MenuAnNam Mobile App — Code Notes

Nguyen Thien Nguyen - 10422059

January 21, 2026

Contents

1 Use Case: Request Authentication Token

1.1 LoginScreen.kt

Purpose: Request token from Lambda; send token to email.

- Composable function: lines **27–32**; state at line **34**; coroutine scope at line **35**.
- UI layout: lines **40–43** Column container; email TextField at lines **44–53**.
- Submit button: lines **54–72** launches coroutine; calls ‘networkService.generateToken()‘ at lines **57–60**.
- Response handling: lines **61–65** navigate on success; lines **66–68** display error; lines **69–71** exception catch.

1.2 NetworkService.kt

Purpose: Retrofit interface with Lambda endpoints.

- Interface: line **6** with suspend functions.
- Token endpoint: lines **8–13** ‘@PUT‘ at line **9** ‘https://egsbwqh7kildllpkijk6nt4soq0wlgpe.lambda-url.ap-southeast-1.on.aws/‘.
- Audio endpoint: lines **15–20** ‘@PUT‘ at line **16** ‘https://ityqwv3rx5vifjpyufgnpkv5te0ibrcx.lambda-url.ap-southeast-1.on.aws/‘.

1.3 DataTypes.kt

Purpose: Serializable data classes for Lambda payloads.

- UserCredential: line **6** email field.
- TokenResponse: lines **9–11** code and message.
- AudioRequest: lines **14–18** word, email, token fields.
- AudioResponse: lines **21–24** code and Base64 MP3 or error.

1.4 Routes.kt

Purpose: Navigation route definitions.

- LoginRoute: line **20** object for authentication screen.
- TokenRoute: line **29** data class with email parameter.

1.5 Navigator.kt (Login Route)

Purpose: Route implementation and navigation.

- LoginRoute composable: lines **111–122** instantiates LoginScreen with networkService; navigateToToken callback at line **118**.

2 Use Case: Save Email/Token Pair

2.1 TokenScreen.kt

Purpose: Save email/token pair to DataStore after token received.

- Composable function: lines **29–34**; coroutine scope at line **36**; context at lines **37–38**; token state at line **39**.
- UI layout: lines **45–50** Column; token TextField at lines **51–58**; email TextField at lines **59–66**.
- Save button: lines **62–73** launches coroutine at line **66**.
- DataStore save: lines **67–72** ‘dataStore.edit‘ saves EMAIL and TOKEN keys.
- Navigation: line **78** calls ‘navigateToHome(token)‘.

2.2 MainActivity.kt (DataStore Setup)

Purpose: Initialize DataStore singleton and credential keys.

- DataStore delegate: line **20** ‘preferencesDataStore(name = ”user_credentials”)‘.
- TOKEN key: line **21** ‘stringPreferencesKey(”token”)‘.
- EMAIL key: line **22** ‘stringPreferencesKey(”email”)‘.

2.3 Routes.kt

Purpose: Navigation route definitions.

- TokenRoute: line **29** data class with email parameter.
- HomeRoute: line **9** object for main menu.

2.4 Navigator.kt (Token Route)

Purpose: Route implementation and navigation.

- TokenRoute composable: lines **228–243** extracts email parameter at line **232**; navigateToHome callback at lines **236–240**.

3 Use Case: Add Flashcard

3.1 AddScreen.kt

Purpose: Insert flashcard with Room IGNORE strategy for duplicate prevention.

- Composable function: lines **16–21**; state variables at lines **22–23**.
- UI layout: lines **28–36** Column; English TextField at lines **37–45**; Vietnamese TextField at lines **47–55**.
- Add button: lines **59–72** try-catch block; calls ‘insertFlashCard(FlashCard(0, english, vietnamese))‘ at line **62**; handles duplicates at lines **66–68**.
- Clear button: lines **75–83** resets fields at lines **77–78**.

3.2 FlashCard.kt

Purpose: Room entity with unique constraint on English/Vietnamese pair.

- Entity: lines **11–16** ‘@Entity‘ with unique index on english_card and vietnamese_card.
- Data class: line **14** ‘FlashCard‘ with uid, englishCard, vietnameseCard.

3.3 FlashCardDao.kt (Insert Methods)

Purpose: DAO with IGNORE strategy for duplicates.

- Interface: line **12** ‘@Dao interface FlashCardDao‘.
- Insert single: lines **32–33** ‘@Insert(onConflict = IGNORE)‘; returns row id or -1.
- Insert batch: lines **35–36** ‘insertAll(vararg flashCard: FlashCard)‘.

3.4 Routes.kt

Purpose: Navigation route definitions.

- AddCardRoute: line **12** object for add flashcard screen.

3.5 Navigator.kt (Add Route)

Purpose: Route implementation with insertFlashCard callback.

- AddCardRoute composable: lines **137–151** instantiates AddScreen; insertFlashCard callback at lines **140–147** calls ‘flashCardDao.insertAll()‘ at line **143**.

3.6 MenuDatabase.kt

Purpose: Room database singleton.

- Database annotation: line **12** ‘@Database‘ with FlashCard entity.
- Singleton: lines **20–30** ‘getDatabase()‘ with double-checked locking.

4 Use Case: Search Flashcards

4.1 FilterScreen.kt

Purpose: Capture search filters with 4 exact/partial match combinations.

- Composable function: lines **24–29**; state variables at lines **30–33**.
- UI layout: lines **37–45** Column; English row at lines **47–65**; Vietnamese row at lines **67–82**.
- Search button: lines **87–104**; converts booleans to 0/1 at lines **92–93**; calls onSearch at lines **95–96**.

4.2 SearchScreen.kt

Purpose: Display filtered flashcards in LazyColumn with Edit and Delete buttons.

- FlashCardList component: lines **37–88** LazyColumn with card rows; Edit button at lines **71–77**; Delete button at lines **78–83**.
- SearchScreen composable: lines **91–104**; state at lines **105–107**; coroutine scope at line **108**.
- performSearch lambda: lines **110–122** calls ‘flashCardDao.getFilteredFlashCards()‘ at line **114**.
- UI layout: lines **128–135** Column; loading state at lines **137–139**.
- FlashCardList call: lines **142–151** onDelete handler calls ‘flashCardDao.delete()‘ and refreshes.

4.3 FlashCardDao.kt (Filter Query)

Purpose: Execute parameterized CASE WHEN query for 4 filter combinations.

- Query method: lines **60–63** ‘@Query‘ with SQL CASE WHEN logic for exact/partial match.
- Parameters: ‘en‘, ‘exactEn‘, ‘vn‘, ‘exactVn‘.
- Return: ‘List<FlashCard>‘ with matching records.

4.4 Routes.kt

Purpose: Navigation route definitions.

- FilterRoute: line **32** object for search input screen.
- SearchCardsRoute: lines **12–17** data class with 4 search parameters.
- EditCardRoute: line **26** data class with cardId.

4.5 Navigator.kt (Search Routes)

Purpose: Route implementations for filter and search.

- FilterRoute composable: lines **166–177** instantiates FilterScreen; onSearch callback at lines **169–176** navigates to SearchCardsRoute.
- SearchCardsRoute composable: lines **153–181** extracts route parameters at line **154**; passes to SearchScreen; onEdit callback at line **160** navigates to EditCardRoute.

5 Use Case: Edit Flashcard

5.1 EditScreen.kt

Purpose: Update flashcard text and manage audio files.

- Composable function: lines **43–48**; context and scope at lines **49–51**; state at lines **53–58**.
- Load flashcard: lines **60–88** ‘LaunchedEffect‘ fetches card at line **62**; checks audio at lines **69–74**.
- UI layout: lines **110–289** Column with TextFields at lines **113–131**.
- Update button: lines **147–167** calls ‘flashCardDao.update()‘ at lines **151–155**.
- Clean audio: lines **169–195** deletes file at line **175**.
- Play audio: lines **197–233** creates ExoPlayer at line **204**; plays at lines **220–221**.
- Generate audio: lines **235–289** loads credentials at lines **240–243**; calls ‘networkService.generateAudio()‘ at lines **250–253**; decodes Base64 at line **256**; saves with MD5 filename at lines **257–258**.

5.2 Utils.kt

Purpose: MD5 hashing, audio storage, cache checking.

- MD5 extension: lines **8–11** converts string to MD5 hex digest.
- Save audio: lines **13–18** writes bytes to internal storage; returns File.
- Cache check: lines **21–24** checks if ‘${word.toMd5()}.mp3‘ exists; returns File or null.

5.3 FlashCardDao.kt (Update and GetById Methods)

Purpose: Update flashcard and fetch by ID.

- Update: lines **51–54** ‘@Query‘ UPDATE statement with id, english, vietnamese parameters.
- GetById: line **26** ‘SELECT WHERE uid = :id‘; returns nullable FlashCard.
- Delete: lines **27–28** ‘@Delete‘ method.

5.4 NetworkService.kt

Purpose: Retrofit interface for audio synthesis.

- Audio endpoint: lines **15–20** ‘@PUT‘ with audio synthesis URL.

5.5 DataTypes.kt

Purpose: Audio API payloads.

- AudioRequest: lines **14–18** word, email, token fields.
- AudioResponse: lines **21–24** code and Base64 MP3.

5.6 MainActivity.kt (DataStore)

Purpose: DataStore for EMAIL/TOKEN credentials.

- DataStore delegate: line **20** singleton.
- TOKEN key: line **21** for audio authentication.
- EMAIL key: line **22** for audio authentication.

5.7 Routes.kt

Purpose: Navigation route definitions.

- EditCardRoute: line **26** data class with cardId.

5.8 Navigator.kt (Edit Route)

Purpose: Route implementation.

- EditCardRoute composable: lines **183–197** extracts cardId at line **184**; passes networkService and flashCardDao; onCardUpdated callback at line **189**.

6 Use Case: Study Flashcards

6.1 StudyScreen.kt

Purpose: Random lesson generation, card flip, audio playback with caching.

- CardViewMode enum: lines **53–56** SINGLE_CARD and STUDY_SESSION modes.
- Composable function: lines **60–68**; state at lines **71–76**.
- Load data: lines **78–114** ‘LaunchedEffect‘; SINGLE_CARD at lines **82–91**; STUDY_SESSION at lines **92–105** creates 3-card lesson at lines **94–98**.
- playAudio lambda: lines **116–203** cache check at lines **124–125**; cached playback at lines **126–149**; load credentials at lines **150–157**; API call at lines **161–164**; decode and save at lines **167–169**; ExoPlayer at lines **171–190**.
- SINGLE_CARD UI: lines **224–287** card display at lines **226–264**; Delete button at lines **269–282**; Play button at lines **284–288**.
- STUDY_SESSION UI: lines **289–358** flip card at lines **294–316**; Next button at lines **318–330**; Play button at lines **332–341**.

6.2 Utils.kt

Purpose: Audio cache management.

- MD5: lines **8–11** generates hex filenames.
- Save: lines **13–18** persists MP3 bytes.
- Cache check: lines **21–24** returns File or null.

6.3 FlashCardDao.kt (GetAll and GetById Methods)

Purpose: Fetch flashcards for lesson generation and single card view.

- GetAll: line **17** ‘SELECT * FROM FlashCards‘; returns all cards.
- GetById: line **26** ‘SELECT WHERE uid = :id‘; returns nullable FlashCard.
- Delete: lines **27–28** ‘@Delete‘ for SINGLE_CARD mode.

6.4 NetworkService.kt

Purpose: Retrofit interface for audio synthesis.

- Audio endpoint: lines **15–20** ‘@PUT‘ with audio synthesis URL.

6.5 DataTypes.kt

Purpose: Audio API payloads.

- AudioRequest: lines **14–18** word, email, token fields.
- AudioResponse: lines **21–24** code and Base64 MP3.

6.6 MainActivity.kt (DataStore)

Purpose: DataStore for EMAIL/TOKEN credentials.

- DataStore delegate: line **20** singleton.
- TOKEN key: line **21** for audio authentication.
- EMAIL key: line **22** for audio authentication.

6.7 Routes.kt

Purpose: Navigation route definitions.

- StudyCardsRoute: line **9** object for study session.
- ShowCardRoute: line **23** data class with cardId for single card view.

6.8 Navigator.kt (Study Routes)

Purpose: Route implementations.

- StudyCardsRoute composable: lines **124–135** instantiates StudyScreen in STUDY_SESSION mode; passes networkService and coroutineScope.
- ShowCardRoute composable: lines **210–227** extracts cardId at line **211**; instantiates StudyScreen in SINGLE_CARD mode; onCardDeleted callback at line **218**.

7 Use Case: Audio Cache & Regeneration

7.1 Utils.kt

Purpose: Audio caching with deterministic filenames.

- MD5 extension: lines **8–11** converts word to 32-char hex digest for cache consistency.
- Save audio: lines **13–18** writes Base64-decoded MP3 to internal storage.
- Cache check: lines **21–24** checks if '${word.toMd5()}.mp3' exists; returns File or null.

7.2 NetworkService.kt (Audio Endpoint)

Purpose: Retrofit interface for Lambda audio synthesis.

- Audio endpoint: lines **15–20** '@PUT' at line **16** with default URL.

7.3 DataTypes.kt (Audio Payloads)

Purpose: Request/response for audio endpoint.

- AudioRequest: lines **14–18** word, email, token fields.
- AudioResponse: lines **21–24** code and Base64 MP3 or error.

7.4 EditScreen.kt (Audio Cache Flow)

Purpose: Cache-check-then-fetch pattern.

- Generate audio: (detailed implementation); loads credentials; calls 'networkService.generateAudio()'; decodes Base64; saves with MD5 filename.

7.5 StudyScreen.kt (Audio Cache Flow)

Purpose: Cache-first playback logic.

- playAudio lambda: lines **116–203** cache check at lines **124–125**; if cached: immediate playback at lines **126–149**; if not: fetch from Lambda at lines **161–163**; decode and save at lines **167–169**.

8 Use Case: Navigation with Back Button and Status Feedback

8.1 TopBarComponent.kt

Purpose: Display title with conditional back button.

- Composable function: lines **15–17**; TopAppBar at lines **18–41**.
- Back button: lines **20–30** conditional IconButton; calls ‘showBack()‘ at line **23**.
- Colors: lines **34–40** Material3 theming.

8.2 BottomBarComponent.kt

Purpose: Display status messages for feedback.

- Composable function: lines **12–14**; Box container at line **15**.
- Status Text: lines **16–22** displays message parameter with center alignment.

8.3 Navigator.kt (Back Navigation)

Purpose: Manage navigation stack and shared message state.

- Composable function: lines **28–33**; shared message state at lines **34–35**.
- changeMessage callback: lines **37–39** updates shared state.
- Scaffold structure: lines **41–246** TopBar at lines **42–48**; BottomBar at line **43**; NavHost at lines **50–245**.
- Back button: All routes except HomeRoute show back button calling ‘navigation.popBackStack()‘.

8.4 Routes.kt

Purpose: Type-safe navigation routes with ‘@Serializable‘.

- HomeRoute: line **9** main menu.
- SearchCardsRoute: lines **18–23** data class with 4 search parameters.
- EditCardRoute: line **32** data class with cardId.
- TokenRoute: line **35** data class with email.

9 Supporting Infrastructure

9.1 Navigator.kt (AppNavigation)

Purpose: Centralized navigation with type-safe routes and dependencies.

- Composable function: lines **28–33**; shared message at lines **35–36**.
- Scaffold: lines **60–88** TopBar at lines **62–67**; BottomBar at lines **68–70**; NavHost at lines **73–246**.
- Route implementations: HomeRoute at lines **78–89**; LoginRoute at lines **91–102**; TokenRoute at lines **190–200**; AddCardRoute at lines **116–130** with insertFlashCard callback; FilterRoute at lines **132–142**; SearchCardsRoute at lines **144–172** with parameter extraction; EditCardRoute at lines **174–188**; StudyCardsRoute at lines **104–114**; ShowCardRoute at lines **202–219**.

9.2 Routes.kt

Purpose: Type-safe route definitions with parameters.

- HomeRoute: line **9** main menu.
- AddCardRoute: line **12** add flashcard.
- StudyCardsRoute: line **15** study session.
- SearchCardsRoute: lines **18–23** data class with 4 search parameters.
- LoginRoute: line **26** authentication.
- ShowCardRoute: line **29** data class with cardId.
- EditCardRoute: line **32** data class with cardId.
- TokenRoute: line **35** data class with email.
- FilterRoute: line **38** search input.

9.3 MenuScreen.kt

Purpose: Main menu with navigation buttons.

- Composable function: lines **15–19** callbacks for navigation.
- UI layout: lines **25–84** Column with 4 buttons.
- Add Card button: lines **28–34** calls ‘onAdd()‘.
- Study Cards button: lines **36–42** calls ‘onStudy()‘.
- Search Cards button: lines **44–50** calls ‘onSearch()‘.

9.4 MenuDatabase.kt

Purpose: Room database definition and singleton.

- Database annotation: lines **12–14** ‘@Database‘ with FlashCard entity.
- Abstract class: lines **15–16** extends RoomDatabase; DAO accessor.
- Companion singleton: lines **17–30** double-checked locking at lines **21–29**.

9.5 FlashCardDao.kt

Purpose: Data access object for all CRUD operations.

- Interface: line **14** ‘@Dao‘.
- GetAll: line **19** ‘SELECT * FROM FlashCards‘.
- GetById: line **28** ‘SELECT WHERE uid = :id‘.
- Delete: lines **29–30** ‘@Delete‘.
- Insert single: lines **34–35** ‘@Insert(onConflict = IGNORE)‘.
- Insert batch: lines **37–38** ‘insertAll(vararg)‘.
- Update: lines **40–50** ‘@Query UPDATE‘ with id, english, vietnamese.
- Filter query: lines **52–82** CASE WHEN logic for exact/partial match.

9.6 MainActivity.kt

Purpose: Initialize database, Retrofit, and DataStore.

- DataStore setup: line **17** singleton delegate; keys at lines **18–19**.
- onCreate: lines **25–63** initialize MenuDatabase at lines **28–29**; Retrofit at lines **33–45**; AppNavigation call at lines **55–60**.