POLITEHNICA UNIVERSITY OF BUCHAREST

SOFTWARE ENGINEERING

PROJECT

Software Design Document

Team:

Sakka Mohamad-Mario

Coordinator:

N/A

Date created:

15.11 .2025

DELIVERY REPORT

Name	Group	Project implementation [%, reason]	Signature
Sakka Mohamad-Mario	1241EB	_____ _____ _____	

Delivery date:

_____

Table of Contents

SYSTEM DESIGN

According to the IEEE STD-1016-1998, *IEEE Recommended Practice for Software Design Descriptions*.

Document Change History

Version	Date	Author(s)	Details of changes
1.14	2025-11-16	Sakka Mohamad-Mario	Table headers bold throughout; reference access dates set between 05–10 Nov 2025.
1.13	2025-11-16	Sakka Mohamad-Mario	Reformatted all modules/classes/interfaces/processes to tables per template; diagrams as PlantUML.
1.12	2025-11-16	Sakka Mohamad-Mario	Expanded Section 4; exact formatting for Sections 5 & 6; full references.
1.10	2025-11-16	Sakka Mohamad-Mario	Initial full-code SDD (Android + Node server).
1.00	2025-10-15	Sakka Mohamad-Mario	Initial baseline based on SRS + partial code.

1.1. Purpose

State the purpose of this document: the presentation of the design of a system that solves the requirements of the proposed project. This SDD reflects the **actual** code in the Android client (Kotlin) and the Node/Express backend, including Room entities/DAOs, background services, WorkManager, Retrofit HTTP client, and server routes.

1.2. Target Public

Describe the people that this document is addressed to: programmers, designers, and project managers working on RecipeGPT. The client/end-user is not part of the target public.

1.3. Definitions, Acronyms and Abbreviations

Field	Description
API	Application Programming Interface
APK/AAB	Android application package / app bundle
CI/CD	Continuous Integration / Continuous Deployment
DAO	Data Access Object (Room)
DB	Database (Room/SQLite)
DTO	Data Transfer Object
GCP	Google Cloud Platform

JSON	JavaScript Object Notation
LLM	Large Language Model
Room	Android persistence library
TLS/HTTPS	Transport Layer Security / HTTPS
UoM	Unit of Measure (grams, ml, pcs)
WorkManager	Android background scheduler

1.4. Document Structure

Chapter 1 – Introduction
Presents the purpose of the system and this document, the intended audience, key terminology, and an overview of how the SDD is organized.

Chapter 2 – References
Lists the standards, official documentation, and external resources that were used when designing and documenting RecipeGPT.

Chapter 3 – Decomposition Description
Describes how the system is broken down into modules, concurrent processes, and data modules, giving a conceptual map of the overall architecture.

Chapter 4 – Dependency Description
Details the dependencies and interactions between modules, processes, and data modules, including how they cooperate at runtime.

Chapter 5 – Interface Description
Specifies the interfaces of the main modules and processes, including their inputs, outputs, and observable behavior, in a more formal and precise way.

Chapter 6 – Detailed Design
Provides the low-level design of modules and data structures, including classes, methods, algorithms, and internal responsibilities.

Appendices
Contain supporting material such as use case diagrams, class and sequence diagrams, document evolution notes, and a short conclusion on the design activity.

---

2. References

[1] IEEE Std 1016-1998. IEEE Recommended Practice for Software Design Descriptions. IEEE. https://cs.bilkent.edu.tr/~cagatay/cs413/1016-1998_00741934.pdf (Accessed: 05 Nov 2025).

[2] Getting started with WorkManager. Android Developers. https://developer.android.com/develop/background-work/background-tasks/persistent/getting-started (Accessed: 06 Nov 2025).

[3] WorkManager API reference. Android Developers. https://developer.android.com/reference/androidx/work/WorkManager (Accessed: 07 Nov 2025).

[4] Save data in a local database using Room. Android Developers. https://developer.android.com/training/data-storage/room/ (Accessed: 08 Nov 2025).

[5] Define data using Room entities. Android Developers. https://developer.android.com/training/data-storage/room/defining-data (Accessed: 09 Nov 2025).

[6] Foreground services overview. Android Developers. https://developer.android.com/develop/background-work/services/fgs (Accessed: 10 Nov 2025).

[7] Retrofit — A type-safe HTTP client for Android and Java. Square (GitHub). https://github.com/square/retrofit (Accessed: 05 Nov 2025).

[8] OkHttp — Square's HTTP client. Square (GitHub). https://github.com/square/okhttp (Accessed: 06 Nov 2025).

[9] Express — Node.js web application framework. Express.js. https://expressjs.com/ (Accessed: 07 Nov 2025).

[10] Node.js Documentation. Node.js. https://nodejs.org/api/all.html (Accessed: 08 Nov 2025).

[11] Cloud Run documentation. Google Cloud. https://docs.cloud.google.com/run/docs (Accessed: 09 Nov 2025).

[12] GitHub Actions documentation. GitHub Docs. https://docs.github.com/en/actions (Accessed: 10 Nov 2025).

[13] Workflow syntax for GitHub Actions. GitHub Docs. https://docs.github.com/en/actions/reference/workflows-and-actions/workflow-syntax (Accessed: 05 Nov 2025).

3. Decomposition Description

3.1. Modules Description

3.1.1. Description of Module 1

Field	Description
Name	MobileApp (Android/Kotlin)
Type	Code module
Purpose	Provide UI/UX; orchestrate recipe generation; local persistence; notifications.
Way of operating	Activities host navigation; Fragments+ViewModels manage state; Services execute work; Room for persistence; WorkManager for periodic jobs; Retrofit/OkHttp for HTTP.
Subordination	Activities (HomeActivity, RecipeDetailsActivity); Services (GenerateRecipeService, DatabaseBackgroundService, SearchNotificationForegroundService); Worker (RandomQuoteWorker); Data (AppDatabase, RecipeEntity, IngredientEntity; RecipeDao, IngredientDao); Network (ApiClient, ApiInterface); Utils (NotificationUtils, SharedPreferencesManager); Adapters (RecipeAdapter, etc.).
Dependencies	Android SDK; Room; Retrofit/OkHttp; WorkManager; LocalBroadcastManager; Notifications.
Resources	SQLite (Room), Notification channels, Internet

3.1.2. Description of Module 2

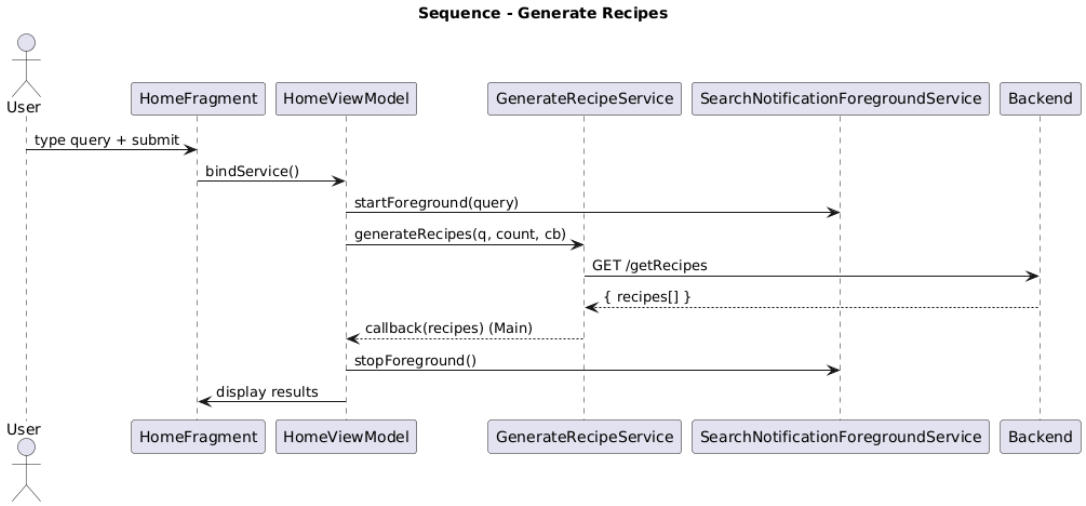
Field	Description
Name	BackendAPI (Node/Express)
Type	Process / Code module
Purpose	Validate inputs; call OpenAI; return JSON schemas for recipes and random quotes.
Way of operating	Express GET endpoints with query parameters; JSON responses; invokes OpenAI client using schema-guided prompts.

Subordination	server.js (route handlers); schemas.js (JSON schemas); env config (.env); OpenAI client.
Dependencies	Node.js; Express; OpenAI API; dotenv; HTTPS.
Resources	Node 18+, network connectivity, API key

3.2. Description of Concurrent Processes

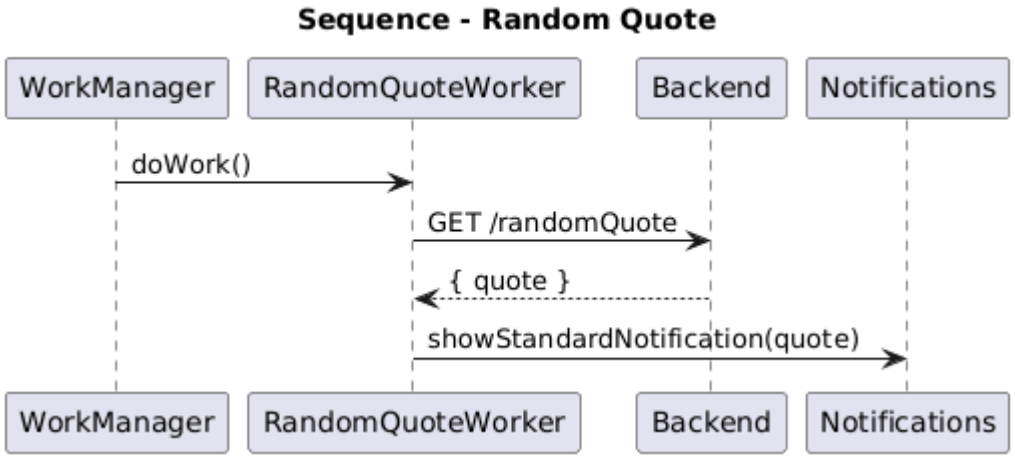
3.2.1. Description of Process 1

Field	**Description**
Name	GenerateRecipesForegroundProcess
Type	Process
Purpose	Keep UI responsive during recipe generation; show persistent progress; handle timeouts/errors.
Way of operating	UI → ViewModel → bind GenerateRecipeService → dynamic timeout call → callback on main thread → stop foreground.
Dependencies	Backend availability; Permissions & channel for notifications
Resources	Foreground service notification; Retrofit client



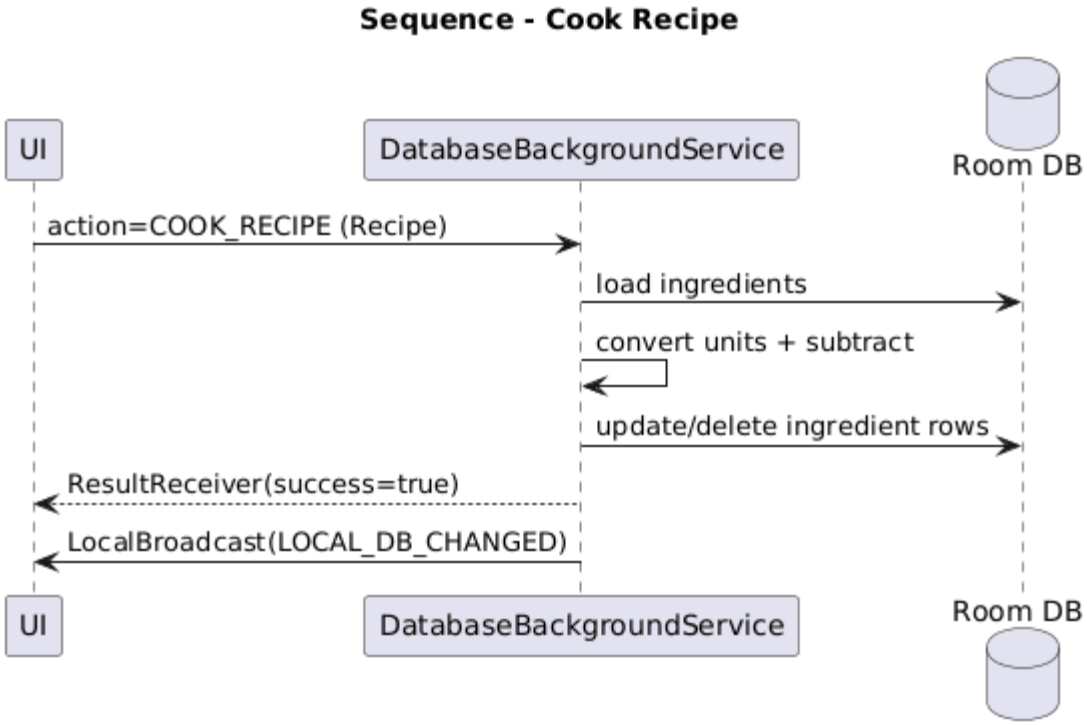
3.2.2. Description of Process 2

Field	Description
Name	QuotesPeriodicProcess
Type	Process
Purpose	Periodically fetch and display a random cooking quote.
Way of operating	WorkManager runs RandomQuoteWorker; Retrofit call; show success/error notification.
Dependencies	Notification channel; preferences-driven schedule
Resources	Worker runtime; Retrofit client



3.2.3. Description of Process 3

Field	Description
Name	DBConsistencyProcess
Type	Process
Purpose	Encapsulate DB operations and enforce consistency rules (save/update, canCook, cook).
Way of operating	UI sends Intent to DatabaseBackgroundService; DAO calls on IO; reply via ResultReceiver; broadcast LOCAL_DB_CHANGED
Dependencies	Room (RecipeDao, IngredientDao); UnitConverter
Resources	SQLite, LocalBroadcastManager



3.3. Description of Data Modules

3.3.1. Description of Data Module 1

Field	Description
Name	RecipeEntity
Type	Data module (Room Entity, table 'recipes')
Purpose	Persist generated/saved recipes and the listed-for-shopping flag.
Way of operating	Accessed via RecipeDao; complex fields (ingredients, instructions) serialized with Gson TypeConverters.
Subordination	Declared in AppDatabase.entities; used by DatabaseBackgroundService via RecipeDao.
Dependencies	Room; GsonConverters; models.Recipe; models.Ingredient
Resources	SQLite (Room)

Field	Type	Notes
title	String	PRIMARY KEY; unique recipe identifier
estimatedCookingTime	Int	Minutes
servings	Int	Suggested servings

listed	Boolean	Shopping-list marker
ingredients	List<Ingredient>	Serialized via GsonConverters
instructions	List<String>	Serialized via GsonConverters

3.3.2. Description of Data Module 2

Field	**Description**
Name	IngredientEntity
Type	Data module (Room Entity, table 'ingredients', composite PK: item+unit)
Purpose	Persist inventory items with per-unit quantities to support cooking and shopping flows.
Way of operating	Accessed via IngredientDao; names normalized (lowercase) at insert; unit conversions handled at service layer.
Subordination	Declared in AppDatabase.entities; used by DatabaseBackgroundService via IngredientDao.
Dependencies	Room; models.Ingredient; UnitConverter (at call sites)
Resources	SQLite (Room)

Field	**Type**	**Notes**
item	String	Composite PK (part); lowercased
unit	String	Composite PK (part); UoM label
amount	Double	Quantity in given unit; non-negative

3.3.3. Description of Data Module 3

Field	**Description**
Name	Recipe (model)
Type	Data module (Domain model)
Purpose	In-memory representation of a recipe used by UI and services.
Way of operating	Mapped to/from RecipeEntity; transported

	via Intents/ResultReceiver bundles.
Subordination	Used by UI, GenerateRecipeService, DatabaseBackgroundService.
Dependencies	models.Ingredient; RecipeEntity (mapping)
Resources	RAM (in-memory)

3.3.4. Description of Data Module 4

Field	**Description**
Name	Ingredient (model)
Type	Data module (Domain model)
Purpose	In-memory representation of an inventory item with amount and unit.
Way of operating	Mapped to/from IngredientEntity; used in canCook/cook logic.
Subordination	Used by UI and DatabaseBackgroundService.
Dependencies	QuantUnit (via UnitConverter at call sites)
Resources	RAM (in-memory)

3.3.5. Description of Data Module 5

Field	**Description**
Name	RecipeDao
Type	Data module (Data Access Object, Room DAO)
Purpose	Provide CRUD operations for RecipeEntity.
Way of operating	Suspend functions executed on IO dispatcher; used by DatabaseBackgroundService.
Subordination	Exposed by AppDatabase.recipeDao()

Dependencies	Room; RecipeEntity
Resources	SQLite (Room)

3.3.6. Description of Data Module 6

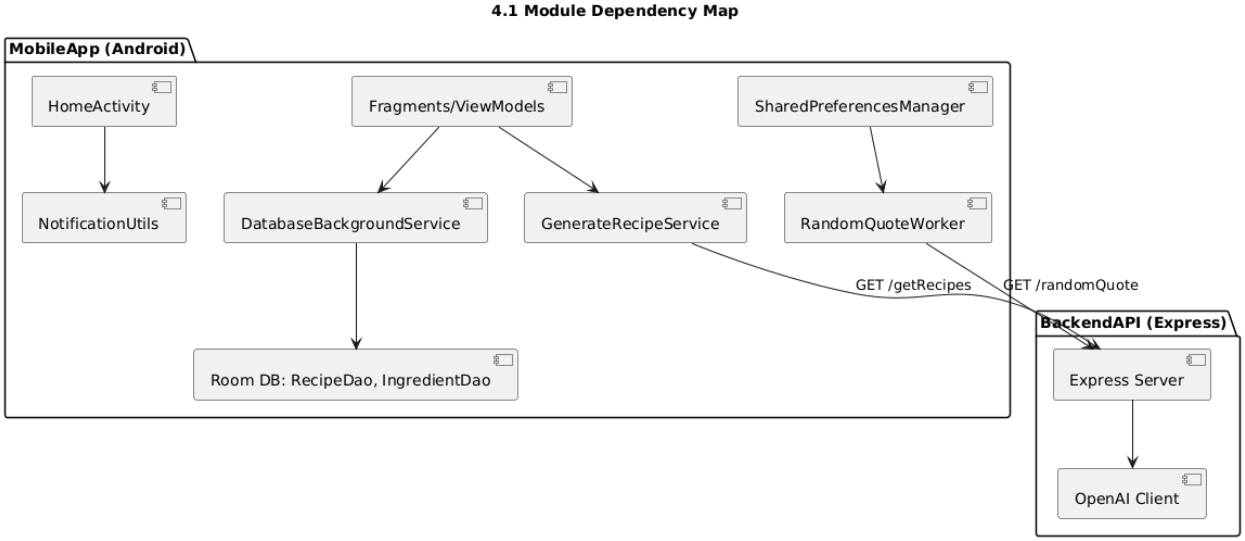
Field	**Description**
Name	IngredientDao
Type	Data module (Data Access Object, Room DAO)
Purpose	Provide CRUD operations for IngredientEntity.
Way of operating	Suspend functions executed on IO dispatcher; used by DatabaseBackgroundService.
Subordination	Exposed by AppDatabase.ingredientDao()
Dependencies	Room; IngredientEntity
Resources	SQLite (Room)

4. Dependency Description

4.1. Dependencies among modules

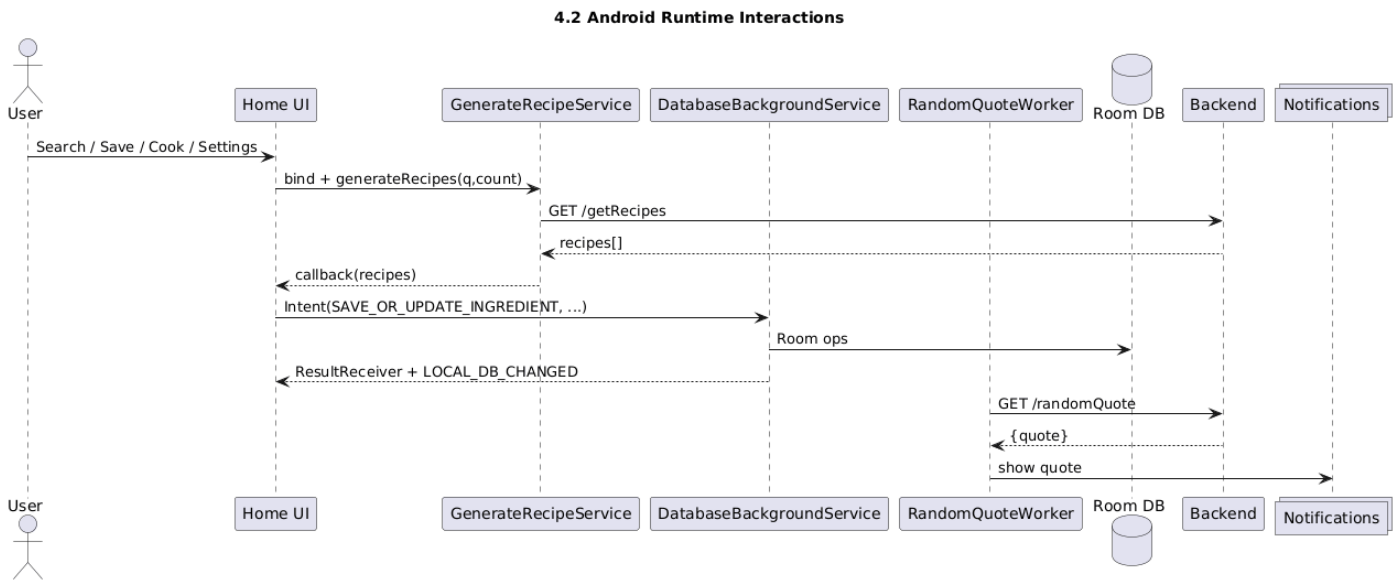
Origin	Target	Contract / API	Direction	Sync/ Async	Failure modes	Handling & Notes
HomeActivity	NotificationUtils	Permission & Channel creation	→	Sync	Permission denied; channel missing	Toast and channel creation at startup
UI (Fragments/View Models)	GenerateRecipeService	Binder + Kotlin callback	→	Async (callback on Main)	Timeout; HTTP errors	Dynamic timeout; notifications; empty list on non-200
UI (Fragments/View Models)	DatabaseBackgroundService	Intent + ResultReceiver + LocalBroadcast	↔	Async	DB contention; null inputs	IO dispatcher; broadcast DB_CHANGED
SharedPreferencesManager	RandomQuoteWorker	ACTION_SETTINGS_UPDATED broadcast	→	Async	Out-of-date schedule	Reschedule worker
RandomQuoteWorker	Backend	Retrofit GET /randomQuote	→	Sync	Timeout; 4xx/5xx	10s timeout; error notification
GenerateRecipeService	Backend	Retrofit GET /getRecipes	→	Sync	Timeout; 4xx/5xx	5s + 10s/recipe (≤120s); error notifications
DatabaseBackgroundService	RecipeDao	Room DAO	→	Sync	Constraint errors	REPLACE insert; targeted deletes
DatabaseBackgroundService	IngredientDao	Room DAO	→	Sync	Unit mismatch	UnitConverter.convert/add; normalize unit
Express Server	OpenAI	HTTPS	→	Sync/A	429/5	Backoff/retry

				sync	xx	(impl dependent)



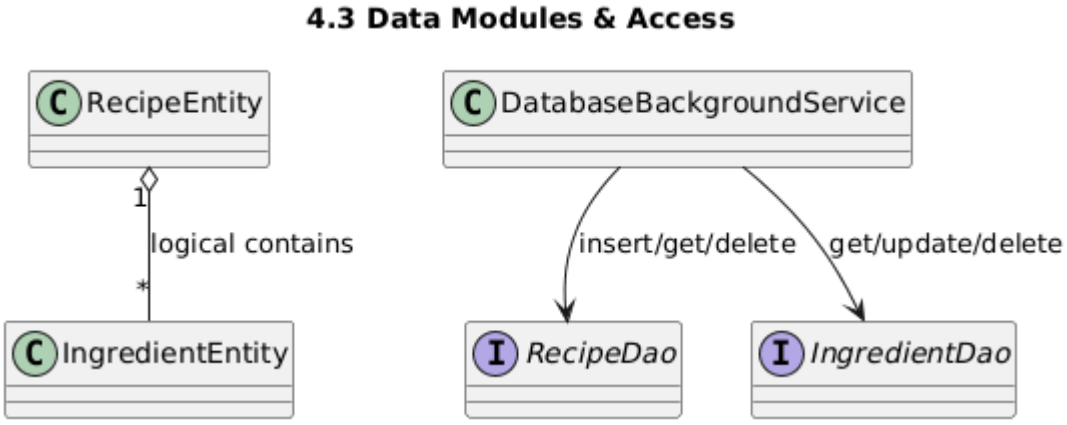
4.2. Dependencies among processes

Process	Trigger	Relies on	Sync points	Timeouts/ Backoff	Cancellation/Stop	Observability
GenerateRecipesForegroundProcess	User submits query	Service binding; Retrofit; Notifications	Callback on main	5s + 10s/recipe (≤120s)	Stop foreground on end	Persistent notif + UI flags
QuotesPeriodicProcess	WorkManager schedule	Retrofit; Notifications; Prefs	doWork() result	~10s default timeout	WorkManager reschedule	Notification message
DBConsistencyProcess	UI Intent actions	Room DAOs; UnitConverter	ResultReceiver	N/A	N/A	LOCAL_DB_CHANGED broadcast



4.3. Dependencies among data modules

Data module	Depends on	Nature	Notes
RecipeEntity	Ingredient (embedded list)	Containment	Stored via GsonConverters
IngredientEntity	UnitConverter	Computation	Normalization and addition across units
DAOs	Room/SQLite	Storage	Suspend functions, IO dispatcher



5. Interface Description

This chapter corresponds to chapter 6.2.3, Interface description, from [1].

5.1. Module Interfaces

5.1.1. Module 1 Interface — BackendAPI (Express)

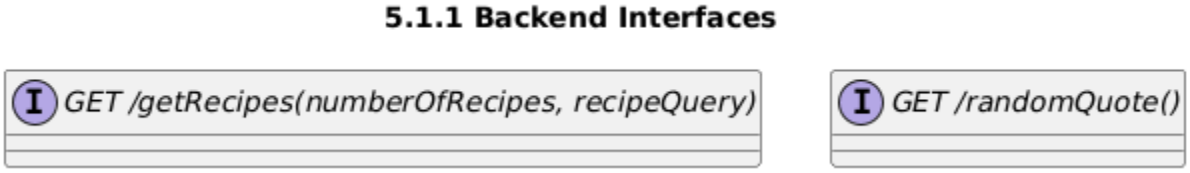
Field	Description
Name	BackendAPI (Express)
Type	Process / Code module
Purpose	Provide HTTP endpoints to generate recipes and fetch a random quote
Way of operating	Express handlers respond to GET requests with JSON; server calls OpenAI

Interface 1 — GET /getRecipes(numberOfRecipes, recipeQuery)

Input	Output	Description
numberOfRecipes:int; recipeQuery:string (query parameters over HTTPS)	{ recipes: Array<Recipe> }	Validate → OpenAI → normalize → JSON

Interface 2 — GET /randomQuote()

Input	Output	Description
—	{ quote: string }	Return concise quote JSON



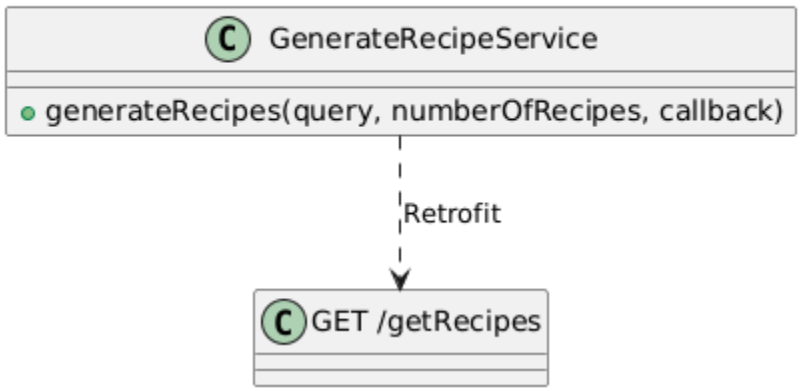
5.1.2. Module 2 Interface — GenerateRecipeService (Android)

Field	Description
Name	GenerateRecipeService
Type	Code module / Service (bound)
Purpose	Execute recipe generation with dynamic timeout and notify on errors
Way of operating	Bound from UI; network call on IO; result via callback on main; notifications for timeouts/errors

Interface — result generateRecipes(query:String, numberOfRecipes:Int, callback:(List<Recipe>) -> Unit)

Input	Output	Description
query:String; numberOfRecipes:Int	callback receives List<Recipe>	Timeout = 5s + 10s/recipe (≤120s); Retrofit → GET /getRecipes; exceptions → NotificationUtils

5.1.2 GenerateRecipeService API

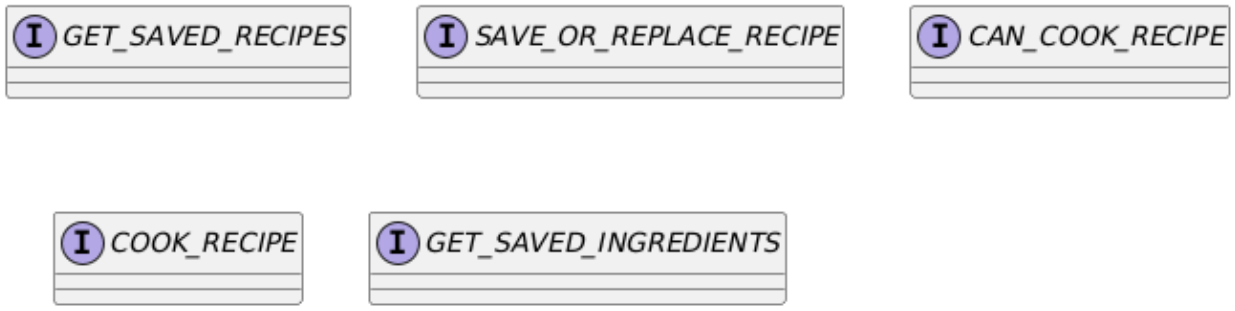


5.1.3. Module 3 Interface — DatabaseBackgroundService (Android)

Field	Description
Name	DatabaseBackgroundService
Type	Code module / Service (background)
Purpose	Serialize DB operations and enforce domain rules; IPC via Intents/ResultReceiver
Way of operating	onStartCommand routes actions; Room on IO; broadcasts LOCAL_DB_CHANGED

Action (Interface)	Input	Output / Description
GET_SAVED_RECIPES	—	ArrayList<Recipe> (Bundle key "data")
GET_RECIPE_BY_NAME	name:String	Recipe? ("data")
SAVE_OR_REPLACE_RECIPE	recipe:Recipe	— ; broadcast LOCAL_DB_CHANGED
DELETE_RECIPE_BY_NAME	name:String	— ; broadcast LOCAL_DB_CHANGED
GET_SAVED_INGREDIENTS	—	ArrayList<Ingredient> ("data")
GET_INGREDIENT_BY_NAME_AND_UNIT	name:String, unit:String	Ingredient? ("data")
SAVE_OR_UPDATE_INGREDIENT	ingredient:Ingredient	merge & normalize; broadcast LOCAL_DB_CHANGED
DELETE_INGREDIENT_BY_NAME_AND_UNIT	name:String, unit:String	— ; broadcast LOCAL_DB_CHANGED
FETCH_LISTED_RECIPES	—	ArrayList<Recipe> ("recipes")
CAN_COOK_RECIPE	recipe:Recipe	canCook:Boolean; missing_ingredients:ArrayList<Ingredient>
COOK_RECIPE	recipe:Recipe	success:Boolean ; broadcast LOCAL_DB_CHANGED

5.1.3 DatabaseBackgroundService - Intent Interfaces



5.1.4. Module 4 Interface — SharedPreferencesManager (Android)

Field	Description
Name	SharedPreferencesManager
Type	Code module / Utility
Purpose	Persist settings and broadcast ACTION_SETTINGS_UPDATED with extras
Way of operating	SharedPreferences writes followed by LocalBroadcast broadcast

getRandomQuoteFrequency()

Input	Output	Description
—	String (default "15 minutes")	Returns current frequency

saveRandomQuoteFrequency(frequency:String)

Input	Output	Description
frequency:String	—	Persists + broadcasts

getMaxResults()

Input	Output	Description
—	Int (default 10)	Returns current max results

saveMaxResults(maxResults:Int)

Input	Output	Description
maxResults:Int	—	Persists + broadcasts

5.1.5. Module 5 Interface — RandomQuoteWorker (Android)

Field	Description
Name	RandomQuoteWorker
Type	Code module / Worker
Purpose	Fetch and notify a random quote at scheduled intervals
Way of operating	WorkManager calls doWork(); Retrofit GET with ~10s timeout; notification on success/error

doWork(): Result

Input	Output	Description
—	Result.success()	Shows notification, returns success

5.1.6. Module 6 Interface — RecipeDao (Room)

Method	Input	Output	Notes
getAllRecipesSync()	—	List<RecipeEntity>	Suspend function
getRecipeByNameSync(na	name:String	RecipeEntity?	Suspend function

me)			
getListedRecipes()	—	List<RecipeEntity>	listed=1
insertRecipe(recipe)	recipe:RecipeEntity	Unit	OnConflictStrategy.REPLACE
deleteRecipeByName(name)	name:String	Unit	—

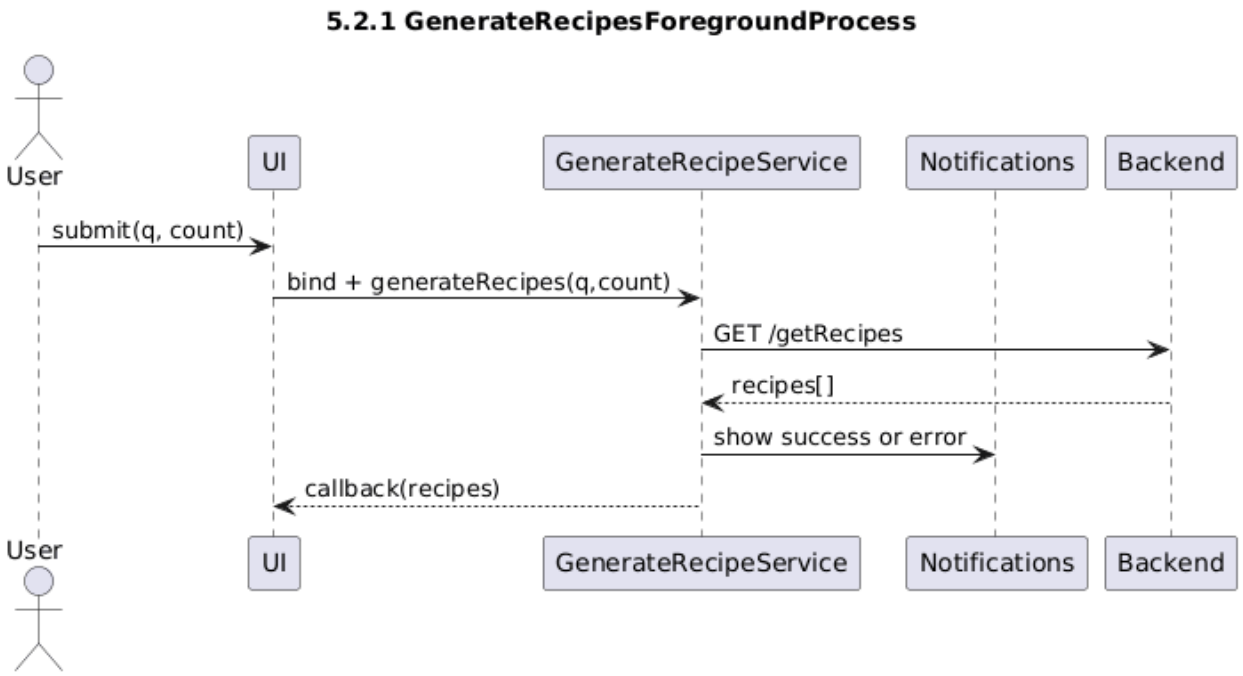
5.1.7. Module 7 Interface — IngredientDao (Room)

Method	Input	Output	Notes
getAllIngredientsSync()	—	List<IngredientEntity>	Suspend function
getIngredientByNameAndUnitSync(name,unit)	name:String, unit:String	IngredientEntity?	Suspend function
insertIngredient(ingredient)	ingredient:IngredientEntity	Unit	OnConflictStrategy.REPLACE
updateIngredient(ingredient)	ingredient:IngredientEntity	Unit	—
deleteIngredientByNameAndUnit(name,unit)	name:String, unit:String	Unit	—

5.2. Processes Interfaces

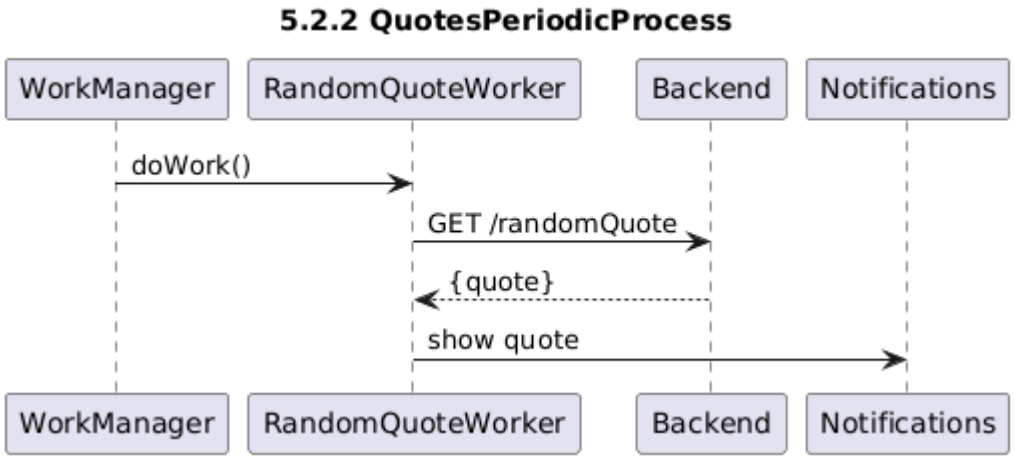
5.2.1. Process 1 Interface — GenerateRecipesForegroundProcess

Field	Description
Trigger	User submits query
Input	(query:String, numberOfRecipes:Int)
Output	UI updates and notifications
Contracts	Service binding; Retrofit GET /getRecipes; callback on main; stop foreground



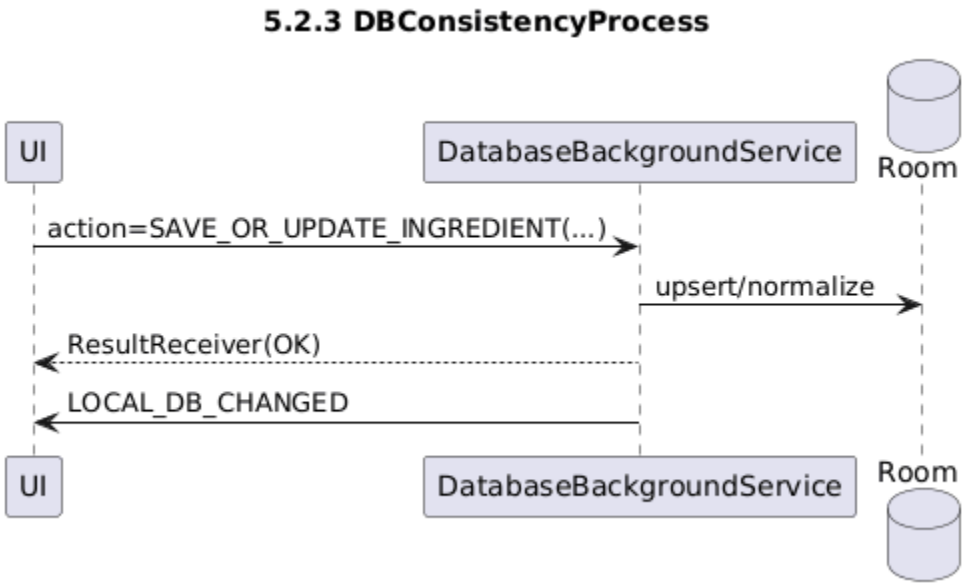
5.2.2. Process 2 Interface — QuotesPeriodicProcess

Field	Description
Trigger	WorkManager schedule
Input	—
Output	Notification with quote; Result.success()
Contracts	Retrofit GET /randomQuote; NotificationUtils



5.2.3. Process 3 Interface — DBConsistencyProcess

Field	Description
Trigger	UI Intent actions
Input	Action-specific (e.g., recipe, ingredient)
Output	ResultReceiver Bundle; broadcasts for state update
Contracts	Room DAOs + UnitConverter; LocalBroadcastManager



6. Detailed Design

6.1. Modules detailed design

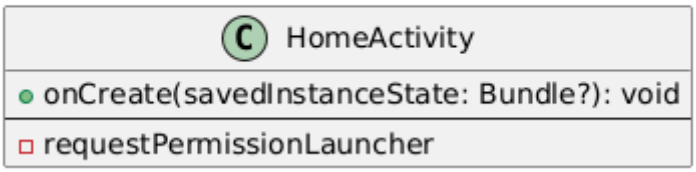
6.1.1. Module 1 — MobileApp (Android/Kotlin)

Field	Description
Name	MobileApp (Android/Kotlin)
Type	Code module
Purpose	Provide UI, orchestrate background work, and persist data locally
Way of operating	Activities/Fragments + ViewModels; Services; Room; Worker
Classes	HomeActivity; GenerateRecipeService; DatabaseBackgroundService; RandomQuoteWorker; SharedPreferencesManager; RecipeDao; IngredientDao; RecipeEntity; IngredientEntity

6.1.1.1. Module 1, class 1 — HomeActivity

Field	Description
Name	HomeActivity
Purpose	Entry activity; sets content view, requests POST_NOTIFICATIONS permission, wires NavController & BottomNavigationView, creates notification channel
Members	requestPermissionLauncher
Methods	onCreate(savedInstanceState: Bundle?)

6.1.1.1 — Module 1, Class 1 — HomeActivity



6.1.1.1.1. Module 1, class 1, method 1 — onCreate

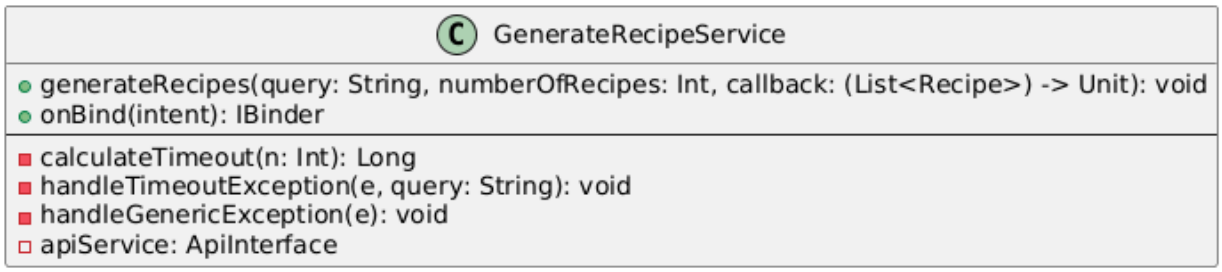
Field	Description
Name	onCreate
Purpose	Initialize UI, permissions, navigation, and notification channel
Prototype	fun onCreate(savedInstanceState: Bundle?)
Input	savedInstanceState: Bundle?
Output	—

Caller	Android framework
Calls	findNavController(), setupWithNavController(), NotificationUtils.createNotificationChannel(this), requestPermissionLauncher.launch(...)
Algorithm	Set content view; request POST_NOTIFICATIONS (API 33+); set up bottom navigation with NavController; create notification channel

6.1.1.2. Module 1, class 2 — GenerateRecipeService

Field	Description
Name	GenerateRecipeService
Purpose	Perform recipe generation with dynamic timeout and error notifications
Members	apiService: ApiInterface? ; binder: LocalBinder
Methods	generateRecipes(query, numberOfRecipes, callback); onBind(intent); handleTimeoutException(); handleGenericException(); calculateTimeout(n)

6.1.1.2 — Module 1, Class 2 — GenerateRecipeService



6.1.1.2.1. Module 1, class 2, method 1 — generateRecipes

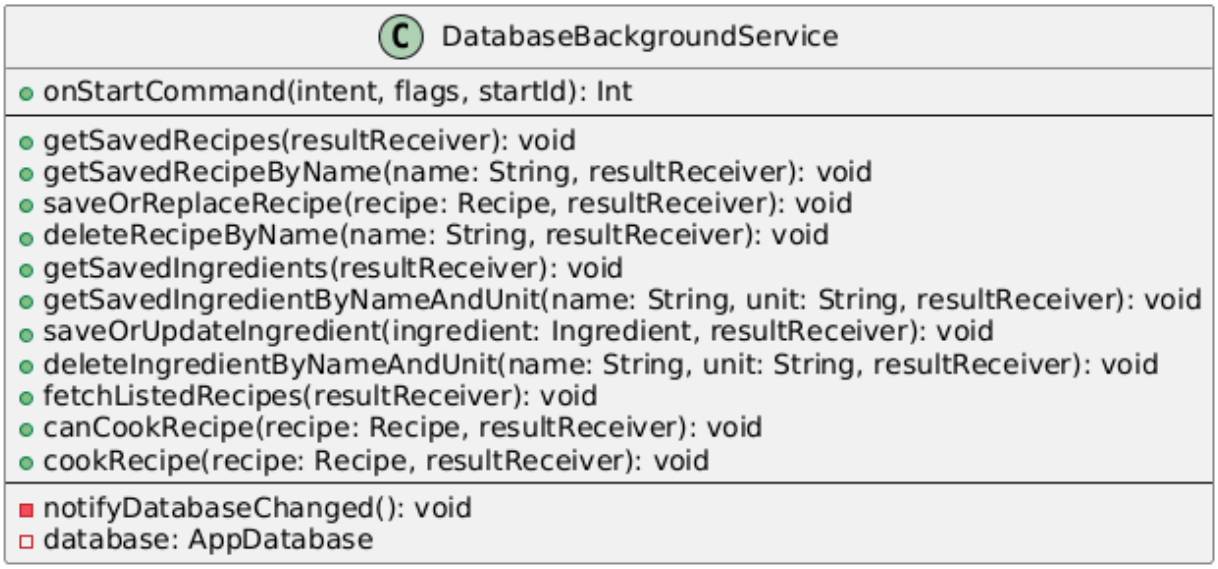
Field	Description
Name	generateRecipes
Purpose	Issue the network request and deliver parsed recipes to the UI
Prototype	fun generateRecipes(query: String, numberOfRecipes: Int, callback: (List<Recipe>) -> Unit)
Input	query: String ; numberOfRecipes: Int
Output	Invokes callback(List<Recipe>) on main thread
Caller	Home/UI via binding
Calls	ApiClient.getInstance(timeout).create(ApiInterface), ApiInterface.searchRecipes(...).execute()
Algorithm	Compute timeout (5s + 10s/recipe, cap 120s);

	execute synchronous call on IO; handle SocketTimeoutException and generic exceptions by showing notifications; callback with response body or empty list

6.1.1.3. Module 1, class 3 — DatabaseBackgroundService

Field	Description
Name	DatabaseBackgroundService
Purpose	Encapsulate Room operations and domain rules behind Intent actions
Members	database: AppDatabase
Methods	onCreate(); onStartCommand(...); getSavedRecipes(); getSavedRecipeByName(); saveOrReplaceRecipe(); deleteRecipeByName(); getSavedIngredients(); fetchListedRecipes(); canCookRecipe(); cookRecipe(); getSavedIngredientByNameAndUnit(); saveOrUpdateIngredient(); deleteIngredientByNameAndUnit(); notifyDatabaseChanged()

6.1.1.3 — Module 1, Class 3 — DatabaseBackgroundService



6.1.1.3.1. Module 1, class 3, method 1 — canCookRecipe

Field	Description
Name	canCookRecipe
Purpose	Determine if all required ingredients are available (1% tolerance)
Prototype	private fun canCookRecipe(recipe: Recipe, resultReceiver: ResultReceiver?)
Input	recipe: Recipe
Output	Bundle { canCook:Boolean; missing_ingredients?: ArrayList<Ingredient> } via ResultReceiver
Caller	onStartCommand(action="CAN_COOK_RECIPE")
Calls	ingredientDao.getAllIngredientsSync(), UnitConverter.convert(...)
Algorithm	For each required ingredient: find saved item; convert required amount to saved unit; if required/saved < 0.99, add to missing list; return canCook flag accordingly

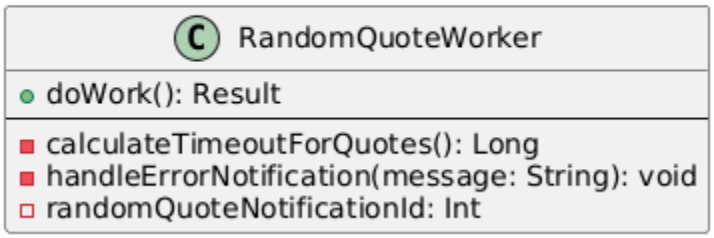
6.1.1.3.2. Module 1, class 3, method 2 — cookRecipe

Field	Description
Name	cookRecipe
Purpose	Subtract consumed ingredients from inventory, normalize units; delete when zero
Prototype	private fun cookRecipe(recipe: Recipe, resultReceiver: ResultReceiver?)
Input	recipe: Recipe
Output	Bundle { success:Boolean } via ResultReceiver; broadcast LOCAL_DB_CHANGED
Caller	onStartCommand(action="COOK_RECIPE")
Calls	ingredientDao.getAllIngredientsSync(), UnitConverter.convert(...), UnitConverter.getOptimalUnit(...), ingredientDao.updateIngredient(...), ingredientDao.deleteIngredientByNameAndUnit(...)
Algorithm	For each ingredient: convert to saved unit; subtract; if >0 → normalize to optimal unit and update; else delete row; broadcast DB change

6.1.1.4. Module 1, class 4 — RandomQuoteWorker

Field	Description
Name	RandomQuoteWorker
Purpose	Fetch a random quote and notify user
Members	randomQuoteNotificationId: Int
Methods	doWork(), calculateTimeoutForQuotes(), handleErrorNotification(message)

6.1.1.4 — Module 1, Class 4 — RandomQuoteWorker



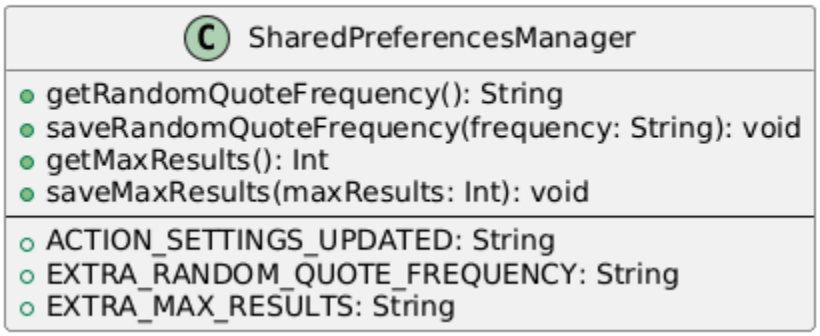
6.1.1.4.1. Module 1, class 4, method 1 — doWork

Field	Description
Name	doWork
Purpose	Execute Retrofit call and display notification
Prototype	override fun doWork(): Result
Input	—
Output	Result.success() after showing notification
Caller	WorkManager runtime
Calls	ApiClient.getInstance(timeout).create(ApiInterface).randomQuote().execute(), NotificationUtils.showStandardNotification(...)
Algorithm	Create client with ~10s timeout; on success show quote; otherwise show error notification; return success

6.1.1.5. Module 1, class 5 — SharedPreferencesManager

Field	Description
Name	SharedPreferencesManager
Purpose	Store settings and broadcast updates
Members	SharedPreferences; constants for keys and ACTION/EXTRAS
Methods	getRandomQuoteFrequency(), saveRandomQuoteFrequency(), getMaxResults(), saveMaxResults(), broadcastSettingsUpdate()

6.1.1.5 — Module 1, Class 5 — SharedPreferencesManager



6.1.2. Module 2 — BackendAPI (Node/Express)

Field	Description
Name	BackendAPI (Node/Express)
Type	Process / Code module
Purpose	Serve GET /getRecipes and GET /randomQuote; call OpenAI; return JSON
Way of operating	Express route handlers; JSON responses; uses environment variables for secrets
Classes	Route handlers; schemas module

6.1.2.1. Module 2, class 1 — GET /getRecipes handler

Field	Description
Name	GET /getRecipes
Purpose	Produce normalized list of recipes from a prompt/query using OpenAI
Members	— (route-local variables)
Methods	handler(req,res)

6.1.2.1.1. Module 2, class 1, method 1 — handler(req,res)

Field	Description
Name	handler(req,res)
Purpose	Validate, call OpenAI, map to schema, send JSON
Prototype	(req: Request, res: Response) => void
Input	Query params: numberOfRecipes:int, recipeQuery:string
Output	JSON: { recipes: [...] }
Caller	HTTP client (Mobile app)
Calls	OpenAI client (chat completions/responses)

Algorithm	Read params; build prompt; call OpenAI; validate/normalize; send JSON or error response

6.1.2.2. Module 2, class 2 — GET /randomQuote handler

Field	Description
Name	GET /randomQuote
Purpose	Call OpenAI with quote schema and return {quote}
Members	—
Methods	handler(req,res)

6.1.2.2.1. Module 2, class 2, method 1 — handler(req,res)

Field	Description
Name	handler(req,res)
Purpose	Call OpenAI with quote schema and return {quote}
Prototype	(req: Request, res: Response) => void
Input	—
Output	JSON: { quote: string }
Caller	HTTP client (Worker)
Calls	OpenAI client
Algorithm	Build prompt for concise quote; call OpenAI; send JSON or error

6.2. Data Modules Detailed Design

6.2.1. Data module 1 — RecipeEntity

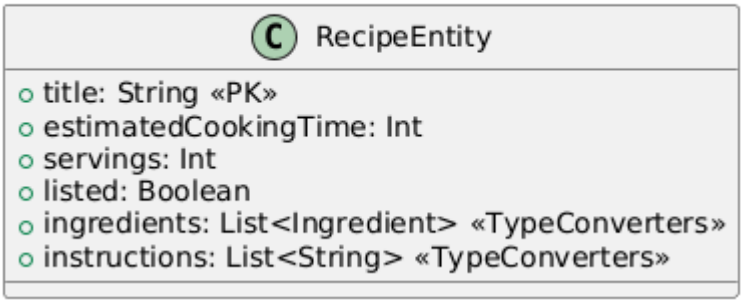
Field	Description
Name	RecipeEntity
Type	Room Entity (table 'recipes')
Purpose	Persist generated/saved recipes and their listed state.
Way of operating	Used by RecipeDao for CRUD; mapped to/from model Recipe; lists persisted with Gson TypeConverters.
Subordination	AppDatabase.entities; accessed by DatabaseBackgroundService via RecipeDao
Dependencies	Room; GsonConverters; models.Recipe; models.Ingredient

Resources	SQLite (Room)

Attribute	Kotlin Type	Nullability	Default	Notes
title	String	Non-null	—	PRIMARY KEY; unique
estimatedCookingTime	Int	Non-null	0	Minutes
servings	Int	Non-null	1	Suggested servings
listed	Boolean	Non-null	false	Shopping-list marker
ingredients	List<Ingredient>	Non-null	[]	Serialized via Gson TypeConverter
instructions	List<String>	Non-null	[]	Serialized via Gson TypeConverter

Constraint / Invariant	Description
Title uniqueness	No duplicate 'title' values.
Serializable lists	ingredients and instructions must serialize/deserialize via GsonConverters.
Domain completeness	At least one instruction recommended for UX integrity.

6.2.1 RecipeEntity



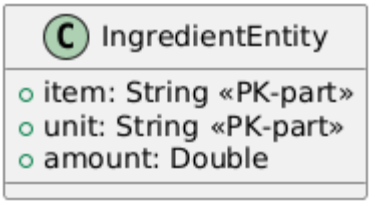
6.2.2. Data module 2 — IngredientEntity

Field	Description
Name	IngredientEntity
Type	Room Entity (table 'ingredients', composite PK: item+unit)
Purpose	Persist inventory items by name+unit with amounts for cooking subtraction and shopping.
Way of operating	Used by IngredientDao for CRUD; model mapping keeps UI decoupled; unit conversions happen at service layer (UnitConverter).
Subordination	AppDatabase.entities; accessed by DatabaseBackgroundService via IngredientDao
Dependencies	Room; models.Ingredient; UnitConverter
Resources	SQLite (Room)

Attribute	Kotlin Type	Nullability	Default	Notes
item	String	Non-null	—	Composite PK (part); lowercased on insert
unit	String	Non-null	—	Composite PK (part); UoM label
amount	Double	Non-null	0.0	Quantity in unit; must remain ≥ 0

Constraint / Invariant	Description
Composite key	(item, unit) pair uniquely identifies a row.
Non-negative amount	Zero triggers deletion during 'cook' operations; negatives are invalid.
Unit normalization	Service layer may convert to optimal unit prior to persistence.

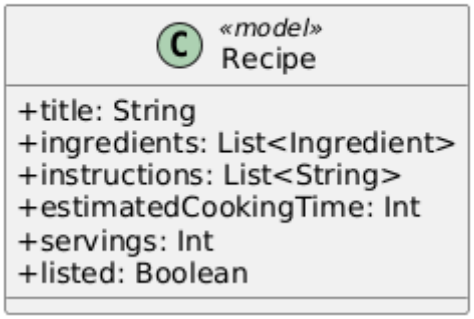
6.2.2 IngredientEntity



6.2.3 Data module 3 — Recipe (model)

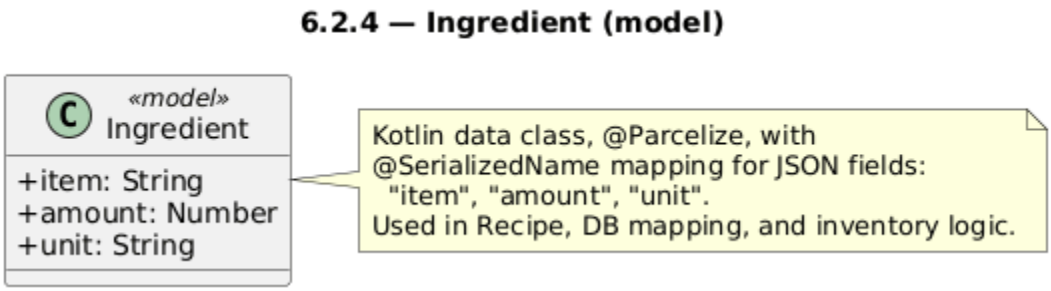
Field	Description
Name	Recipe (model)
Type	Domain model (in-memory)
Purpose	Transport and UI/domain manipulation of data in app layer.
Way of operating	Mapped to/from RecipeEntity using helper functions; transported via Intents/ResultReceiver bundles if needed.
Subordination	Used by UI, services, and DAOs (via mapping).
Dependencies	Kotlin stdlib; mapping helpers; related entity.
Resources	RAM (in-memory)

6.2.3 — Recipe (model)



6.2.4 Data module 4 — Ingredient (model)

Field	Description
Name	Ingredient (model)
Type	Domain model (in-memory)
Purpose	Transport and UI/domain manipulation of data in app layer.
Way of operating	Mapped to/from IngredientEntity using helper functions; transported via Intents/ResultReceiver bundles if needed.
Subordination	Used by UI, services, and DAOs (via mapping).
Dependencies	Kotlin stdlib; mapping helpers; related entity.
Resources	RAM (in-memory)



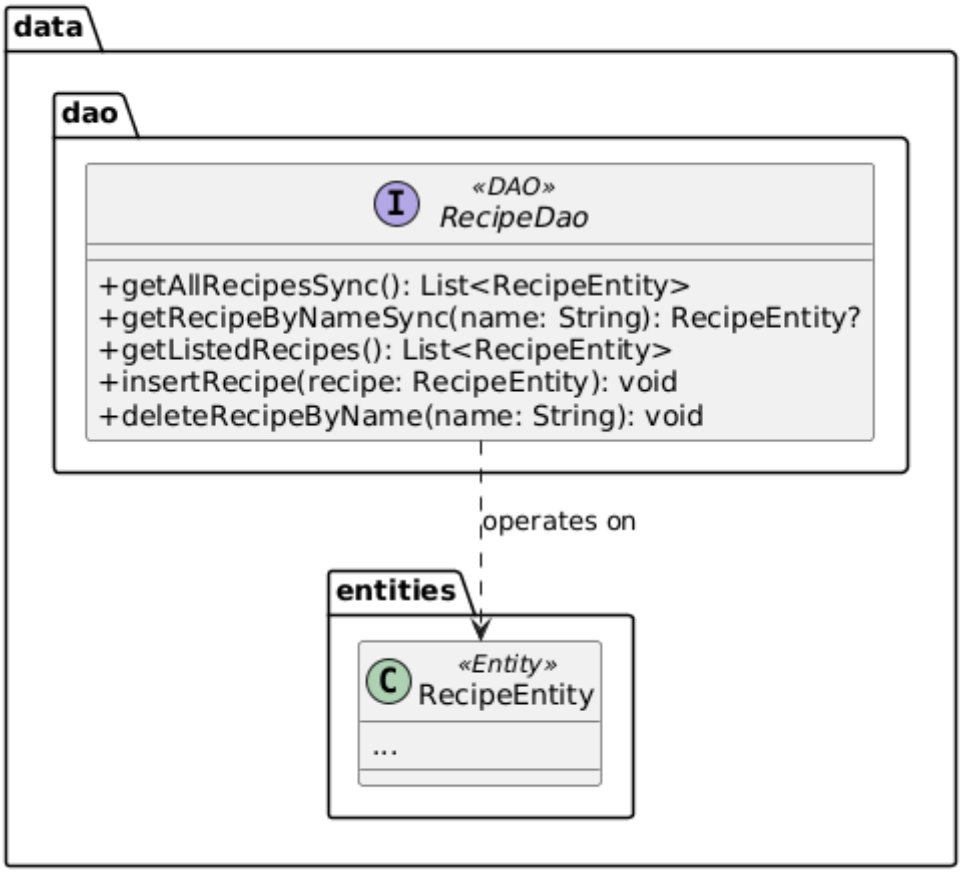
6.2.5 Data module 5 — RecipeDao (DAO)

Field	Description
Name	RecipeDao
Type	Data Access Object (Room DAO)
Purpose	CRUD access for RecipeEntity
Way of operating	Suspend functions executed on IO dispatcher; Room generates implementation.
Subordination	Exposed via AppDatabase; consumed by services.
Dependencies	Room; RecipeEntity
Resources	SQLite (Room)

Method	Annotation	SQL	Parameters	Returns	Purpose
getAllRecipesSync	Query	SELECT * FROM recipes	—	List<RecipeEntity>	Fetch all rows
getRecipeByNameSync	Query	SELECT * FROM	name: String	RecipeEntity?	Fetch rows by

		recipes WHERE title = :name			constraint
getListedRecipes	Query	SELECT * FROM recipes WHERE listed= 1	—	List<RecipeEntity>	Fetch rows by constraint

6.2.5 — RecipeDao (DAO)



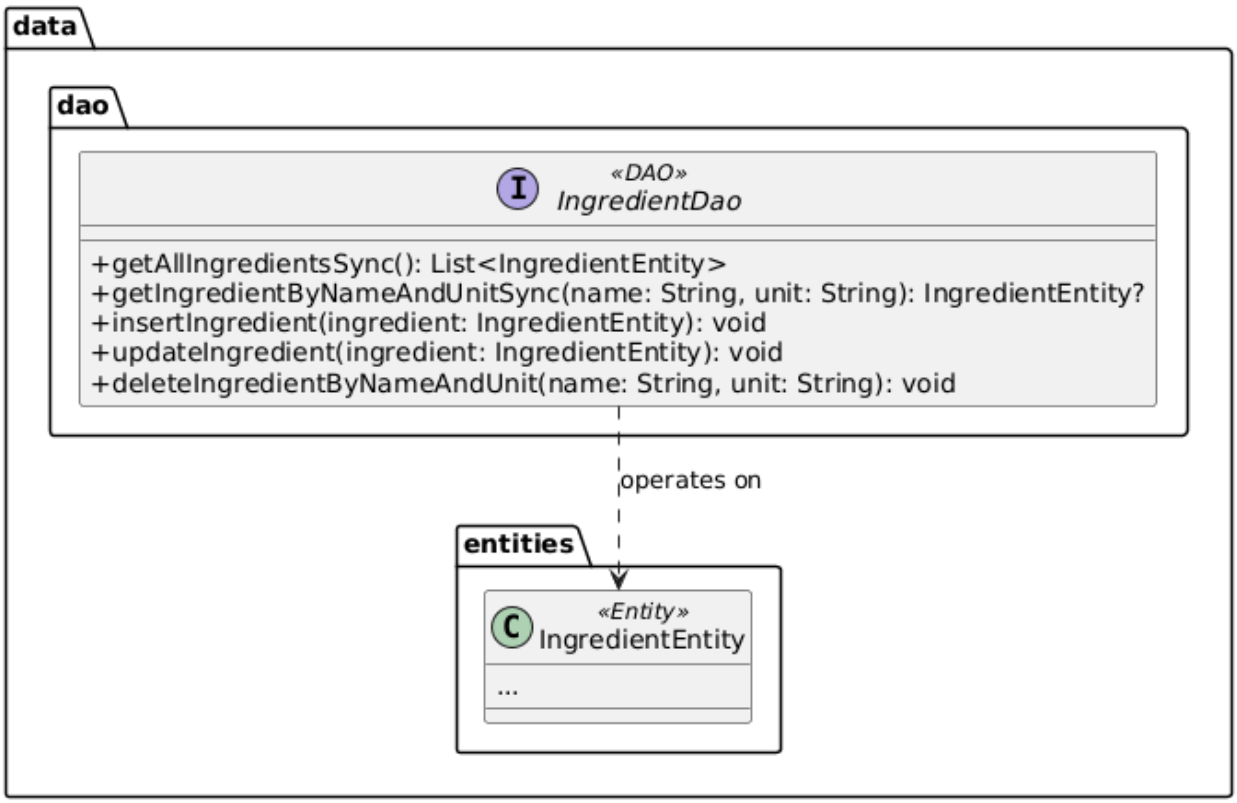
6.2.6 Data module 6 — IngredientDao (DAO)

Field	Description
Name	IngredientDao
Type	Data Access Object (Room DAO)
Purpose	CRUD access for IngredientEntity
Way of operating	Suspend functions executed on IO dispatcher; Room generates implementation.
Subordination	Exposed via AppDatabase; consumed by services.
Dependencies	Room; IngredientEntity
Resources	SQLite (Room)

Method	Annotation	SQL	Parameters	Returns	Purpose
getAllIngredientsSync	Query	SELECT * FROM ingredients	—	List<IngredientEntity>	Fetch all rows
getIngredientByNameAndUnitSync	Query	SELECT * FROM ingredients WHERE item = :name AND unit = :unit	name: String, unit: String	IngredientEntity?	Fetch rows by constraint

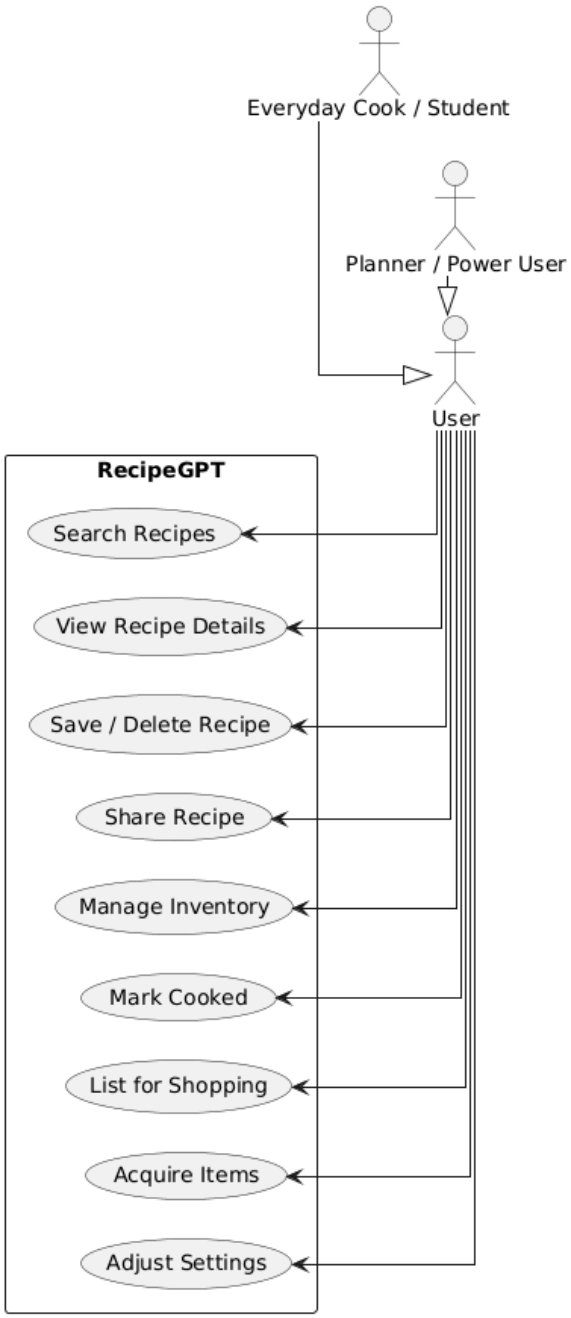
a

6.2.6 — IngredientDao (DAO)



Appendices

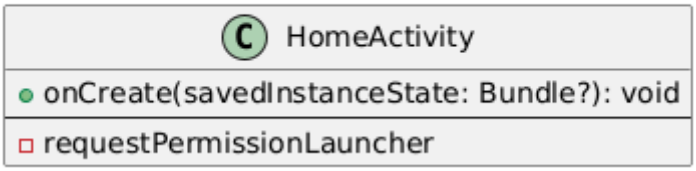
A1. Use cases diagrams



A2. Class diagrams

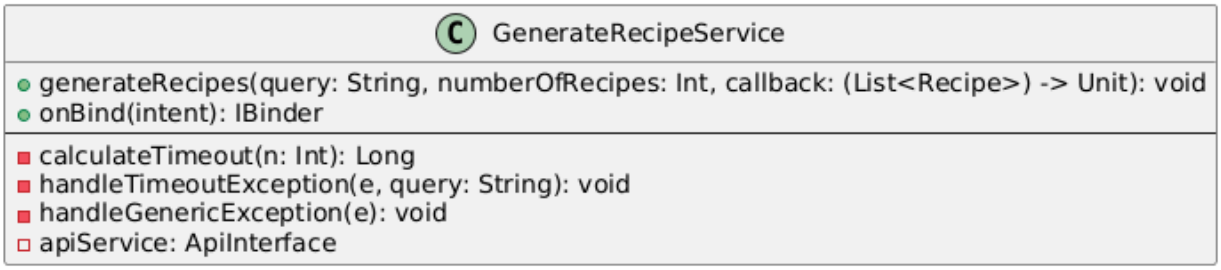
- Home Activity

6.1.1.1 — Module 1, Class 1 — HomeActivity



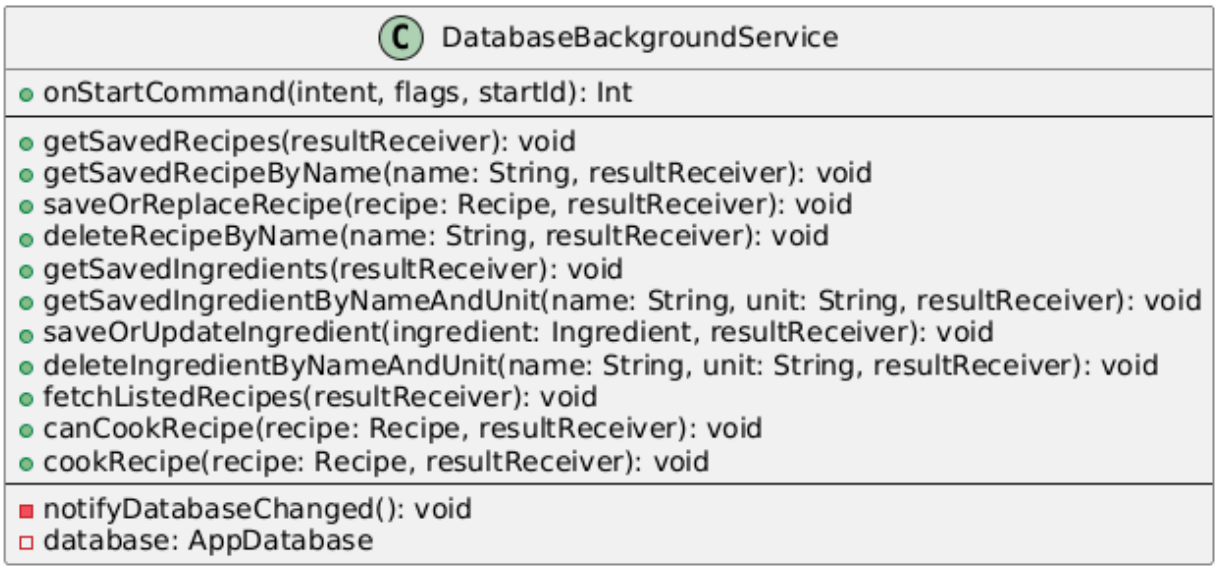
- Generate Recipe Service

6.1.1.2 — Module 1, Class 2 — GenerateRecipeService



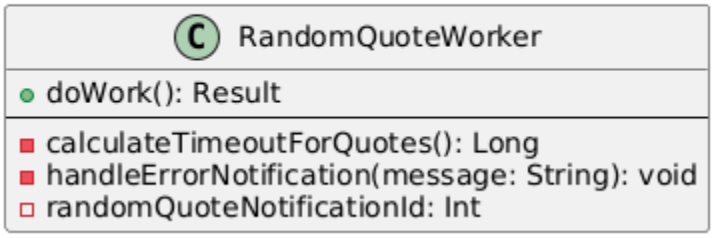
- DataBase Background Service

6.1.1.3 — Module 1, Class 3 — DatabaseBackgroundService



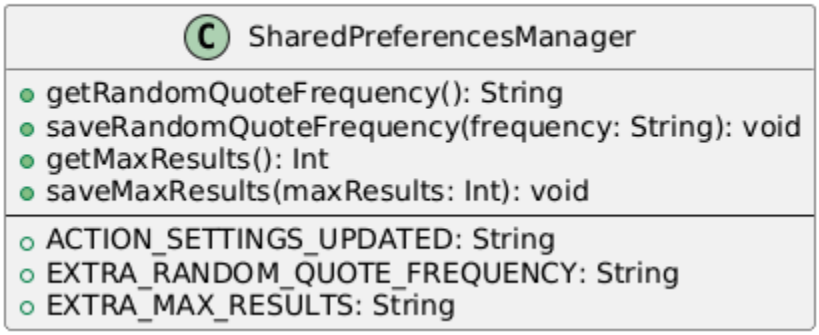
- Random Quote Worker

6.1.1.4 — Module 1, Class 4 — RandomQuoteWorker

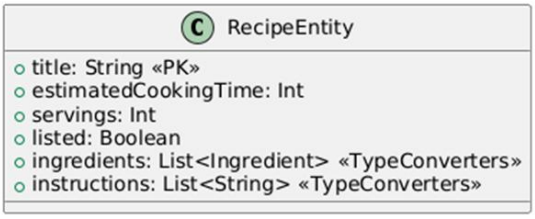


- Shared Preferences Manager

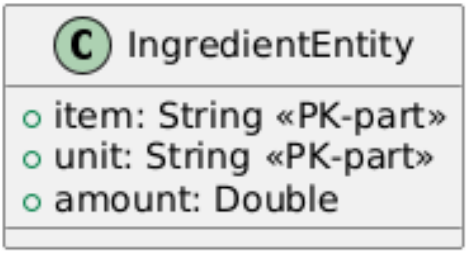
6.1.1.5 — Module 1, Class 5 — SharedPreferencesManager



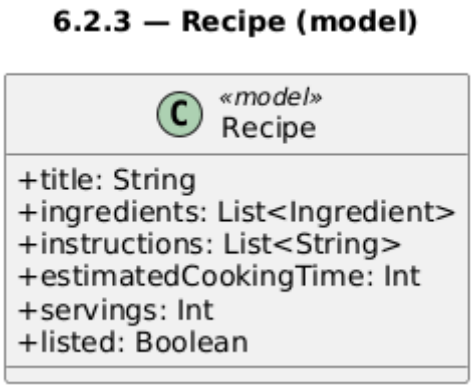
- Recipe Entity



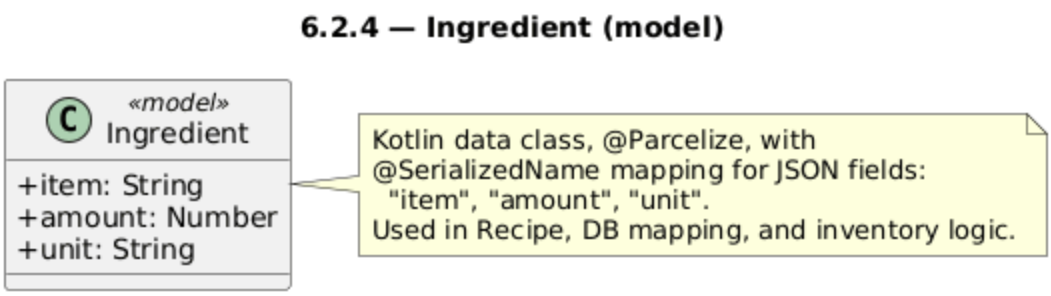
- Ingredient Entity



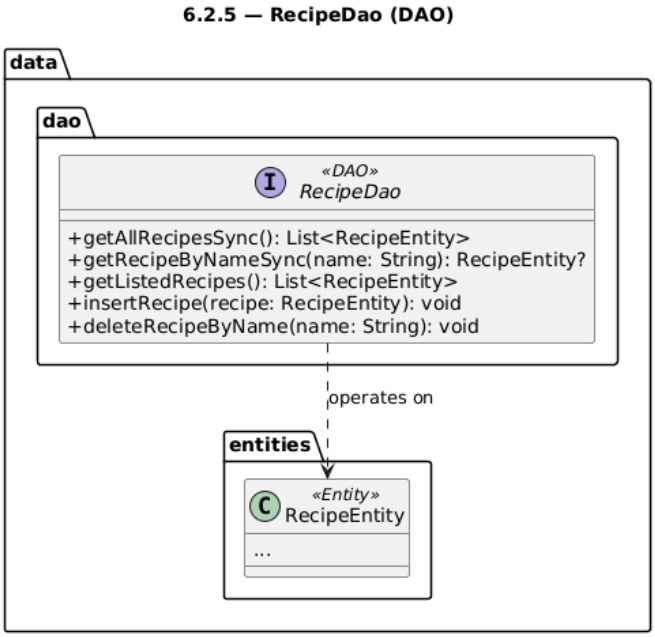
- Recipe (Model)



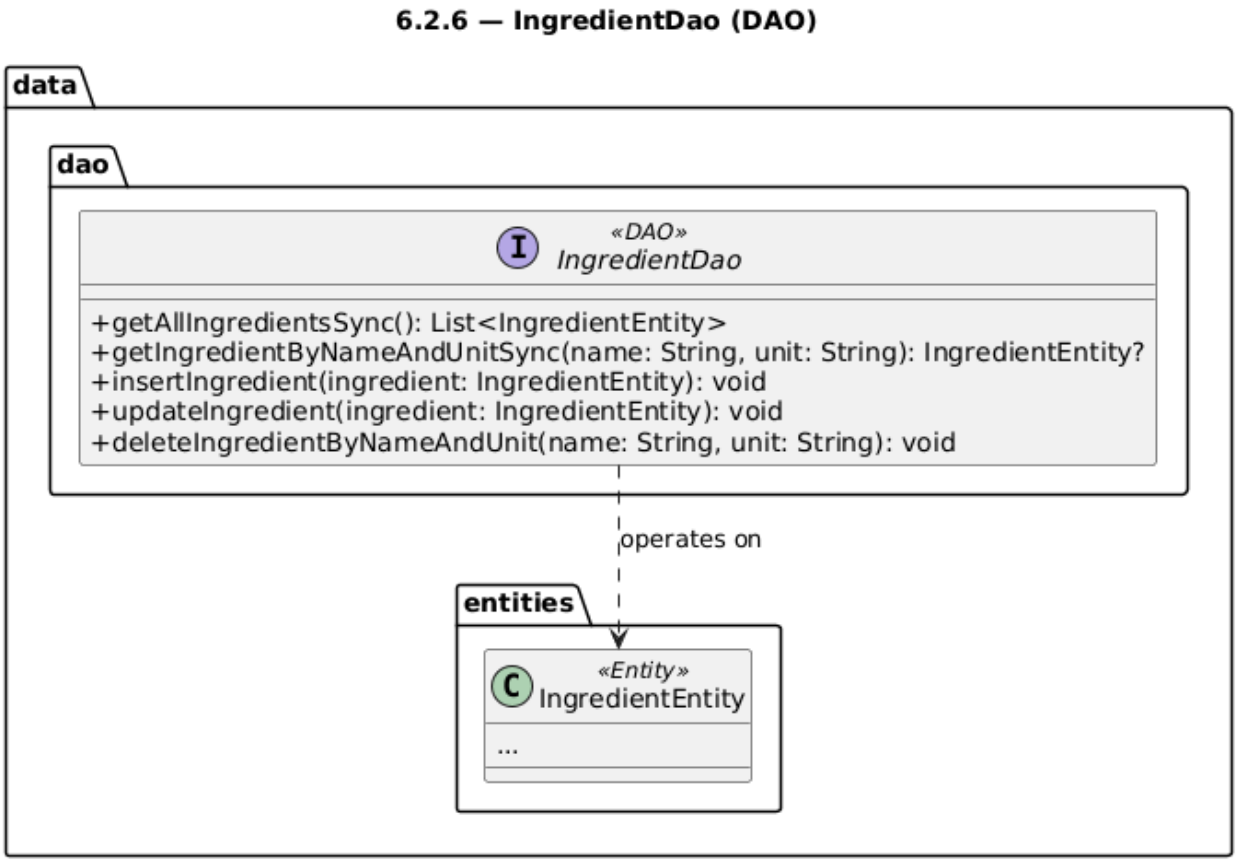
- Ingredient (Model)



- RecipeDAO

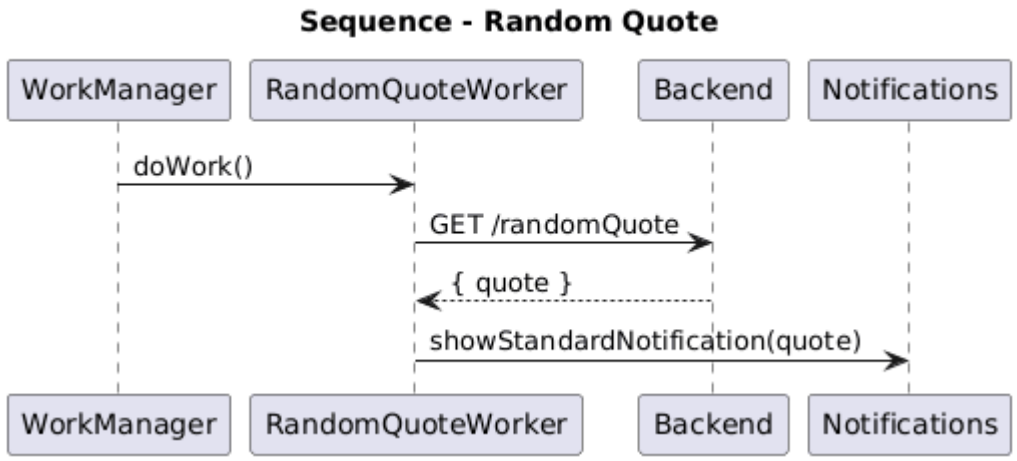


- Ingredient DAO

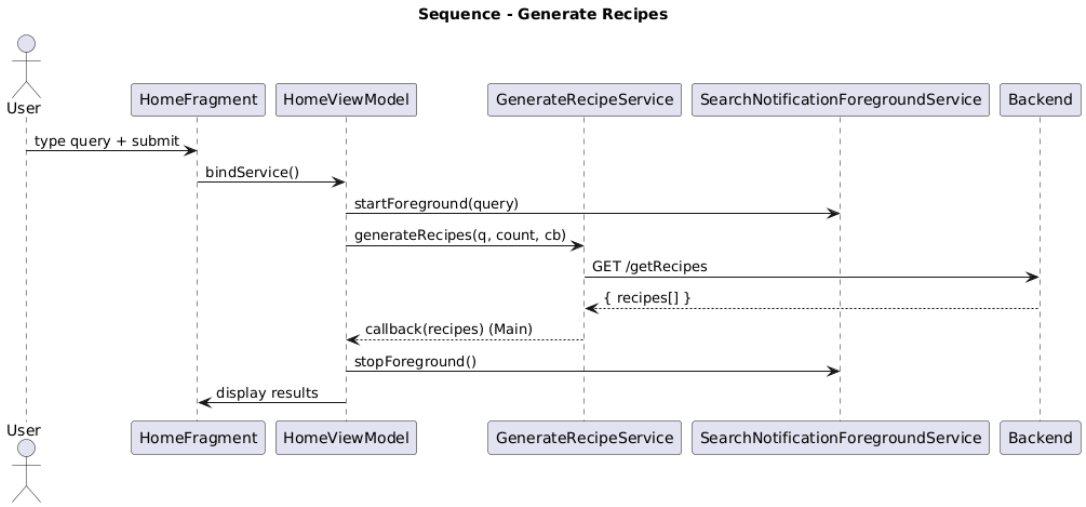


A3. Sequence diagrams

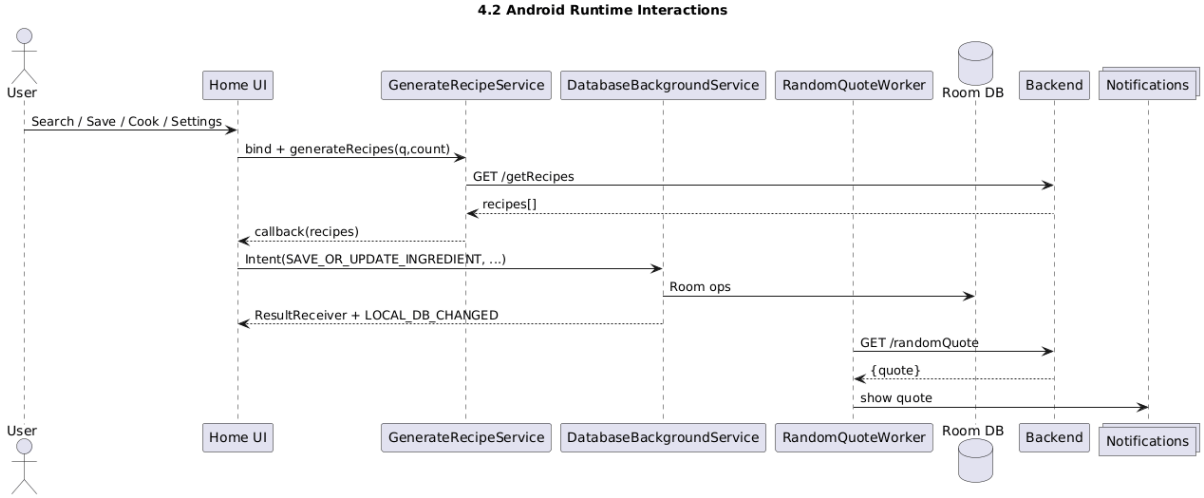
- Random Quote



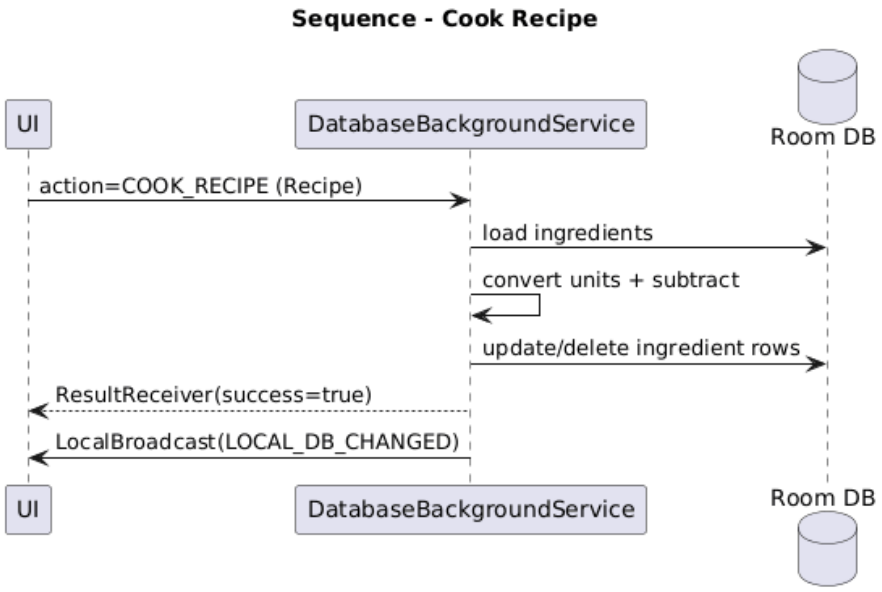
- Generate Recipes



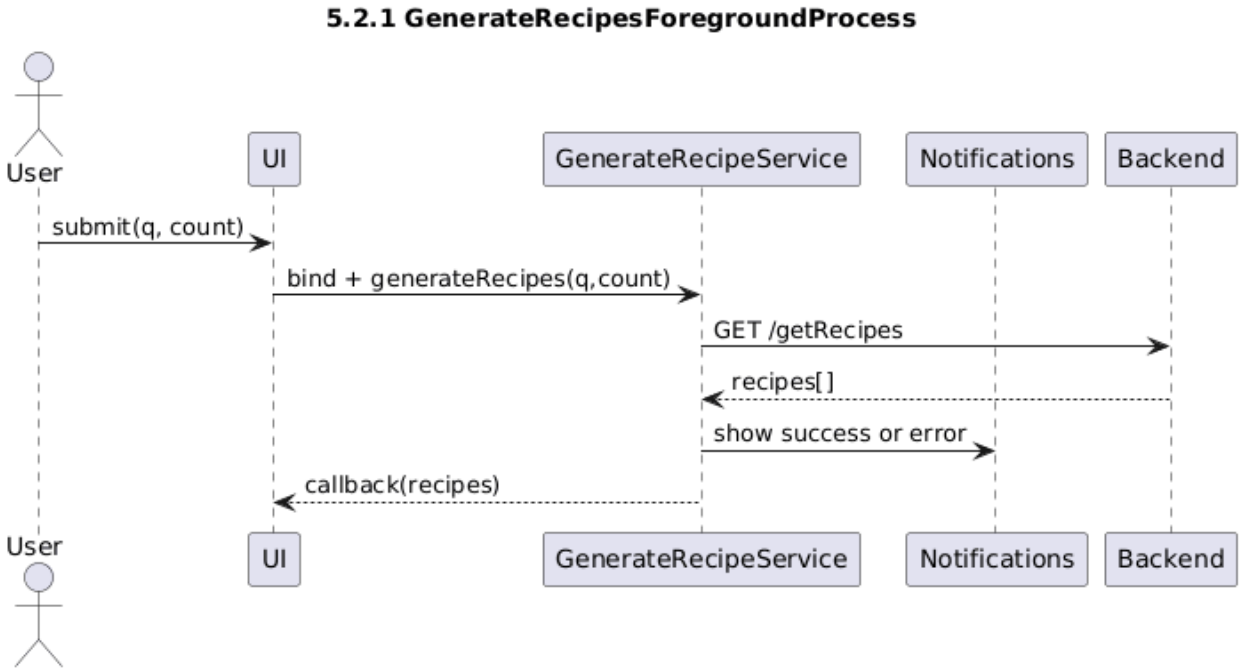
- Android Runtime Interactions



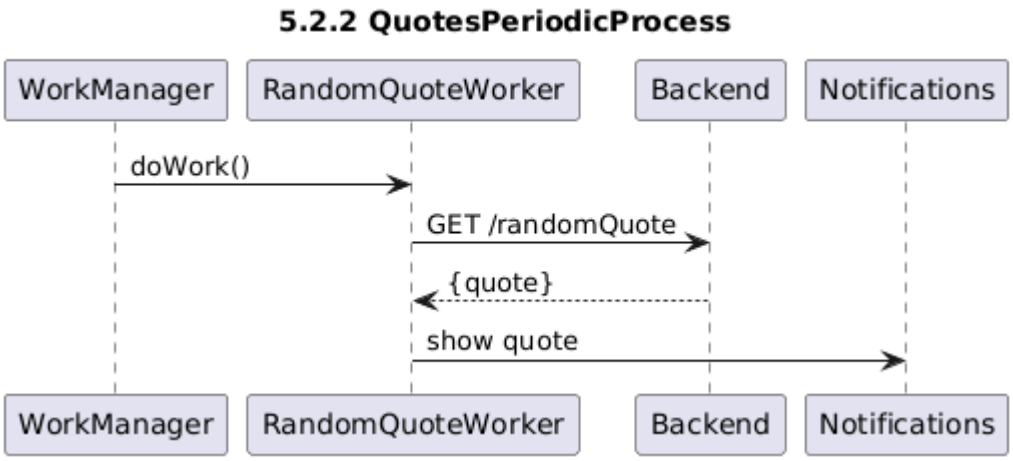
- Cook Recipe



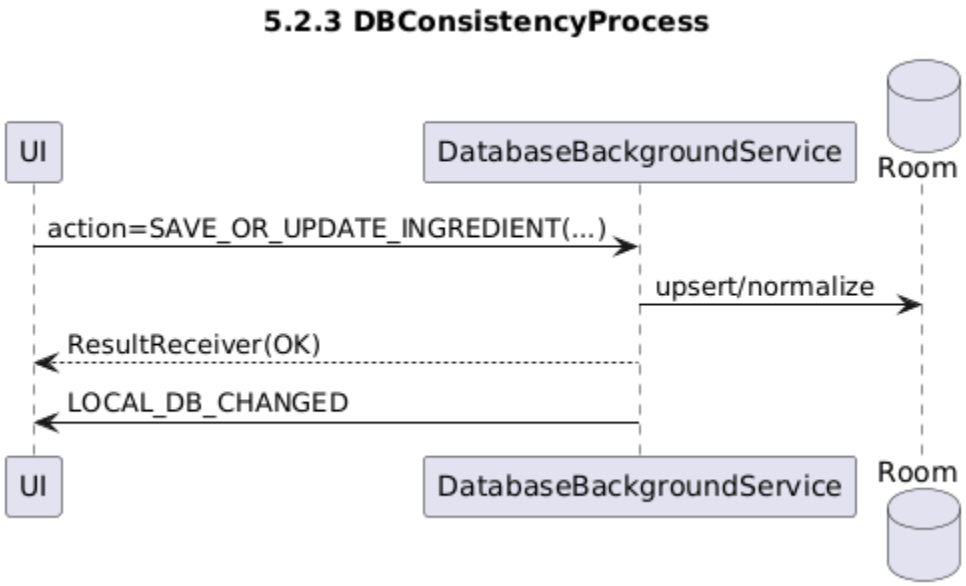
- Generate Recipes Foreground Process



- Quote Period Process



- DB Consistency Process



A4. Document evolution

A4.1 Initial design snapshot
Initial design matched the SRS: thin **Node/Express backend** exposing recipe and quote endpoints, and an **Android client** handling search, basic recipe storage, and a simple ingredient list. Room, WorkManager and notifications were mentioned but not fully structured into explicit entities, DAOs, or processes.

A4.2 Major changes list by date

- **2025-10** – Introduced RecipeEntity / IngredientEntity and RecipeDao / IngredientDao; added mappings from Recipe / Ingredient models.
- **2025-10** – Added DatabaseBackgroundService, GenerateRecipeService, and first version of RandomQuoteWorker for background work and notifications.
- **2025-11** – Formalized unit handling with QuantUnit and UnitConverter in canCookRecipe / cookRecipe; clarified inventory invariants.
- **2025-11** – Expanded dependency and interface sections; documented all IPC mechanisms (Intents, ResultReceiver, LocalBroadcastManager, notifications).
- **2025-11** – Updated SDD to follow IEEE 1016 more strictly, added PlantUML class/sequence diagrams, and converted descriptions to tables.

A4.3 Current state summary
Android app now has **clear modules, entities, models, and DAOs**, with Room-based persistence and background work via services + WorkManager. The backend remains **AI-only**, providing GET /getRecipes and GET /randomQuote with structured JSON. Constraints: internet required for generation/quotes; reliance on OpenAI and backend uptime. Next steps: strengthen tests, refine error/timeout handling, and add better filtering/pagination for recipes and ingredients.

A5. Conclusions regarding the activity

Architectural Summary
RecipeGPT uses a deliberately **thin, stateless backend** and a **feature-rich Android client**. The backend focuses on validating requests, calling the OpenAI API, and returning normalized JSON responses, while the mobile app handles domain logic, user interaction, and local data storage.

Qualities and Future Evolution
On the client side, **Room**, **WorkManager**, and a clear notification strategy ensure offline access, background processing, and continuous user feedback. This separation of concerns—lightweight server and robust client—improves performance, resilience, and maintainability, and leaves room for future extensions such as richer filters, new endpoints, or more advanced personalization features.