385
386     }
387
388     }
389
390     }
391
392     }
393
394     }
395
396     }
397
398     }
399
400     }
401
402     }
403
404     }
405
406     }
407
408     }
409
410     }
411
412     }
413
414     }
415
416     }
417
418     }
419
420     }
421
422     }
423
424     }
425
426     }
427
428     }
429
430     }
431
432     }
433
434     }
435
436     }
437
438     }
439
440     }
441
442     }
443
444     }
445
446     }
447
448     }
449
450     }
451
452     }
453
454     }
455
456     }
457
458     }
459
460     }
461
462     }
463
464     }
465
466     }
467
468     }
469
470     }
471
472     }
473
474     }
475
476     }
477
478     }
479
480     }
481
482     }
483
484     }
485
486     }
487
488     }
489
490     }
491
492     }
493
494     }
495
496     }
497
498     }
499
500     }
501
502     }
503
504     }
505
506     }
507
508     }
509
510     }
511
512     }
513
514     }
515
516     }
517
518     }
519
520     }
521
522     }
523
524     }
525
526     }
527
528     }
529
530     }
531
532     }
533
534     }
535
536     }
537
538     }
539
540     }
541
542     }
543
544     }
545
546     }
547
548     }
549
550     }
551
552     }
553
554     }
555
556     }
557
558     }
559
560     }
561
562     }
563
564     }
565
566     }
567
568     }
569
570     }
571
572     }
573
574     }
575
576     }
577
578     }
579
580     }
581
582     }
583
584     }
585
586     }
587
588     }
589
590     }
591
592     }
593
594     }
595
596     }
597
598     }
599
599 ./app/src/main/java/com/example/menuannam/presentation/screen

```

### 4.3.8 StudyScreen.kt

```

1 package com.example.menuannam.presentation.screens
2
3 import androidx.compose.foundation.clickable
4 import androidx.compose.foundation.layout.Arrangement
5 import androidx.compose.foundation.layout.Box
6 import androidx.compose.foundation.layout.Column
7 import androidx.compose.foundation.layout.Row
8 import androidx.compose.foundation.layout.Spacer
9 import androidx.compose.foundation.layout.fillMaxSize
10 import androidx.compose.foundation.layout.fillMaxWidth
11 import androidx.compose.foundation.layout.padding
12 import androidx.compose.foundation.layout.size
13 import androidx.compose.material3.Button
14 import androidx.compose.material3.Card
15 import androidx.compose.material3.CardDefaults
16 import androidx.compose.material3.MaterialTheme
17 import androidx.compose.material3.Text
18 import androidx.compose.runtime.Composable
19 import androidx.compose.runtime.LaunchedEffect
20 import androidx.compose.runtime.getValue
21 import androidx.compose.runtime.mutableIntStateOf
22 import androidx.compose.runtime.mutableStateOf
23 import androidx.compose.runtime.remember
24 import androidx.compose.runtime.rememberCoroutineScope
25 import androidx.compose.runtime.setValue
26 import androidx.compose.ui.Alignment
27 import androidx.compose.ui.Modifier
28 import androidx.compose.ui.platform.LocalContext
29 import androidx.compose.ui.semantics.contentDescription
30 import androidx.compose.ui.semantics.semantics
31 import androidx.compose.ui.text.font.FontWeight
32 import androidx.compose.ui.text.style.TextAlign
33 import androidx.compose.ui.unit.dp
34 import androidx.compose.ui.unit.sp
35 import androidx.core.net.toUri
36 import android.util.Base64
37 import java.io.File
38 import androidx.media3.common.MediaItem
39 import androidx.media3.exoplayer.ExoPlayer
40 import kotlinx.coroutines.CoroutineScope
41 import kotlinx.coroutines.flow.first
42 import kotlinx.coroutines.launch
43 import com.example.menuannam.EMAIL
44 import com.example.menuannam.TOKEN
45 import com.example.menuannam.dataStore
46 import com.example.menuannam.toMd5
47 import com.example.menuannam.saveAudioToInternalStorage
48 import com.example.menuannam.getCachedAudioFile
49 import com.example.menuannam.data.database.FlashCardDao
50 import com.example.menuannam.data.entity.FlashCard
51 import com.example.menuannam.data.network.AudioRequest
52 import com.example.menuannam.data.network.NetworkService
53
54 // StudyScreen supports two modes: SINGLE_CARD (view one
55 // card) and STUDY_SESSION (interactive 3-card learning)
56 enum class CardViewMode {
57     SINGLE_CARD, // View one specific card with delete
58     STUDY_SESSION // Interactive 3-card learning with flip
59     and audio
60 }
61 @Composable
62 fun StudyScreen(
63     changeMessage: (String) -> Unit = {}, // Updates status
64     bar with feedback
65     flashCardDao: FlashCardDao,
66     networkService: NetworkService,
67     mode: CardViewMode = CardViewMode.STUDY_SESSION,
68     cardId: Int = 0, // Card ID for SINGLE_CARD mode
69     onCardDeleted: () -> Unit = {}, // Navigate back after
70     deletion
71     coroutineScope: CoroutineScope? = null
72 ) {
73     val context = LocalContext.current
74     val appContext = context.applicationContext
75     val scope = coroutineScope ?: rememberCoroutineScope()
76
77     var flashCard by remember { mutableStateOf<FlashCard?>(
78         null) }
79     var lesson by remember { mutableStateOf<List<FlashCard>(
80         >>(emptyList()) ) }
81     var isLoading by remember { mutableStateOf(true) }
82     var errorMessage by remember { mutableStateOf<String?>(
83         null) }
84     var currentIndex by remember { mutableIntStateOf(0) }
85     var showVietnamese by remember { mutableStateOf(false) }
86
87     // Load data based on mode: SINGLE_CARD loads one card,
88     STUDY_SESSION loads 5 random cards
89     LaunchedEffect(Unit) {
90         scope.launch {
91             try {
92                 isLoading = true
93                 when (mode) {
94                     CardViewMode.SINGLE_CARD -> {
95                         cardId)
96                         if (flashCard == null) {
97                             errorMessage = "Flash card not
98                             found"
99                             changeMessage("Flash card not
100                             found")
101                         } else {
102                             changeMessage("Viewing flash
103                             card details")
104                         }
105                     }
106                     CardViewMode.STUDY_SESSION -> {
107                         val allCards = flashCardDao.getAll()
108                         lesson = if (allCards.size >= 3) {
109                             allCards.shuffled().take(3)
110                         } else {
111                             allCards
112                         }
113                         if (lesson.isNotEmpty()) {
114                             changeMessage("Card ${lesson.size}
115                             at index $currentIndex + 1")
116                         } else {
117                             errorMessage = "No flashcards
118                             found"
119                         }
120                     }
121                 }
122             } catch (e: Exception) {
123                 errorMessage = "Error loading data: ${e
124                 .message}"
125                 changeMessage("Error loading data")
126             } finally {
127                 isLoading = false
128             }
129         }
130
131         // Play audio utility
132         val playAudio: (String) -> Unit = { word ->
133             scope.launch {
134                 try {
135                     if (word.isBlank()) {
136                         changeMessage("Missing data to request
137                         audio")
138                     return@launch
139
140                     // Try cached audio first
141                     val cachedFile = getCachedAudioFile(
142                         appContext, word)
143                     if (cachedFile != null) {
144                         val mediaItem = MediaItem.fromUri(
145                             cachedFile.absolutePath.toUri())
146                         val player = ExoPlayer.Builder(
147                             appContext).build()
148                         player.addListener(object : androidx.
149                             media3.common.Player.Listener {
150                             override fun onPlaybackStateChanged(
151                                 playbackState: Int) {
152                                 when (playbackState) {
153                                     androidx.media3.common.
154                                     Player.STATE_BUFFERING -> changeMessage("Buffering...")
155                                     androidx.media3.common.
156                                     Player.STATE_READY -> changeMessage("Playing...")
157                                     androidx.media3.common.
158                                     Player.STATE_ENDED -> {
159                                         player.release()
160                                         changeMessage("Ready")
161                                     }
162                                 }
163                             }
164                         })
165                         player.setMediaItem(mediaItem)
166                         player.prepare()
167                         player.play()
168                         return@launch
169                     }
170
171                     // If no cache, call API to generate audio
172                     val prefs = appContext.dataStore.data.first
173                     ()
174                     val email = prefs[EMAIL] ?: ""
175                     val token = prefs[TOKEN] ?: ""
176                     if (email.isBlank() || token.isBlank()) { //
177                         Missing authentication data
178                         changeMessage("Missing data to request
179                         audio")
180                     }
181                 }
182             }
183         }
184     }

```

```

162         changeMessage("Generating audio...")
163         val resp = networkService.generateAudio( // 237
164             Send Vietnamese word + token to AWS Lambda 238
165             request = AudioRequest(word, email, 239
166             token) 240
167         )
168
169         if (resp.code == 200) { // Response code 200 241
170             = SUCCESS
171             val audioBytes = Base64.decode(resp. 242
172             message, Base64.DEFAULT) // Decode Base64-encoded MP3 243
173             val fileName = "${word.toMd5()}.mp3" 244
174             val file = saveAudioToInternalStorage( 245
175             appContext, audioBytes, fileName) // Save for future 246
176             use
177                 val mediaItem = MediaItem.fromUri(file. 247
178             applicationContext.toUri()) 248
179                 val player = ExoPlayer.Builder( 249
180             appContext).build() 250
181                 player.addListener(object : androidx. 251
182             media3.common.Player.Listener { // ExoPlayer state 252
183                 callbacks
184                     override fun onPlaybackStateChanged( 253
185             playbackState: Int) {
186                         when (playbackState) {
187                             androidx.media3.common. 254
188                             Player.STATE_BUFFERING -> changeMessage("Buffering...")
189                             androidx.media3.common. 255
190                             Player.STATE_READY -> changeMessage("Playing...")
191                             androidx.media3.common. 256
192                             Player.STATE_ENDED -> { 257
193                                 player.release()
194                                 changeMessage("Ready") 258
195                                 }
196                                 }
197                                 }
198
199         // UI Layout
200         Column(
201             modifier = Modifier
202                 .fillMaxWidth()
203                 .padding(16.dp),
204                 horizontalAlignment = Alignment.CenterHorizontally,
205                 verticalArrangement = Arrangement.spacedBy(16.dp)
206         ) {
207             if (isLoading) {
208                 Text("Loading...")
209             } else if (errorMessage != null) {
210                 Text(
211                     text = errorMessage!!,
212                     color = MaterialTheme.colorScheme.error
213                 )
214             } else when (mode) {
215                 CardViewMode.SINGLE_CARD -> {
216                     // Single Card View
217                     if (flashCard != null) {
218                         Card(
219                             modifier = Modifier.fillMaxWidth(),
220                             elevation = CardDefaults.
221                             cardElevation(defaultElevation = 4.dp)
222                         ) {
223                             Column(
224                                 modifier = Modifier.padding(16.
225                                     dp),
226                                 horizontalAlignment = Alignment.
227                                     CenterHorizontally,
228                                 verticalArrangement =
229                                     Arrangement.spacedBy(12.dp)
230                         ) {
231                             Text(
232                                 text = "Flash Card #${ 233
233             flashCard!!.uid}",
234             style = MaterialTheme. 235
235             typography.headlineSmall,
236             fontWeight = FontWeight.Bold
237             )
238             Spacer(modifier = Modifier.size 239
239             (8.dp))
240             Text(
241                 text = "English:",
242             style = MaterialTheme.
243             typography.labelLarge,
244             )
245             fontWeight = FontWeight.
246             Medium
247             )
248             Text(
249                 text = "Vietnamese:",
250             style = MaterialTheme.
251             typography.bodyLarge
252             )
253             Spacer(modifier = Modifier.size
253             (8.dp))
254             Text(
255                 text = "Vietnamese:",
256             style = MaterialTheme.
257             typography.labelLarge,
258             )
259             fontWeight = FontWeight.
260             Medium
261             )
262             Text(
263                 text = "Vietnamese:",
264             style = MaterialTheme.
265             typography.bodyLarge
266             )
267             Row(
268                 horizontalArrangement = Arrangement.
269                 Center,
270                 modifier = Modifier.fillMaxWidth()
271         ) {
272             Button(
273                 onClick = {
274                     scope.launch {
275                         try {
276                             flashCardDao.delete(
277                                 flashCard!!)
278                             changeMessage("Card
279                             deleted successfully!")
280                             onCardDeleted()
281                         } catch (e: Exception) {
282                             changeMessage("Error
283                             : ${e.message}")
284                         }
285                     }
286                 }
287             )
288             Text("Delete")
289             Spacer(modifier = Modifier.size(8.dp
290         ))
291             Button(
292                 onClick = {
293                     playAudio(flashCard!!)
294             }
295             )
296             vietnameseCard ?: ""
297             )
298             Text("Play Audio")
299             )
300             )
301             )
302             )
303             )
304             )
305             )
306             )
307             )
308             )
309             )
310             )
311             )
312             )
313             )
314             )
315             )
316             )
317             )
318             )
319             )
320             )
321             )
322             )
323             )
324             )
325             )
326             )
327             )
328             )
329             )
330             )
331             )
332             )
333             )
334             )
335             )
336             )
337             )
338             )
339             )
340             )
341             )
342             )
343             )
344             )
345             )
346             )
347             )
348             )
349             )
350             )
351             )
352             )
353             )
354             )
355             )
356             )
357             )
358             )
359             )
360             )
361             )
362             )
363             )
364             )
365             )
366             )
367             )
368             )
369             )
370             )
371             )
372             )
373             )
374             )
375             )
376             )
377             )
378             )
379             )
380             )
381             )
382             )
383             )
384             )
385             )
386             )
387             )
388             )
389             )
390             )
391             )
392             )
393             )
394             )
395             )
396             )
397             )
398             )
399             )
400             )
401             )
402             )
403             )
404             )
405             )
406             )
407             )
408             )
409             )
410             )
411             )
412             )
413             )
414             )
415             )
416             )
417             )
418             )
419             )
420             )
421             )
422             )
423             )
424             )
425             )
426             )
427             )
428             )
429             )
430             )
431             )
432             )
433             )
434             )
435             )
436             )
437             )
438             )
439             )
440             )
441             )
442             )
443             )
444             )
445             )
446             )
447             )
448             )
449             )
450             )
451             )
452             )
453             )
454             )
455             )
456             )
457             )
458             )
459             )
460             )
461             )
462             )
463             )
464             )
465             )
466             )
467             )
468             )
469             )
470             )
471             )
472             )
473             )
474             )
475             )
476             )
477             )
478             )
479             )
480             )
481             )
482             )
483             )
484             )
485             )
486             )
487             )
488             )
489             )
490             )
491             )
492             )
493             )
494             )
495             )
496             )
497             )
498             )
499             )
500             )
501             )
502             )
503             )
504             )
505             )
506             )
507             )
508             )
509             )
510             )
511             )
512             )
513             )
514             )
515             )
516             )
517             )
518             )
519             )
520             )
521             )
522             )
523             )
524             )
525             )
526             )
527             )
528             )
529             )
530             )
531             )
532             )
533             )
534             )
535             )
536             )
537             )
538             )
539             )
540             )
541             )
542             )
543             )
544             )
545             )
546             )
547             )
548             )
549             )
550             )
551             )
552             )
553             )
554             )
555             )
556             )
557             )
558             )
559             )
560             )
561             )
562             )
563             )
564             )
565             )
566             )
567             )
568             )
569             )
570             )
571             )
572             )
573             )
574             )
575             )
576             )
577             )
578             )
579             )
580             )
581             )
582             )
583             )
584             )
585             )
586             )
587             )
588             )
589             )
590             )
591             )
592             )
593             )
594             )
595             )
596             )
597             )
598             )
599             )
599             )
600             )
601             )
602             )
603             )
604             )
605             )
606             )
607             )
608             )
609             )
609             )
610             )
611             )
612             )
613             )
614             )
615             )
616             )
617             )
618             )
619             )
619             )
620             )
621             )
622             )
623             )
624             )
625             )
626             )
627             )
628             )
629             )
629             )
630             )
631             )
632             )
633             )
634             )
635             )
636             )
637             )
638             )
639             )
639             )
640             )
641             )
642             )
643             )
644             )
645             )
646             )
647             )
648             )
649             )
649             )
650             )
651             )
652             )
653             )
654             )
655             )
656             )
657             )
658             )
659             )
659             )
660             )
661             )
662             )
663             )
664             )
665             )
666             )
667             )
668             )
669             )
669             )
670             )
671             )
672             )
673             )
674             )
675             )
676             )
677             )
678             )
679             )
679             )
680             )
681             )
682             )
683             )
684             )
685             )
686             )
687             )
688             )
689             )
689             )
690             )
691             )
692             )
693             )
694             )
695             )
696             )
697             )
698             )
699             )
699             )
700             )
701             )
702             )
703             )
704             )
705             )
706             )
707             )
708             )
709             )
709             )
710             )
711             )
712             )
713             )
714             )
715             )
716             )
717             )
718             )
719             )
719             )
720             )
721             )
722             )
723             )
724             )
725             )
726             )
727             )
728             )
729             )
729             )
730             )
731             )
732             )
733             )
734             )
735             )
736             )
737             )
738             )
739             )
739             )
740             )
741             )
742             )
743             )
744             )
745             )
746             )
747             )
748             )
749             )
749             )
750             )
751             )
752             )
753             )
754             )
755             )
756             )
757             )
758             )
759             )
759             )
760             )
761             )
762             )
763             )
764             )
765             )
766             )
767             )
768             )
769             )
769             )
770             )
771             )
772             )
773             )
774             )
775             )
776             )
777             )
778             )
779             )
779             )
780             )
781             )
782             )
783             )
784             )
785             )
786             )
787             )
788             )
789             )
789             )
790             )
791             )
792             )
793             )
794             )
795             )
796             )
797             )
798             )
799             )
799             )
800             )
801             )
802             )
803             )
804             )
805             )
806             )
807             )
808             )
809             )
809             )
810             )
811             )
812             )
813             )
814             )
815             )
816             )
817             )
818             )
819             )
819             )
820             )
821             )
822             )
823             )
824             )
825             )
826             )
827             )
828             )
829             )
829             )
830             )
831             )
832             )
833             )
834             )
835             )
836             )
837             )
838             )
839             )
839             )
840             )
841             )
842             )
843             )
844             )
845             )
846             )
847             )
848             )
849             )
849             )
850             )
851             )
852             )
853             )
854             )
855             )
856             )
857             )
858             )
859             )
859             )
860             )
861             )
862             )
863             )
864             )
865             )
866             )
867             )
868             )
869             )
869             )
870             )
871             )
872             )
873             )
874             )
875             )
876             )
877             )
878             )
879             )
879             )
880             )
881             )
882             )
883             )
884             )
885             )
886             )
887             )
888             )
889             )
889             )
890             )
891             )
892             )
893             )
894             )
895             )
896             )
897             )
898             )
899             )
899             )
900             )
901             )
902             )
903             )
904             )
905             )
906             )
907             )
908             )
909             )
909             )
910             )
911             )
912             )
913             )
914             )
915             )
916             )
917             )
918             )
919             )
919             )
920             )
921             )
922             )
923             )
924             )
925             )
926             )
927             )
928             )
929             )
929             )
930             )
931             )
932             )
933             )
934             )
935             )
936             )
937             )
938             )
939             )
939             )
940             )
941             )
942             )
943             )
944             )
945             )
946             )
947             )
948             )
949             )
949             )
950             )
951             )
952             )
953             )
954             )
955             )
956             )
957             )
958             )
959             )
959             )
960             )
961             )
962             )
963             )
964             )
965             )
966             )
967             )
968             )
969             )
969             )
970             )
971             )
972             )
973             )
974             )
975             )
976             )
977             )
978             )
979             )
979             )
980             )
981             )
982             )
983             )
984             )
985             )
986             )
987             )
988             )
989             )
989             )
990             )
991             )
992             )
993             )
994             )
995             )
996             )
997             )
998             )
999             )
999             )
1000            )
1001            )
1002            )
1003            )
1004            )
1005            )
1006            )
1007            )
1008            )
1009            )
1009            )
1010            )
1011            )
1012            )
1013            )
1014            )
1015            )
1016            )
1017            )
1018            )
1019            )
1019            )
1020            )
1021            )
1022            )
1023            )
1024            )
1025            )
1026            )
1027            )
1028            )
1029            )
1029            )
1030            )
1031            )
1032            )
1033            )
1034            )
1035            )
1036            )
1037            )
1038            )
1039            )
1039            )
1040            )
1041            )
1042            )
1043            )
1044            )
1045            )
1046            )
1047            )
1048            )
1049            )
1049            )
1050            )
1051            )
1052            )
1053            )
1054            )
1055            )
1056            )
1057            )
1058            )
1059            )
1059            )
1060            )
1061            )
1062            )
1063            )
1064            )
1065            )
1066            )
1067            )
1068            )
1069            )
1069            )
1070            )
1071            )
1072            )
1073            )
1074            )
1075            )
1076            )
1077            )
1078            )
1079            )
1079            )
1080            )
1081            )
1082            )
1083            )
1084            )
1085            )
1086            )
1087            )
1088            )
1089            )
1089            )
1090            )
1091            )
1092            )
1093            )
1094            )
1095            )
1096            )
1097            )
1098            )
1099            )
1099            )
1100            )
1101            )
1102            )
1103            )
1104            )
1105            )
1106            )
1107            )
1108            )
1109            )
1109            )
1110            )
1111            )
1112            )
1113            )
1114            )
1115            )
1116            )
1117            )
1118            )
1119            )
1119            )
1120            )
1121            )
1122            )
1123            )
1124            )
1125            )
1126            )
1127            )
1128            )
1129            )
1129            )
1130            )
1131            )
1132            )
1133            )
1134            )
1135            )
1136            )
1137            )
1138            )
1139            )
1139            )
1140            )
1141            )
1142            )
1143            )
1144            )
1145            )
1146            )
1147            )
1148            )
1149            )
1149            )
1150            )
1151            )
1152            )
1153            )
1154            )
1155            )
1156            )
1157            )
1158            )
1159            )
1159            )
1160            )
1161            )
1162            )
1163            )
1164            )
1165            )
1166            )
1167            )
1168            )
1169            )
1169            )
1170            )
1171            )
1172            )
1173            )
1174            )
1175            )
1176            )
1177            )
1178            )
1179            )
1180            )
1181            )
1182            )
1183            )
1184            )
1185            )
1186            )
1187            )
1188            )
1189            )
1189            )
1190            )
1191            )
1192            )
1193            )
1194            )
1195            )
1196            )
1197            )
1198            )
1199            )
1199            )
1200            )
1201            )
1202            )
1203            )
1204            )
1205            )
1206            )
1207            )
1208            )
1209            )
1209            )
1210            )
1211            )
1212            )
1213            )
1214            )
1215            )
1216            )
1217            )
1218            )
1219            )
1219            )
1220            )
1221            )
1222            )
1223            )
1224            )
1225            )
1226            )
1227            )
1228            )
1229            )
1230            )
1231            )
1232            )
1233            )
1234            )
1235            )
1236            )
1236            )
1237            )
1238            )
1239            )
1239            )
1240            )
1241            )
1242            )
1243            )
1244            )
1245            )
1246            )
1247            )
1248            )
1249            )
1249            )
1250            )
1251            )
1252            )
1253            )
1254            )
1255            )
1256            )
1257            )
1258            )
1259            )
1259            )
1260            )
1261            )
1262            )
1263            )
1264            )
1265            )
1266            )
1267            )
1268            )
1269            )
1269            )
1270            )
1271            )
1272            )
1273            )
1274            )
1275            )
1276            )
1277            )
1278            )
1279            )
1279            )
1280            )
1281            )
1282            )
1283            )
1284            )
1285            )
1286            )
1287            )
1288            )
1289            )
1289            )
1290            )
1291            )
1292            )
1293            )
1294            )
1295            )
1296            )
1297            )
1298            )
1299            )
1299            )
1300            )
1301            )
1302            )
1303            )
1304            )
1305            )
1306            )
1307            )
1308            )
1309            )
1309            )
1310            )
1311            )
1312            )
1313            )
1314            )
1315            )
1316            )
1317            )
1318            )
1319            )
1319            )
1320            )
1321            )
1322            )
1323            )
1324            )
1325            )
1326            )
1327            )
1328            )
1329            )
1329            )
1330            )
1331            )
1332            )
1333            )
1334            )
1335            )
1336            )
1337            )
1338            )
1339            )
1339            )
1340            )
1341            )
1342            )
1343            )
1344            )
1345            )
1346            )
1347            )
1348            )
1349            )
1349            )
1350            )
1351            )
1352            )
1353            )
1354            )
1355            )
1356            )
1357            )
1358            )
1359            )
1359            )
1360            )
1361            )
1362            )
1363            )
1364            )
1365            )
1366            )
1367            )
1368            )
1369            )
1369            )
1370            )
1371            )
1372            )
1373            )
1374            )
1375            )
1376            )
1377            )
1378            )
1379            )
1379            )
1380            )
1381            )
1382            )
1383            )
1384            )
1385            )
1386            )
1387            )
1388            )
1389            )
1389            )
1390            )
1391            )
1392            )
1393            )
1394            )
1395            )
1396            )
1397            )
1398            )
1399            )
1399            )
1400            )
1401            )
1402            )
1403            )
1404            )
1405            )
1406            )
1407            )
1408            )
1409            )
1409            )
1410            )
1411            )
1412            )
1413            )
1414            )
1415            )
1416            )
1417            )
1418            )
1419            )
1419            )
1420            )
1421            )
1422            )
1423            )
1424            )
1425            )
1426            )
1427            )
1428            )
1429            )
1429            )
1430            )
1431            )
1432            )
1433            )
1434            )
1435            )
1436            )
1437            )
1438            )
1439            )
1439            )
1440            )
1441            )
1442            )
1443            )
1444            )
1445            )
1446            )
1447            )
1448            )
1449            )
1449            )
1450            )
1451            )
1452            )
1453            )
1454            )
1455            )
1456            )
1457            )
1458            )
1459            )
1459            )
1460            )
1461            )
1462            )
1463            )
1464            )
1465            )
1466            )
1467            )
1468            )
1469            )
1469            )
1470            )
1471            )
1472            )
1473            )
1474            )
1475            )
1476            )
1477            )
1478            )
1479            )
1479            )
1480            )
1481            )
1482            )
1483            )
1484            )
1485            )
1486            )
1487            )
1488            )
1489            )
1489            )
1490            )
1491            )
1492            )
1493            )
1494            )
1495            )
1496            )
1497            )
1498            )
1499            )
1499            )
1500            )
1501            )
1502            )
1503            )
1504            )
1505            )
1506            )
1507            )
1508            )
1509            )
1509            )
1510            )
1511            )
1512            )
1513            )
1514            )
1515            )
1516            )
1517            )
1518            )
1519            )
1519            )
1520            )
1521            )
1522            )
1523            )
1524            )
1525            )
1526            )
1527            )
1528            )
1529            )
1529            )
1530            )
1531            )
1532            )
1533            )
1534            )
1535            )
1536            )
1537            )
1538            )
1539            )
1539            )
1540            )
1541            )
1542            )
1543            )
1544            )
1545            )
1546            )
1547            )
1548            )
1549            )
1549            )
1550            )
1551            )
1552            )
1553            )
1554            )
1555            )
1556            )
1557            )
1558            )
1559            )
1559            )
1560            )
1561            )
1562            )
1563            )
1564            )
1565            )
1566            )
1567            )
1568            )
1569            )
1569            )
1570            )
1571            )
1572            )
1573            )
1574            )
1575            )
1576            )
1577            )
1578            )
1579            )
1579            )
1580            )
1581            )
1582            )
1583            )
1584            )
1585            )
1586            )
1587            )
1588            )
1589            )
1589            )
1590            )
1591            )
1592            )
1593            )
1594            )
1595            )
1596            )
1597            )
1598            )
1599            )
1599            )
1600            )
1601            )
1602            )
1603            )
1604            )
1605            )
1606            )
1607            )
1608            )
1609            )
1609            )
1610            )
1611            )
1612            )
1613            )
1614            )
1615            )
1616            )
1617            )
1618            )
1619            )
1619            )
1620            )
1621            )
1622            )
1623            )
1624            )
1625            )
1626            )
1627            )
1628            )
1629            )
1629            )
1630            )
1631            )
1632            )
1633            )
1634            )
1635            )
1636            )
1637            )
1638            )
1639            )
1639            )
1640            )
1641            )
1642            )
1643            )
1644            )
1645            )
1646            )
1647            )
1648            )
1649            )
1649            )
1650            )
1651            )
1652            )
1653            )
1654            )
1655            )
1656            )
1657            )
1658            )
1659            )
1659            )
1660            )
1661            )
1662            )
1663            )
1664            )
1665            )
1666            )
1667            )
1668            )
1669            )
1669            )
1670            )
1671            )
1672            )
1673            )
1674            )
1675            )
1676            )
1677            )
1678            )
1679            )
1679            )
1680            )
1681            )
1682            )
1683            )
1684            )
1685            )
1686            )
1687            )
1688            )
1689            )
1689            )
1690            )
1691            )
1692            )
1693            )
1694            )
1695            )
1696            )
1697            )
1698            )
1699            )
1699            )
1700            )
1701            )
1702            )
1703            )
1704            )
1705            )
1706            )
1707            )
1708            )
1709            )
1709            )
1710            )
1711            )
1712            )
1713            )
1714            )
1715            )
1716            )
1717            )
1718            )
1719            )
1719            )
1720            )
1721            )
1722            )
1723            )
1724            )
1725            )
1726            )
1727            )
1728            )
1729            )
1729            )
1730            )
1731            )
1732            )
1733            )
1734            )
1735            )
1736            )
1737            )
1738            )
1739            )
1739            )
1740            )
1741            )
1742            )
1743            )
1744            )
1745            )
1746            )
1747            )
1748            )
1749            )
1749            )
1750            )
1751            )
1752            )
1753            )
1754            )
1755            )
1756            )
1757            )
1758            )
1759            )
1759            )
1760            )
1761            )
1762            )
1763            )
1764            )
1765            )
1766            )
1767            )
1768            )
1769            )
1769            )
1770            )
1771            )
1772            )
1773            )
1774            )
1775            )
1776            )
1777            )
1778            )
1779            )
1779            )
1780            )
1781            )
1782            )
1783            )
1784            )
1785            )
1786            )
1787            )
1788            )
1789            )
1789            )
1790            )
1791            )
1792            )
1793            )
1794            )
1795            )
1796            )
1797            )
1798            )
1799            )
1799            )
1800            )
1801            )
1802            )
1803            )
1804            )
1805            )
1806            )
1807            )
1808            )
1809            )
1809            )
1810            )
1811            )
1812            )
1813            )
1814            )
1815            )
1816            )
1817            )
1818            )
1819            )
1819            )
1820            )
1821            )
1822            )
1823            )
1824            )
1825            )
1826            )
1827            )
1828            )
1829            )
1829            )
1830            )
1831            )
1832            )
1833            )
1834            )
1835            )
1836            )
1837            )
1838            )
1839            )
1839            )
1840            )
1841            )
1842            )
1843            )
1844            )
1845            )
1846            )
1847            )
1848            )
1849            )
1849            )
1850            )
1851            )
1852            )
1853            )
1854            )
1855            )
1856            )
1857            )
1858            )
1859            )
1859            )
1860            )
1861            )
1862            )
1863            )
1864            )
1865            )
1866            )
1867            )
1868            )
1869            )
1869            )
1870            )
1871            )
1872            )
1873            )
1874            )
1875            )
1876            )
1877            )
1878            )
1879            )
1879            )
1880            )
1881            )
1882            )
1883            )
1884            )
1885            )
1886            )
1887            )
1888            )
1889            )
1889            )
1890            )
1891            )
1892            )
1893            )
1894            )
1895            )
1896            )
1897            )
1898            )
1899            )
1899            )
1900            )
1901            )
1902            )
1903            )
1904            )
1905            )
1906            )
1907            )
1908            )
1909            )
1909            )
1910            )
1911            )
1912            )
1913            )
1914            )
1915            )
1916            )
1917            )
1918            )
1919            )
1919            )
1920            )
1921            )
1922            )
1923            )
1924            )
19
```

```
315     vietnameseCard ?: ""          337
316         } else {                338
317             currentCard.englishCard 339
318             ?: ""                 340
319         },
320         modifier = Modifier.padding 341
321     (32.dp)                      342
322         )
323     }
324
325     if (showVietnamese) {
326         Button(
327             modifier = Modifier
328                 .fillMaxWidth()
329                 .semantics {
contentDescription = "PlayAudio",
330             onClick = {
331                 playAudio(currentCard.
vietnameseCard ?: "")          352
332             }
333         ) { Text("Play Audio")      353
334     }
335 }
336
337     .,
338     fontSize = 32.sp,
339     textAlign = TextAlign.Center
340
341     ),
342     modifier = Modifier.padding
343     344
345     )
346
347     348
348     if (showVietnamese) {
349         Button(
350             modifier = Modifier
351                 .semantics {
352                 contentDescription = "PlayAudio"
353             onClick = {
354                 playAudio(currentCard.
355             )
356         }
357     )
358 }
```

# Chapter 5

## UI Theme

### 5.1 Color.kt

```
1 package com.example.menuannam.ui.theme
2
3 import androidx.compose.ui.graphics.Color
4
5 val Purple80 = Color(0xFFD0BCFF)
6 val PurpleGrey80 = Color(0xFFCCC2DC)
7 val Pink80 = Color(0xFFEB8C8)
8
9 val Purple40 = Color(0xFF6650a4)
10 val PurpleGrey40 = Color(0xFF625b71)
11 val Pink40 = Color(0xFF7D5260)
12
13 ./app/src/main/java/com/example/menuannam/ui/theme/Color.kt
```

### 5.2 Theme.kt

```
1 package com.example.menuannam.ui.theme
2
3 import android.app.Activity
4 import android.os.Build
5 import androidx.compose.foundation.isSystemInDarkTheme
6 import androidx.compose.material3.MaterialTheme
7 import androidx.compose.material3.darkColorScheme
8 import androidx.compose.material3.dynamicDarkColorScheme
9 import androidx.compose.material3.dynamicLightColorScheme
10 import androidx.compose.material3.lightColorScheme
11 import androidx.compose.runtime.Composable
12 import androidx.compose.ui.platform.LocalContext
13
14 private val DarkColorScheme = darkColorScheme(
15     primary = Purple80,
16     secondary = PurpleGrey80,
17     tertiary = Pink80
18 )
19
20 private val LightColorScheme = lightColorScheme(
21     primary = Purple40,
22     secondary = PurpleGrey40,
23     tertiary = Pink40
24
25     /* Other default colors to override
26     background = Color(0xFFFFFBFE),
27     surface = Color(0xFFFFFBFE),
28     onPrimary = Color.White,
29     onSecondary = Color.White,
30     onTertiary = Color.White,
31     onBackground = Color(0xFF1C1B1F),
32     onSurface = Color(0xFF1C1B1F),
33     */
34 )
35
36 @Composable
37 fun MenuAnNamTheme(
38     darkTheme: Boolean = isSystemInDarkTheme(),
39     // Dynamic color is available on Android 12+
40     dynamicColor: Boolean = true,
41     content: @Composable () -> Unit
42 ) {
43     val colorScheme = when {
44         dynamicColor && Build.VERSION.SDK_INT >= Build.
45             VERSION_CODES.S -> {
46                 val context = LocalContext.current
47                 if (darkTheme) dynamicDarkColorScheme(context)
48                 else dynamicLightColorScheme(context)
49             }
50
51             darkTheme -> DarkColorScheme
52             else -> LightColorScheme
53     }
54     MaterialTheme(
55         colorScheme = colorScheme,
56         typography = Typography,
57         content = content
58     )
59
60 ./app/src/main/java/com/example/menuannam/ui/theme/Theme.kt
```

### 5.3 Type.kt

```
1 package com.example.menuannam.ui.theme
2
3 import androidx.compose.material3.Typography
4 import androidx.compose.ui.text.TextStyle
5 import androidx.compose.ui.text.font.FontFamily
6 import androidx.compose.ui.text.font.FontWeight
7 import androidx.compose.ui.unit.sp
8
9 // Set of Material typography styles to start with
10 val Typography = Typography(
11     bodyLarge = TextStyle(
12         fontFamily = FontFamily.Default,
13         fontWeight = FontWeight.Normal,
14         fontSize = 16.sp,
15         lineHeight = 24.sp,
16         letterSpacing = 0.5.sp
17     )
18     /* Other default text styles to override
19     titleLarge = TextStyle(
20         fontFamily = FontFamily.Default,
21         fontWeight = FontWeight.Normal,
22         fontSize = 22.sp,
23         lineHeight = 28.sp,
24         letterSpacing = 0.sp
25     ),
26     labelSmall = TextStyle(
27         fontFamily = FontFamily.Default,
28         fontWeight = FontWeight.Medium,
29         fontSize = 11.sp,
30         lineHeight = 16.sp,
31         letterSpacing = 0.5.sp
32     )
33     */
34 )
35
36 ./app/src/main/java/com/example/menuannam/ui/theme/Type.kt
```

# Chapter 6

## Build Configuration

### 6.1 build.gradle.kts (App Module)

```
1 plugins {
2     alias(libs.plugins.android.application)
3     alias(libs.plugins.kotlin.android)
4     alias(libs.plugins.kotlin.compose)
5     id("org.jetbrains.kotlin.plugin.serialization")
6     id("com.google.devtools.ksp")
7 }
8
9 android {
10    namespace = "com.example.menuannam"
11    compileSdk = 36
12
13    defaultConfig {
14        applicationId = "com.example.menuannam"
15        minSdk = 26 // originally 24
16        targetSdk = 36
17        versionCode = 27
18        versionName = "1.0"
19        //testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
20    }
21
22    // For Kotlin projects using KSP:
23    ksp {
24        arg("room.schemaLocation", "$projectDir/schemas")
25    }
26
27    buildTypes {
28        release {
29            isMinifyEnabled = false
30            isDebuggable = false
31            proguardFiles(
32                getDefaultProguardFile("proguard-android-optimize.txt"),
33                "proguard-rules.pro"
34            )
35        }
36    }
37
38    compileOptions {
39        sourceCompatibility = JavaVersion.VERSION_21
40        targetCompatibility = JavaVersion.VERSION_21
41    }
42
43    kotlin {
44        compilerOptions {
45            jvmTarget.set(org.jetbrains.kotlin.gradle.dsl.JvmTarget.JVM_21)
46        }
47    }
48
49    buildFeatures {
50        compose = true
51    }
52
53    androidResources {
54        generateLocaleConfig = true
55    }
56
57    testOptions {
58        unitTests {
59            isIncludeAndroidResources = true
60        }
61    }
62 }
63
64 dependencies {
65     implementation(libs.androidx.core.ktx)
66     implementation(libs.androidx.lifecycle.runtime.ktx)
67     implementation(libs.androidx.activity.compose)
68     implementation(platform(libs.androidx.compose.bom))
69     implementation(libs.androidx.material3)
70     implementation(libs.androidx.navigation.runtime.ktx)
71     implementation(libs.androidx.navigation.compose)
72     implementation(libs.androidx.navigation.testing)
```

```

73     implementation(libs.core.ktx)
74     implementation(libs.androidx.compose.ui.test.junit4)
75
76     implementation(libs.androidx.compose.ui)
77     implementation(libs.androidx.compose.foundation)
78     implementation(libs.androidx.compose.foundation.layout)
79     implementation(libs.androidx.media3.exoplayer)
80     //implementation(libs.androidx.compose.ui.test)
81     testImplementation(libs.junit)
82     // For local unit tests
83     //testImplementation(libs.androidx.core.testing)
84     testImplementation(libs.robolectric)
85     // Needed for createComposeRule(), but not for createAndroidComposeRule<YourActivity>():
86     debugImplementation(libs.androidx.compose.ui.test.manifest)
87
88     //androidTestImplementation(libs.androidx.junit)
89     //androidTestImplementation(libs.androidx.espresso.core)
90     androidTestImplementation(platform(libs.androidx.compose.bom))
91     // Test rules and transitive dependencies:
92     androidTestImplementation(libs.androidx.compose.ui.test.junit4)
93
94     /* Dependencies related to room */
95     // room-compiler: is for the code generation that happens
96     // during the build process to create the necessary database
97     // infrastructure based on your annotations.
98     // implementation(libs.androidx.room.compiler)
99     // room-runtime: is for the code that runs on
100    // your device to interact with the database.
101    implementation(libs.androidx.room.runtime)
102    // If this project uses any Kotlin source, use Kotlin Symbol Processing (KSP)
103    // See Add the KSP plugin to your project
104    ksp(libs.androidx.room.compiler)
105    // optional - Kotlin Extensions and Coroutines support for Room
106    implementation(libs.androidx.room.ktx)
107    // optional - Test helpers
108    testImplementation(libs.androidx.room.testing)
109
110   // serialization
111   implementation(libs.kotlinx.serialization.json)
112   implementation(libs.androidx.navigation.compose.v280alpha08)
113
114   // retrofit
115   implementation("com.squareup.retrofit2:retrofit:2.9.0")
116   implementation("com.squareup.retrofit2:converter-gson:2.9.0")
117
118   // datastore
119   implementation("androidx.datastore:datastore-preferences:1.2.0")
120   implementation("androidx.datastore:datastore-preferences-core:1.2.0")
121 }

```

./app/build.gradle.kts

## 6.2 libs.versions.toml

```

1 [versions]
2
3 agp = "8.12.0"
4
5 composeVersion = "1.9.4"
6
7 kotlin = "2.1.10"
8
9 coreKtx = "1.17.0"
10
11 junit = "4.13.2"
12
13 #junitVersion = "1.3.0"
14
15 #espressoCore = "3.7.0"
16
17 lifecycleRuntimeKtx = "2.9.4"
18
19 activityCompose = "1.11.0"
20
21 composeBom = "2025.10.01"
22
23 navigationCompose = "2.9.5"
24
25 material3 = "1.4.0"
26
27 navigationRuntimeKtx = "2.9.5"
28
29 navigationTesting = "2.9.5"
30
31 robolectric = "4.16"
32
33 roomVersion = "2.8.3"
34
35 coreKtxVersion = "1.7.0"
36
37 ui = "1.9.4"
38
39 composeFoundation = "1.9.4"
40
41 media3Version = "1.4.1"
42
43 kotlinxSerializationVersion = "1.7.3"

```

```

44
45
46
47
48 [libraries]
49
50 androidx-compose-ui-test-junit4 = { module = "androidx.compose.ui:ui-test-junit4", version.ref = "composeVersion" }
51
52 androidx-compose-ui-test-manifest = { module = "androidx.compose.ui:ui-test-manifest", version.ref = "composeVersion" }
53
54 androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
55
56 androidx-room-compiler = { module = "androidx.room:room-compiler", version.ref = "roomVersion" }
57
58 androidx-room-ktx = { module = "androidx.room:room-ktx", version.ref = "roomVersion" }
59
60 androidx-room-runtime = { module = "androidx.room:room-runtime", version.ref = "roomVersion" }
61
62 androidx-room-testing = { module = "androidx.room:room-testing", version.ref = "roomVersion" }
63
64 junit = { group = "junit", name = "junit", version.ref = "junit" }
65
66 #androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref = "junitVersion" }
67
68 #androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-core", version.ref = "espressoCore" }
69
70 androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-ktx", version.ref = "
    lifecycleRuntimeKtx" }
71
72 androidx-activity-compose = { group = "androidx.activity", name = "activity-compose", version.ref = "activityCompose" }
73
74 androidx-compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref = "composeBom" }
75
76 androidx-navigation-compose = { group = "androidx.navigation", name = "navigation-compose", version.ref = "navigationCompose"
    }
77
78 androidx-material3 = { group = "androidx.compose.material3", name = "material3", version.ref = "material3" }
79
80 androidx-navigation-runtime-ktx = { group = "androidx.navigation", name = "navigation-runtime-ktx", version.ref = "
    navigationRuntimeKtx" }
81
82 androidx-navigation-testing = { group = "androidx.navigation", name = "navigation-testing", version.ref = "navigationTesting"
    }
83
84 robolectric = { module = "org.robolectric:robolectric", version.ref = "robolectric" }
85
86 core-ktx = { group = "androidx.test", name = "core-ktx", version.ref = "coreKtxVersion" }
87
88 androidx-compose-ui = { group = "androidx.compose.ui", name = "ui", version.ref = "ui" }
89
90 androidx-compose-foundation = { group = "androidx.compose.foundation", name = "foundation", version.ref = "composeFoundation"
    }
91
92 androidx-compose-foundation-layout = { group = "androidx.compose.foundation", name = "foundation-layout", version.ref = "
    composeFoundation" }
93
94 androidx-media3-exoplayer = { group = "androidx.media3", name = "media3-exoplayer", version.ref = "media3Version" }
95
96 kotlinx-serialization-json = { group = "org.jetbrains.kotlinx", name = "kotlinx-serialization-json", version.ref = "
    kotlinxSerializationVersion" }
97
98 androidx-navigation-compose-v280alpha08 = { group = "androidx.navigation", name = "navigation-compose", version = "2.8.0-
    alpha08" }
99
100
101
102 [plugins]
103
104 android-application = { id = "com.android.application", version.ref = "agp" }
105
106 kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
107
108 kotlin-compose = { id = "org.jetbrains.kotlin.plugin.compose", version.ref = "kotlin" }

```

./gradle/libs.versions.toml

# Chapter 7

# Mobile Application Use Cases & Implementation

## 7.1 Use Cases Implementation Status

### 7.1.1 1. Request an Authentication Token

- ✓ **Status:** Implemented
- **Requirement:** The user can request a token (for authentication) by providing his/her email address. The token will be sent to the user's provided email address.
- **Implementation:**
  - **Screen:** LoginScreen.kt
  - **Lambda API:** Token Generation Endpoint
  - **Flow:** User enters email → API call to Lambda → Token sent to email → User receives token
- **Code Location:** LoginScreen.kt; NetworkService.kt; DataTypes.kt

### 7.1.2 2. Save the Pair Email Address/Authentication Token in the App

- ✓ **Status:** Implemented
- **Requirement:** The user can save the authentication token and the corresponding email address in the app's preferences datastore.
- **Implementation:**
  - **Screen:** TokenScreen.kt
  - **Storage:** DataStore Preferences (EMAIL and TOKEN keys)
  - **Flow:** User enters token from email → Click save → Both email and token persisted in DataStore
- **Code Location:** TokenScreen.kt; MainActivity.kt (DataStore Setup)

### 7.1.3 3. Add a Flashcard

- ✓ **Status:** Implemented
- **Requirement:** The user can add a flashcard to the app's (Room) database by providing the corresponding English and Vietnamese words. The (Room) database should not store duplicated flashcards. If the user requests to add a flashcard that already exists in the (Room) database, the app will refuse to do so and will inform the user accordingly. After the user successfully adds a flashcard to the (Room) database, the user should be allowed to continue adding more flashcards until he/she decides not to do so (e.g., when the user clicks on a "back" button).
- **Implementation:**
  - **Screen:** AddScreen.kt
  - **Database:** FlashCard entity with unique index on (english\_card, vietnamese\_card)
  - **Strategy:** OnConflictStrategy.IGNORE prevents duplicates
  - **User Feedback:** App informs user if duplicate exists
  - **UI:** Clear button resets fields, Back button available for navigation
  - **Continue Adding:** User can add multiple flashcards without leaving the screen
- **Code Location:** AddScreen.kt; FlashCard.kt; FlashCardDao.kt (Insert Methods)

## 7.1.4 4. Search Flashcards

✓ **Status:** Implemented

- **Requirement:** The user can search the flashcards stored in the app's (Room) database using "filters". For each search, the user can specify one of the following filters:

- a) The flashcard's English word should match exactly the English word provided by the user and the flashcard's Vietnamese word should match exactly the Vietnamese word provided by the user.
- b) The flashcard's English word should match exactly the English word provided by the user and the flashcard's Vietnamese word should contain the Vietnamese word provided by the user.
- c) The flashcard's Vietnamese word should match exactly the Vietnamese word provided by the user and the flashcard's English word should contain the English word provided by the user.
- d) The flashcard's English word and the flashcard's Vietnamese word should contain, respectively, the English word and the Vietnamese word provided by the user.

When requested to do so (e.g., when the user clicks on a "search" button), the app will show in a table the flashcards that satisfy the "filter" specified by the user. Each row in the table will contain an "edit" button that allows the user to Edit the corresponding flashcard.

- **Implementation:**

- **Screens:** FilterScreen.kt (search form), SearchScreen.kt (results table)
  - **Database Query:** CASE WHEN logic in FlashCardDao.getFilteredFlashCards()
  - **UI:** Checkboxes for exact/partial match per field
  - **Results Display:** LazyColumn table with Edit button per row
- **Code Location:** FilterScreen.kt; SearchScreen.kt; FlashCardDao.kt (Filter Query)

## 7.1.5 5. Edit a Flashcard

✓ **Status:** Implemented

- **Requirement:** The user can edit a flashcard. When editing a flashcard, the app will initially show the flashcard's English and Vietnamese words as they are stored in the app's (Room) database, and the name of the file in the app's internal storage containing the audio corresponding to the pronunciation of the flashcard's Vietnamese word (if exists). Afterwards:

- The user can update the flashcard's English and Vietnamese words in the (Room) database by providing the new English and Vietnamese words.
- The user can delete the audio file shown in the editing-screen.
- If no audio file is shown in the editing-screen, if the user is authenticated, he/she can request the audio file corresponding to the pronunciation of the Vietnamese word shown in the editing-screen, and save it in the internal storage.
- If an audio file is shown in the editing screen, the user can play it on his/her device.

- **Implementation:**

- **Screen:** EditScreen.kt
  - **Initial Display:** Shows English/Vietnamese words and audio filename (if exists)
  - **Update Feature:** Update button saves changes to Room database
  - **Delete Audio:** Clean audio button removes file from internal storage
  - **Generate Audio:** Generate audio button (shown if no audio exists and user authenticated)
  - **Play Audio:** Play audio button (shown if audio file exists)
  - **Audio Storage:** Internal app storage with MD5 hash filenames
  - **Audio Player:** ExoPlayer (Media3)
  - **Lambda API:** Audio Synthesis Endpoint
- **Code Location:** EditScreen.kt; Utils.kt; FlashCardDao.kt (Update and GetById Methods); NetworkService.kt; DataTypes.kt

## 7.1.6 6. Study Flashcards

✓ **Status:** Implemented

- **Requirement:** The user can request the app to create a "lesson" for him/her to study. A lesson is a group of 3 flashcards randomly chosen from the app's (Room) database. When studying a lesson, the app will show first the flashcard's English word. Then, if the user asks to see the flashcard's Vietnamese word (e.g., by clicking on the flashcard's English word), the flashcard's Vietnamese word will be shown. When the Vietnamese word is shown:

- The user can move to study the "next" flashcard in the lesson (in a loop).
- The user can play the audio file corresponding to the pronunciation of the flashcard's Vietnamese word, if exists in the app's internal storage.
- If the audio file corresponding to the pronunciation of the flashcard's Vietnamese word does not exist in the app's internal storage, if the user is authenticated, he/she can request the audio, and save it in his/her app's internal storage.

- **Implementation:**

- **Screen:** StudyScreen.kt
- **Lesson Size:** 3 flashcards randomly chosen from Room database
- **Mode:** STUDY\_SESSION enum value
- **Initial Display:** Shows English word first
- **Reveal Mechanism:** Tap card to reveal Vietnamese word
- **Navigation:** Next button moves to next flashcard (loops back to first)
- **Play Audio:** Play audio button (if audio exists in internal storage)
- **Generate Audio:** Generate audio button (if authenticated and audio doesn't exist)
- **Audio Caching:** Checks local cache first, then API fallback
- **Audio Helpers:** Utils.kt provides 'toMd5', 'getCachedAudioFile', and 'saveAudioToInternalStorage' used by StudyScreen playback/generation
- **Audio Playback:** ExoPlayer with state callbacks

- **Code Location:** StudyScreen.kt; FlashCardDao.kt (GetAll Method); Utils.kt; NetworkService.kt; DataTypes.kt