EduAI

AI-Powered Educational Learning Platform

Flutter | Firebase | Gemini AI

Version:	1.0.0
Platform:	Android, iOS, Web, Windows, macOS, Linux
Backend:	Firebase Auth + Cloud Firestore
AI:	Google Gemini API (gemini-2.5-flash-lite)
Type:	College / Demo Architecture
Date:	February 2026

PROJECT DOCUMENTATION

1. Project Overview

EduAI is an AI-powered educational mobile application built with Flutter and Firebase. It serves two distinct user roles: Students and Admins.

Students can access study materials, take quizzes, track their wellbeing, and interact with an AI chatbot powered by Google's Gemini API for academic help.

Admins can manage classes, subjects, study materials, quiz questions, view student performance, and configure system settings including the Gemini API key.

Feature Matrix

Feature	Student	Admin
Email registration & login	Yes	Yes
Subject-wise study materials (text + PDF)	Read	Full CRUD
Chapter-wise random quizzes (MCQ)	Attempt	Full CRUD
Quiz score history & progress	Own	View all
Daily mood tracking & wellbeing tips	Yes	--
Daily reflection journal	Yes	--
AI Chatbot (Gemini API)	Yes	--
Class & subject management	--	Yes
Student management & class assignment	--	Yes
System usage statistics dashboard	--	Yes
PDF upload via Cloudinary	--	Yes
Gemini API key management	--	Yes

2. Tech Stack & Dependencies

Core Framework

- Flutter (SDK ^3.9.2) - Cross-platform UI framework
- Dart - Programming language

Firebase Services

Package	Version	Purpose
firebase_core	^3.15.2	Firebase initialization
firebase_auth	^5.7.0	Email/password authentication
cloud_firestore	^5.6.12	NoSQL cloud database

AI & External APIs

Package	Version	Purpose
http	^1.5.0	HTTP client for Gemini API calls
file_picker	^10.3.2	File selection for PDF uploads

UI & PDF Libraries

Package	Version	Purpose
google_fonts	^6.2.1	Inter font family
syncfusion_flutter_pdfviewer	^29.2.11	In-app PDF viewing
syncfusion_flutter_pdf	^29.2.11	PDF text extraction for AI context
intl	^0.20.2	Date/time formatting
url_launcher	^6.3.2	External URL opening
path_provider	^2.1.5	File system path access

3. Project Structure

The project follows a clean, layered architecture with clear separation of concerns:

```
EduAi/
  lib/
    main.dart                    # Entry point
    app.dart                     # MaterialApp + auth gate
    firebase_options.dart        # Firebase config
    config/
      app_config.dart            # App constants
      app_theme.dart             # Theme + design widgets
      firebase_options_local.dart # Local Firebase config
    models/                      # 9 data model classes
      app_user.dart              # User profile
      chat_entry.dart            # Chat message pair
      mood_entry.dart            # Mood record
      quiz_attempt.dart          # Quiz result
      quiz_question.dart         # MCQ question
      reflection_entry.dart      # Journal entry
      school_class.dart          # Class (Std 6, 7, 8)
      study_material.dart        # Material (text + PDF)
      subject_model.dart         # Subject
    services/                    # 5 singleton services
      auth_service.dart          # Firebase Auth
      firestore_service.dart     # Firestore CRUD
      chat_api_service.dart      # Gemini API
      cloudinary_service.dart    # PDF upload
      wellbeing_service.dart     # Mood suggestions
    pages/
      auth/auth_page.dart        # Login / Signup
      shared/loading_page.dart   # Loading screen
      admin/                     # 7 admin screens
      student/                   # 8 student screens
```

4. Application Architecture

Architecture Pattern

The app follows a Service-Oriented Architecture with three layers: UI (Pages), Services (Singletons), and Data (Models), backed by external services.

Layer	Components	Responsibility
UI Layer	17 page widgets	User interface, navigation, state display
Service Layer	5 singleton services	Business logic, API calls, data access
Data Layer	9 model classes	Data structures, serialization
External	Firebase, Gemini, Cloudinary	Auth, database, AI, file hosting

Key Design Decisions

Decision	Implementation
State Management	StreamBuilder / FutureBuilder (no external library)
Navigation	Bottom NavigationBar with IndexedStack for tab persistence
Dependency Injection	Singleton pattern (ServiceName.instance)
Auth Flow	StreamBuilder on authStateChanges() -> role-based routing
File Storage	Cloudinary (not Firebase Storage) for PDFs

App Entry & Root Routing

main.dart initializes Firebase and runs EduAiApp. app.dart contains _RootGate with a 3-layer authentication gate:

- Layer 1: StreamBuilder<User?> - Firebase Auth state. No user -> AuthPage.
- Layer 2: FutureBuilder - ensureProfile() creates Firestore doc if missing.
- Layer 3: StreamBuilder<AppUser?> - Streams profile. Admin -> AdminShell, Student -> StudentShell.

5. Configuration

App Configuration (app_config.dart)

Config Key	Value / Default	Purpose
cloudinaryCloudName	dzzjiiwuy	Cloudinary account for PDF uploads
cloudinaryUnsignedPreset	eduai_pdf_unsigned	Unsigned upload preset
bootstrapAdminEmails	['admin@eduai.com']	Auto-admin emails at signup
moodCategories	Happy, Calm, Neutral, ...	Predefined mood options (6 total)

Gemini API Configuration

The Gemini API key is stored in Firestore at app_config/gemini (not hardcoded). Fields: apiKey (required), model (optional, default: gemini-2.5-flash-lite). Managed via Admin panel -> API Settings page.

6. Data Models

All 9 models in lib/models/ follow a consistent pattern: immutable final fields, toMap() for Firestore serialization, static fromDoc() factory from DocumentSnapshot, and Timestamp <-> DateTime conversion via _readDate() helper.

AppUser

File: lib/models/app_user.dart | Collection: users

Field	Type	Description
id	String	Firebase Auth UID (document ID)
name	String	Display name
email	String	Email address
role	UserRole	Enum: student or admin
classId	String?	Assigned class ID (students only)
createdAt	DateTime?	Account creation timestamp

Role is determined at signup by checking if email is in bootstrapAdminEmails. Includes copyWith() method for immutable updates.

SchoolClass

File: lib/models/school_class.dart | Collection: classes

Field	Type	Description
id	String	Document ID
name	String	Class name (e.g., Std 6, Std 7)
subjectIds	List<String>	IDs of assigned subjects
createdAt	DateTime?	Creation timestamp

SubjectModel

File: lib/models/subject_model.dart | Collection: subjects

Field	Type	Description
id	String	Document ID
name	String	Subject name (e.g., Mathematics)
createdAt	DateTime?	Creation timestamp

StudyMaterial

File: lib/models/study_material.dart | Collection: materials

Field	Type	Description
id	String	Document ID
classId	String	Linked class ID
subjectId	String	Linked subject ID

chapter	String	Chapter name/number
title	String	Material title
content	String	Text content body
pdfUrl	String?	Optional PDF URL (Cloudinary/external)
createdBy	String?	Admin UID who created it
createdAt	DateTime?	Creation timestamp
updatedAt	DateTime?	Last update timestamp

QuizQuestion

File: lib/models/quiz_question.dart | Collection: quiz_questions

Field	Type	Description
id	String	Document ID
classId	String	Linked class ID
subjectId	String	Linked subject ID
chapter	String	Chapter name
question	String	Question text
options	List<String>	4 answer options
correctIndex	int	Index of correct answer (0-3)
createdBy	String?	Admin UID who created it
createdAt	DateTime?	Creation timestamp

QuizAttempt

File: lib/models/quiz_attempt.dart | Collection: quiz_attempts

Field	Type	Description
id	String	Document ID
studentId	String	Student's UID
classId	String	Class ID
subjectId	String	Subject ID
chapter	String	Chapter name
totalQuestions	int	Total questions in the quiz
correctAnswers	int	Number of correct answers
scorePercent	double	Score as percentage
questionIds	List<String>	IDs of questions in this attempt
attemptedAt	DateTime?	Attempt timestamp

MoodEntry

File: lib/models/mood_entry.dart | Collection: mood_entries

Field	Type	Description
id	String	Document ID
studentId	String	Student's UID
dateLabel	String	Formatted date string
mood	String	Selected mood category

suggestion	String	Wellbeing suggestion for this mood
createdAt	DateTime?	Entry timestamp

ReflectionEntry

File: lib/models/reflection_entry.dart | Collection: reflections

Field	Type	Description
id	String	Document ID
studentId	String	Student's UID
dateLabel	String	Formatted date string
text	String	Reflection journal text
createdAt	DateTime?	Entry timestamp

ChatEntry

File: lib/models/chat_entry.dart | Collection: chats

Field	Type	Description
id	String	Document ID
studentId	String	Student's UID
userMessage	String	User's message
aiReply	String	AI's response
createdAt	DateTime?	Message timestamp

7. Firestore Database Design

The app uses 10 Firestore collections. The entity relationships form a hierarchical structure: Classes contain subjects, materials and quiz questions are linked to both a class and subject. Student activity records (quiz attempts, mood entries, reflections, chats) reference the student via studentId.

Collections Summary

Collection	Doc ID	Key Fields	Owner
users	Auth UID	name, email, role, classId	Self / Admin
classes	Auto	name, subjectIds[]	Admin
subjects	Auto	name	Admin
materials	Auto	classId, subjectId, chapter, title	Admin
quiz_questions	Auto	classId, subjectId, chapter, question	Admin
quiz_attempts	Auto	studentId, scorePercent	Student
mood_entries	Auto	studentId, mood, suggestion	Student
reflections	Auto	studentId, text	Student
chats	Auto	studentId, userMessage, aiReply	Student
app_config	Semantic	apiKey, model	Admin

8. Firestore Security Rules

File: firestore.rules - Role-based access control using three helper functions:

- signedIn() - Checks request.auth != null
- isSelf(uid) - Checks if request user matches document owner UID
- isAdmin() - Checks if user's Firestore role field is 'admin'

Access Control Matrix

Collection	Read	Create	Update	Delete
users	Self/Admin	Self	Admin	Admin
classes	Signed-in	Admin	Admin	Admin
subjects	Signed-in	Admin	Admin	Admin
app_config	Signed-in	Admin	Admin	Admin
materials	Signed-in	Admin	Admin	Admin
quiz_questions	Signed-in	Admin	Admin	Admin
quiz_attempts	Admin/Owner	Owner*	Admin	Admin
mood_entries	Admin/Owner	Owner*	NEVER	NEVER
reflections	Admin/Owner	Owner*	NEVER	NEVER
chats	Admin/Owner	Owner*	NEVER	Admin/Owner

** Owner create rules verify request.resource.data.studentId == request.auth.uid to prevent impersonation.*

9. Firestore Composite Indexes

File: firestore.indexes.json - Six composite indexes required for the app's queries:

Collection	Fields	Purpose
materials	classId ASC, subjectId ASC, chapter ASC	Materials filtered by class+subject
quiz_questions	classId ASC, subjectId ASC, chapter ASC	Questions filtered by class+subject
quiz_attempts	studentId ASC, attemptedAt DESC	Student's quiz history by date
mood_entries	studentId ASC, createdAt DESC	Student's mood history by date
reflections	studentId ASC, createdAt DESC	Student's reflections by date
chats	studentId ASC, createdAt DESC	Student's chat history by date

10. Services Layer

All services use the Singleton pattern: private constructor + static instance field. This provides global access without dependency injection frameworks.

AuthService

File: lib/services/auth_service.dart (94 lines) - Wraps Firebase Authentication.

Method	Returns	Description
authStateChanges()	Stream<User?>	Firebase Auth state stream
signIn(email, password)	Future<void>	Email/password sign in
signUp(name, email, password)	Future<void>	Create account + Firestore user doc
signOut()	Future<void>	Sign out current user
profileStream(uid)	Stream<AppUser?>	Real-time user profile stream
getProfile(uid)	Future<AppUser?>	One-shot profile fetch
ensureProfile(user)	Future<void>	Create profile doc if missing

Signup flow: Create Firebase Auth account -> Update display name -> Check bootstrapAdminEmails -> Create users/{uid} document with role.

FirestoreService

File: lib/services/firestore_service.dart (524 lines) - Central data access layer with CRUD for all 10 collections.

Class & Subject Methods

Method	Description
streamClasses()	Real-time ordered list of all classes
saveClass(id?, name, subjectIds)	Create or update a class
updateClassSubjects(classId, subjectIds)	Update subject assignments
deleteClass(classId)	Delete a class document
streamSubjects()	Real-time ordered list of all subjects
saveSubject(id?, name)	Create or update a subject
deleteSubject(subjectId)	Delete a subject document

Material Methods

Method	Description
streamMaterials(classId, subjectId)	Filtered materials stream
streamClassMaterials(classId)	All materials for a class
saveMaterial(...)	Create or update a material
deleteMaterial(materialId)	Delete a material

Quiz Methods

Method	Description
streamQuizQuestions(classId, subjectId, chapter?)	Filtered question stream
saveQuizQuestion(...)	Create or update a question
deleteQuizQuestion(questionId)	Delete a question
getRandomQuestions(classId, subjectId, count, ch?)	Fetch N random questions
saveQuizAttempt(attempt)	Save quiz result
streamStudentAttempts(studentId)	Student's quiz history
streamAllAttempts()	All quiz attempts (admin)

Wellbeing & Chat Methods

Method	Description
streamMoodEntries(studentId)	Student's mood history
saveMood(studentId, dateLabel, mood, suggestion)	Save mood entry
streamReflections(studentId)	Student's reflections
saveReflection(studentId, dateLabel, text)	Save reflection
streamChats(studentId)	Last 50 chat entries (real-time)
getRecentChats(studentId, limit)	Recent chats for AI context
saveChat(studentId, userMessage, aiReply)	Save chat pair
clearChats(studentId)	Batch-delete all chats (400/batch)

ChatApiService

File: lib/services/chat_api_service.dart (291 lines) - Google Gemini AI integration.

Configuration: API key and model loaded from Firestore app_config/gemini. Supports flexible field name matching (apiKey, api_key, apikey, geminiApiKey). Default model: gemini-2.5-flash-lite.

ask() Method - Two Modes

Parameter	Strict Mode	Normal Mode
answerOnlyFromMaterials	true	false
Behavior	Only answers from provided context	General tutor, any question
Temperature	0.2 (more focused)	0.35 (more creative)
Max Output Tokens	1400	900
System Instruction	Answer from context only	Helpful tutor for students

The method accepts conversation history (List<ChatEntry>), optional study material context text, and sends the full payload to the Gemini API. Errors are translated to user-friendly messages.

CloudinaryService

File: lib/services/cloudinary_service.dart (85 lines) - PDF upload to Cloudinary free tier.

- pickAndUploadPdf() - Opens file picker (PDF only), uploads, returns secure URL
- uploadPdf(fileName, filePath?, bytes?) - Low-level upload, supports byte data and file path
- Upload endpoint: https://api.cloudinary.com/v1_1/{cloudName}/raw/upload

WellbeingService

File: lib/services/wellbeing_service.dart (25 lines) - Mood-to-suggestion mapping.

Mood	Suggestion
Happy	Great energy today. Use it to finish one tough chapter.
Calm	Keep your rhythm. Do one revision session and one quiz.
Neutral	Try a 25-minute focus block with no phone distractions.
Stressed	Pause for 5 minutes of deep breathing, then start small.
Tired	Do light review now and attempt harder topics after rest.
Anxious	Write your top 3 worries and convert each into one action step.

11. UI Layer - Pages & Screens

Authentication

AuthPage (auth_page.dart, 300 lines)

A single page toggling between Login and Sign Up modes. Fields: _isLogin toggle, text controllers for name/email/password, loading state, error display. Login calls AuthService.signIn(), Signup calls AuthService.signUp().

Shared

LoadingPage (loading_page.dart)

Animated loading screen with CircularProgressIndicator and customizable label. Used during auth transitions.

Admin Module (7 Screens)

AdminShellPage (109 lines)

Bottom navigation shell with 5 tabs: Classes, Materials, Quiz, Students, Usage. Uses IndexedStack for tab persistence.

AdminClassesPage (~16 KB)

Manages classes (Std 6, 7, 8, etc.) and subjects. Create/edit/delete classes, create/delete subjects, assign subjects to classes via multi-select. Real-time StreamBuilder updates.

AdminMaterialsPage (~17 KB)

Study material CRUD. Filter by class and subject. Add material with title, chapter, content text, optional PDF URL. Includes Cloudinary PDF upload button. Edit/delete existing materials.

AdminQuizPage (~18 KB)

Quiz question bank management. Filter by class, subject, chapter. Add MCQ questions with 4 options and correct answer. Delete existing questions.

AdminStudentsPage (~11 KB)

List all students, view profiles, assign classes via dropdown, view quiz performance summary.

AdminUsagePage (~7.5 KB)

System usage dashboard with stat cards: total students, quiz attempts, chats, materials.

AdminApiSettingsPage (218 lines)

Gemini API key management: view (masked/unmasked), edit, save to Firestore, reload, force-refresh ChatApiService config cache.

Student Module (8 Screens)

StudentShellPage (109 lines)

Bottom navigation shell with 5 tabs: Home, Materials, Quiz, Mood, Chat. Uses IndexedStack.

StudentDashboardPage (377 lines)

Home dashboard: welcome message, assigned class display, subject list, quiz stats (total quizzes, average score, best score), recent quiz history.

StudentMaterialsPage (242 lines)

Subject browser showing subjects assigned to student's class. Tapping navigates to StudentSubjectDetailsPage.

StudentSubjectDetailsPage (~5.4 KB)

Material viewer per subject. Lists materials grouped by chapter. Displays text inline, PDF button for materials with pdfUrl.

StudentPdfViewerPage (~1.9 KB)

Full-screen in-app PDF viewing using Syncfusion PDF viewer widget.

StudentQuizPage (521 lines)

Three-state quiz system: (1) Setup - select subject/chapter, (2) In-Progress - answer MCQs one at a time, (3) Results - score with pass/fail badge (green >= 50%, red < 50%). Fetches random questions, saves QuizAttempt on submit.

StudentWellbeingPage (394 lines)

Two sections: (1) Mood Tracker - select daily mood from categories, get personalized suggestion, view history. (2) Reflection Journal - free-text daily entry, view past reflections.

StudentChatPage (947 lines) - Most Complex Screen

AI-powered chatbot with: real-time Gemini chat, two modes (Normal general tutor / Strict material-based), material selector for context, PDF text extraction via Syncfusion, conversation history for context-aware responses, batch chat clear, bubble UI with user/AI differentiation, auto-scroll, loading indicators, and error handling.

12. Theming & Design System

File: lib/config/app_theme.dart (373 lines) - Material 3 theme built with Google Fonts Inter.

Color Palette

Name	Hex Code	Usage
primaryBlue	#1565C0	Primary brand, buttons, active states
accentBlue	#42A5F5	Secondary accents, gradient endpoints
darkText	#1A1A2E	Primary text color
secondaryText	#5C6B8A	Subtitles, labels, hints
surfaceWhite	#FFFFFF	Card and surface backgrounds
backgroundTint	#F5F8FF	Subtle tinted backgrounds
borderLight	#E0E8F5	Card and input borders

Reusable Widget Builders

Method	Purpose
cardDecoration(radius)	White card with blue shadow
accentLeftBorder(radius)	Card with blue left accent border
tintedContainer(radius)	Light blue tinted background
sectionHeader(context, title, icon?)	Section title with optional icon
statCard(label, value, icon, ...)	Statistics display card
scoreBadge(percent)	Color-coded score badge (green/red)
chapterChip(text)	Blue chip for chapter labels

Themed Components

The Material 3 theme configures: AppBar, Card, InputDecoration, Buttons (Filled/Outlined/Text), NavigationBar, Chip, SnackBar, Dialog, Switch, Divider, ProgressIndicator, and DropdownMenu - all using the blue palette with consistent border radius (10-16px) and elevation settings.

13. User Flows

Student Registration & Login

- Student opens app -> AuthPage (login/signup toggle)
- Signup: AuthService.signUp() creates Firebase Auth account + Firestore user doc
- Root gate streams auth state -> streams profile -> routes to StudentShellPage

Student Takes a Quiz

- Step 1: Select subject and chapter on Quiz page
- Step 2: App fetches N random questions from Firestore
- Step 3: Student answers MCQs one at a time with navigation
- Step 4: Submit -> calculate score -> save QuizAttempt -> show results with badge

Student Uses AI Chat

- Step 1: Type message in chat input
- Step 2: Optionally select study materials for context (Strict mode)
- Step 3: App builds context (material text + extracted PDF text)
- Step 4: Sends prompt + history + context to Gemini API
- Step 5: Display AI reply in chat bubble, save to Firestore

Admin Manages Content

- Admin logs in -> Root gate routes to AdminShellPage (5 tabs)
- Classes tab: Add/edit/delete classes, create/delete subjects, assign subjects to classes
- Materials tab: Add materials with text + optional PDF (Cloudinary upload)
- Quiz tab: Add MCQ questions with 4 options and correct answer
- Students tab: View students, assign classes, check performance

14. Setup & Deployment Guide

Prerequisites

- Flutter SDK ^3.9.2 (includes Dart)
- Firebase account with a project
- Google Gemini API key (free tier available)
- Cloudinary account (optional, for PDF hosting)

Step-by-Step Setup

Step 1: Clone the repository

```
git clone <repository-url>
cd EduAi
```

Step 2: Create Firebase project

- Go to Firebase Console (https://console.firebase.google.com)
- Enable Email/Password Authentication in Auth > Sign-in method
- Create Cloud Firestore database

Step 3: Configure Firebase in the app

- Replace placeholders in lib/config/firebase_options_local.dart
- Alternatively: firebase login && flutterfire configure

Step 4: Deploy Firestore rules & indexes

```
firebase deploy --only firestore:rules
firebase deploy --only firestore:indexes
```

Step 5: Set app configuration

- In app_config.dart: set cloudinaryCloudName, cloudinaryUnsignedPreset, bootstrapAdminEmails
- In Firestore app_config/gemini: create doc with apiKey and model fields

Step 6: Install dependencies & run

```
flutter pub get
flutter run
```

Step 7: Create admin account

Sign up with an email listed in bootstrapAdminEmails. The account will auto-receive the admin role.

15. Important Notes & Limitations

Architecture Notes

Item	Detail
Demo architecture	Intentionally college/demo, not production-grade
No Firebase Storage	PDFs hosted on Cloudinary or external URLs
No state management lib	Uses built-in StreamBuilder/FutureBuilder
Singleton services	Simple but not unit-test friendly

Security Considerations

Issue	Detail
API key in Firestore	Readable by signed-in users; use Cloud Functions in production
Direct API calls	Gemini called from client; should proxy via server in production
Unsigned uploads	Cloudinary preset allows uploads without auth; fine for demo

Known Limitations

Limitation	Detail
No push notifications	Students not notified of new content
No offline support	Requires internet for all features
No image/media materials	Only text + PDF URL supported
No per-question tracking	Only total scores saved, not individual answers
Admin creation	Only via bootstrapAdminEmails before signup
Immutable mood/reflections	Cannot edit or delete once saved
Chat history limit	AI context: last 8 messages; stream: last 50

16. File Reference

File	Lines	Description
main.dart	28	Entry point
app.dart	79	Root widget + auth gate
config/app_config.dart	24	App constants
config/app_theme.dart	373	Theme + design widgets
models/app_user.dart	81	User model
models/chat_entry.dart	49	Chat model
models/mood_entry.dart	53	Mood model
models/quiz_attempt.dart	73	Quiz attempt model
models/quiz_question.dart	67	Quiz question model
models/reflection_entry.dart	49	Reflection model
models/school_class.dart	47	Class model
models/study_material.dart	69	Material model
models/subject_model.dart	37	Subject model
services/auth_service.dart	94	Auth service
services/firestore_service.dart	524	Firestore CRUD
services/chat_api_service.dart	291	Gemini API
services/cloudinary_service.dart	85	PDF uploads
services/wellbeing_service.dart	25	Mood suggestions
pages/auth/auth_page.dart	300	Login/signup
pages/shared/loading_page.dart	~80	Loading screen
pages/admin/ (7 files)	~1,600	Admin module
pages/student/ (8 files)	~3,000	Student module

Total: ~37 source files, ~6,200+ lines of Dart code.

Table of Contents