Faculty of Technology

BSc (Hons) in Computer Science

ReanMate

Ai Tutor And Private Classroom

Student Name	Yin Keo Odom, Thai Sokchea, Toek Pheakdey, Pou Chhiangheng
Lecturer/Professor	Prof. Sek Sokcheat
Academic Year - Term	Year2 Term5
Word Count	3017

Table of Content

Abstract

ReanMate is an AI-powered learning platform for people in Cambodia. The intended Minimum Viable Product (MVP) features teachers setting up private classes, inviting students to join online via a join code, uploading PDFs and videos to the class, enabling the creation of assignments and quizzes, and managing each class's student usage of AI assistance. Furthermore, students are able to enroll in authorized courses, view lessons created by the teacher, complete assignments and quizzes, and use AI tools to ask questions or get answers from lessons, quizzes, and assignments, all based on the teachers' uploaded content. This system is engineered to provide Khmer, English, or mixed-language questions, and to indicate the source(s) and the pages in which each AI answer is generated.

This project illustrates advanced object-oriented programming concepts, such as: classes, service classes, repository classes, validation, composition and separation of responsibility. It shows the structure of a procedural system which allows to decouple routes/controllers, business services, repositories, models, templates, and database access together. Users, classes, memberships, lessons, videos, documents, assignments, conversations, messages, sources, practice questions and AI usage records are stored in a MySQL DB. The AI workflow involves processing, chunking, embedding retrieval of the PDFs by page along with a grounded prompt that asks the OpenAI API or Gemini for an answer.

Introduction

Background

Digital learning systems is increasingly which combine course materials, assignments, communications, and personalized learning support. ReanMate proposes a focused learning environment in which teacher's own lesson material become the main knowledge source for AI tutor instead of threating AI as a general chatbot, this platform connects the AI experience directly to private class and teacher also provided PDF lessons.

Problems Statement:

- Students need additional explanations after class.
- PDF materials alone do not provide interactive learning support.
- General AI may provide answers outside the teacher's materials.
- Students need answers they can verify from the original lesson.
- Teachers need control over AI usage in their classes.
- Private class materials must be protected from unauthorized access.
- Khmer/English bilingual AI support is needed.
- Teachers need classes, lessons, PDFs, and assignments in one platform.
- AI usage must be controlled and monitored.

Solutions

ReanMate will provides private classes, teacher-managed PDF lessons, assignments, and a controlled AI tutor that answers questions based on authorized class materials.

Objectives:

- Implement secure login and role-based access.
- Manage lessons and PDF learning materials.
- Manage assignments and student progress.
- Build a PDF-grounded AI Tutor.
- Support Khmer and English learning.
- Provide AI answers with source-page references.
- Generate practice questions.
- Track and control AI usage.
- Apply secure OOP-based system architecture.

Research and Development Questions:

1. How can object-oriented design be used to separate learning-domain responsibilities from database and web concerns?
2. How can private class membership be enforced throughout normal requests and AI retrieval?
3. How can teacher-uploaded PDF material be transformed into searchable chunks while preserving page information?
4. How can a RAG workflow provide useful AI explanations while reducing unsupported answers?
5. How can the system support Khmer, English, and mixed-language questions?
6. How can AI usage be measured and limited during a small demonstration?

Scope

- Student, teacher, and administrator roles.
- Private classes with join codes.
- Class membership management.
- Teacher-created lessons and private PDF materials.
- PDF viewing and page navigation.
- Teacher-created and published assignments.
- Student assignment dashboard and status filtering.
- Class-scoped AI Tutor and Study Workspace.
- AI quick actions: Explain in Khmer, Summarize, Generate Practice Questions, and Quiz Me.
- Source document/page citation for grounded AI answers.
- Practice sets and questions.
- AI usage tracking and limits.
- Administrative monitoring of users, classes, documents, and AI usage.

Limited

- Live video classes.
- Native mobile application.
- Payments and subscriptions.

- School-wide administration.
- Public course marketplace.
- AI voice tutor.
- Advanced proctoring.
- Full LMS grading engine.
- Automatic high-stakes grading.
- Additional AI providers before the OpenAI-based MVP is reliable.

Target Users and Stakeholder

Role	Main Responsibilities
Student	Join private classes, read lessons, view assignments, ask the AI Tutor, practice, and monitor learning activity.
Teacher	Create classes, manage members, upload lessons, create assignments, and configure class AI behavior.
Admin	Manage users and classes, monitor document processing and AI usage, apply system controls, and review technical logs.
Lecturer / Evaluator	Review the R&D process, OOP design, implementation, testing, and final demonstration.

Chapter 2: Literature and Technology Review

The project uses object-oriented programming to represent important concepts such as User, Class, Lesson, Document, Assignment, Conversation, and AI Usage. The goal is not to create classes only for appearance. Each class should have a clear responsibility and meaningful methods. Services coordinate business workflows, while repositories isolate database operations.

Composition is preferred where one component depends on another, such as an AI service depending on a repository or usage tracker. Inheritance should only be used when the relationship is meaningful, such as role-specific behavior where it improves the design. Type hints, validation, and documentation should be used for public methods.

Python

Python is the main programming language for the course implementation. Its class system, type hints, modules, exception handling, testing ecosystem, and web frameworks make it suitable for demonstrating object-oriented design.

Flask

Flask is proposed as the web framework required by the Advanced OOP course. Flask routes/controllers receive requests, validate access, call service methods, and return HTML views or JSON responses. The route layer should not contain direct SQL queries or complex business rules.

PostgreSQL or Mysql

PostgreSQL is used as the primary relational database for the course implementation. Relational tables represent users, classes, memberships, lessons, documents, assignments, conversations, messages, and AI usage. Primary keys, foreign keys, unique constraints, and appropriate data types are used to maintain data integrity.

Retrieval-Augmented Generation

ReanMate uses a retrieval-augmented generation concept. Teacher PDFs are processed page by page, cleaned, divided into chunks, and represented for semantic retrieval. When a student asks a question, the application checks authentication, class membership, AI availability, and usage limits before retrieving relevant material. The retrieved context is then used to construct a grounded prompt for the AI provider.

A key requirement is that retrieval must remain inside the student's authorized class/document scope. The AI system should not retrieve chunks from another private class. When the material does not sufficiently answer a question, the tutor should state that the material is insufficient rather than inventing a source or page.

OpenAI API

The product specification selects the OpenAI API as the MVP AI provider. The API key must remain on the backend and must never be exposed to the browser. AI requests, token usage, and estimated cost should be recorded for monitoring.

Proposed System Features

This is a comprehensive list of proposed features for a Digital Learning Platform, structured by user role and core functionality.

Session Module:

- Register
- Login
- Logout
- Password management

Authorization Module:

- Student role
- Teacher role
- Admin role
- Role-based access control

User Management Module:

- User list
- CRUD
- Search
- User report

Class Management Module

- Class list
- CRUD
- Search
- Class report

Lesson & PDF Module

- Lesson list
- CRUD
- PDF upload/delete
- Search
- Processing status

Assignment Module

- Assignment list
- CRUD
- Search/filter
- Assignment report

AI Tutor Module

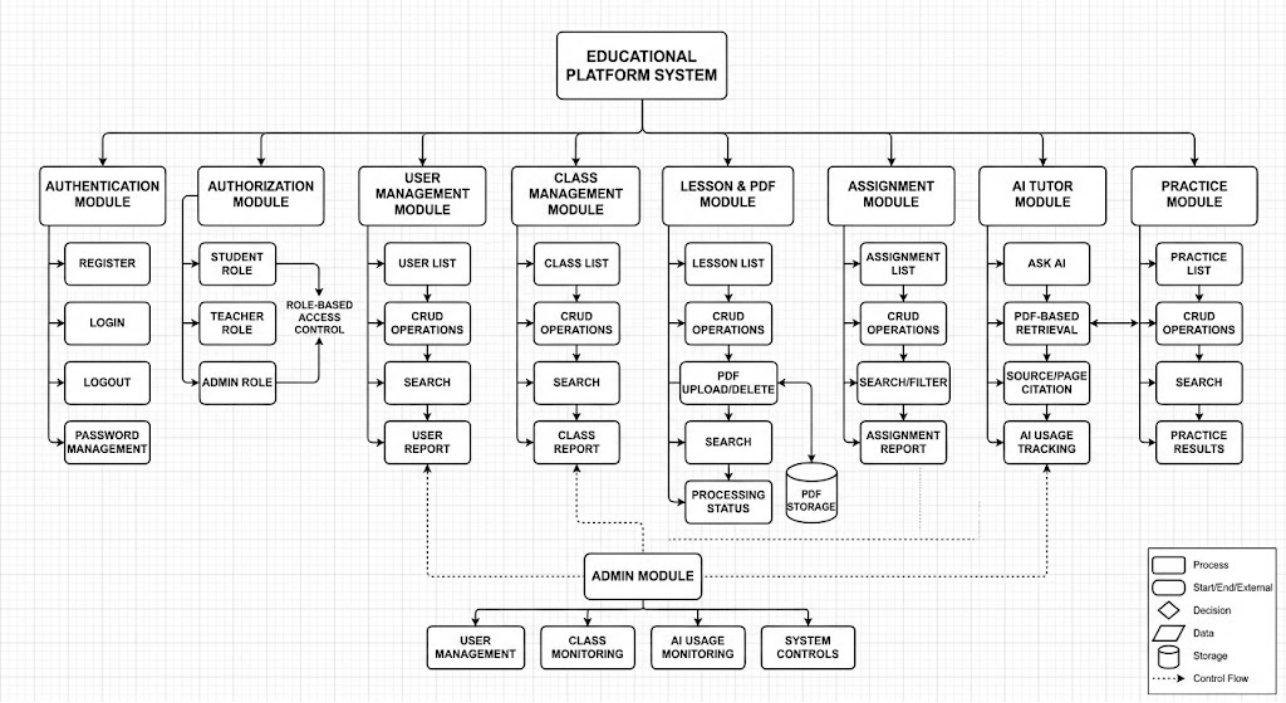
- Ask AI
- PDF-based retrieval
- Source/page citation
- AI usage tracking

Practice Module

- Practice list
- CRUD
- Search
- Practice results

Admin Module

- User management
- Class monitoring
- AI usage monitoring
- System controls



Object-Oriented Programming Class Design

Class	Responsibility	Example Methods
User	Represent account identity, role, and status.	has_role(), is_active()
Classroom	Represent a private learning class.	generate_join_code(), archive()
Class Member	Represent membership between a user and class.	can_access()
Lesson	Represent an ordered published/unpublished lesson.	publish(), unpublish()
Document	Represent an uploaded lesson PDF and processing state.	validate(), mark_ready()
Document Chunk	Represent page-aware searchable document content.	metadata()
Assignment	Represent teacher-created class work.	publish(), close(), is_overdue()
Assignment Progress	Track a student's assignment state.	complete(), submit()
Conversation	Represent a student's AI conversation.	add_message()
Message	Represent an AI/student message.	record_usage()
Message Source	Connect an AI message to a document page/chunk.	open_reference()
Practice Set	Represent saved practice generated from material.	add_question()
Practice Question	Represent one practice question.	check_answer()
AI Usage	Record request type, model, tokens, and estimated cost.	within_limit()

Service layer:

Service	Responsibility
AUTH Service	Registration, login, password verification, session-related business rules.
Class Service	Create classes, join codes, membership, archive and access rules.
Lesson Service	Create/publish lessons and coordinate lesson documents.
Document Service	Validate uploads, coordinate private storage and processing status.
Assignment Service	Create/publish/close assignments and manage assignment state.
Ai Service	Coordinate AI requests, prompts, answers, citations, and usage recording.
Rag Service	Retrieve authorized document chunks and prepare grounded context.
Practice Service	Generate and store practice sets and questions.
Usage Service	Apply request limits and record token/cost information.

Repository Layer:

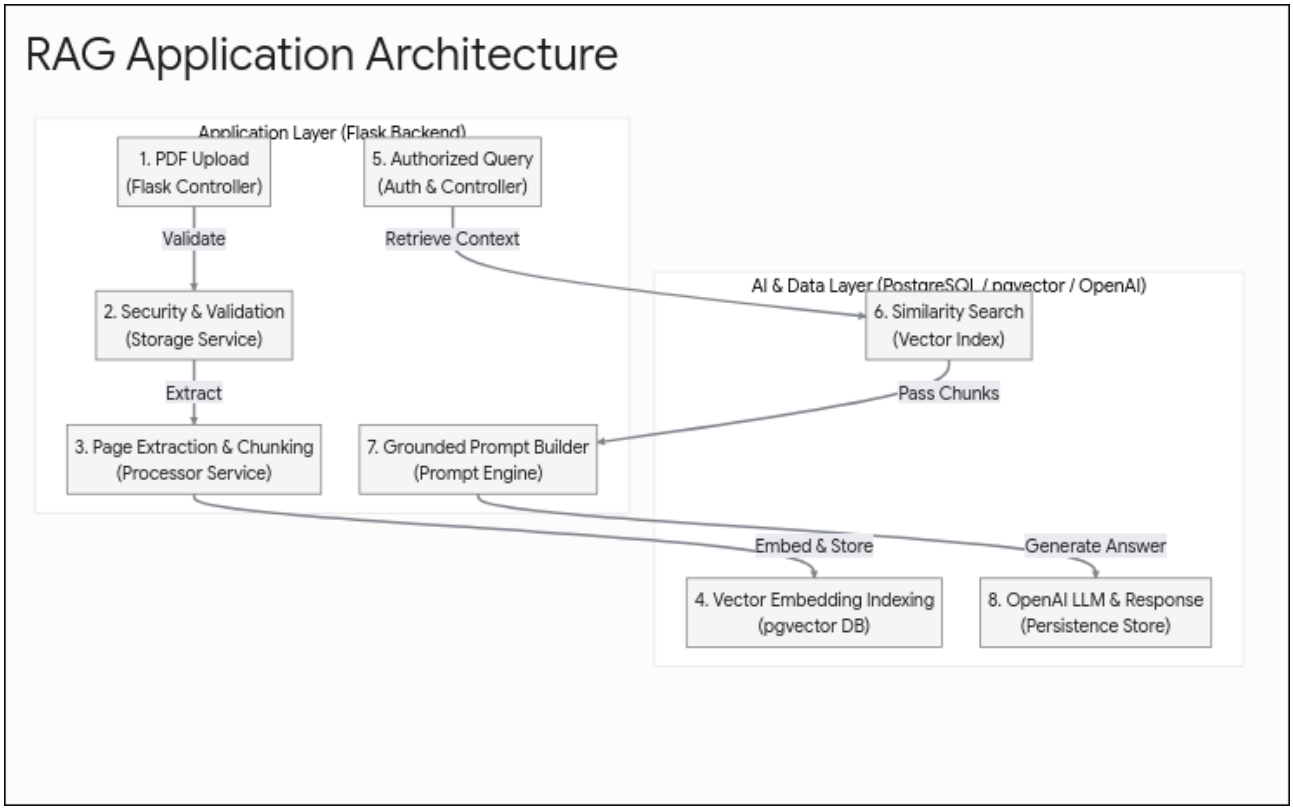
Repositories provide database access without UI logic. Example repositories include UserRepository, ClassRepository, ClassMemberRepository, LessonRepository, DocumentRepository, AssignmentRepository, ConversationRepository, PracticeRepository, and AIUsageRepository. Parameterized SQL or a safe database abstraction must be used instead of concatenating user input into SQL.

Project Structure

```
project_name/
├── app/
│   ├── __init__.py          # Flask application factory
│   ├── config.py            # configuration classes
│   ├── extensions.py        # db, login_manager, bcrypt, etc.
│   ├── models/              # domain/data models
│   ├── repositories/        # MySQL data access
│   ├── services/            # business rules / workflows
│   ├── routes/              # Flask blueprints
│   ├── forms/               # WTForms or validation helpers
│   ├── templates/           # Jinja2 pages
│   └── static/              # CSS, JS, images
├── tests/
├── migrations/ or sql/
├── docs/
├── run.py
├── requirements.txt
└── README.md
```

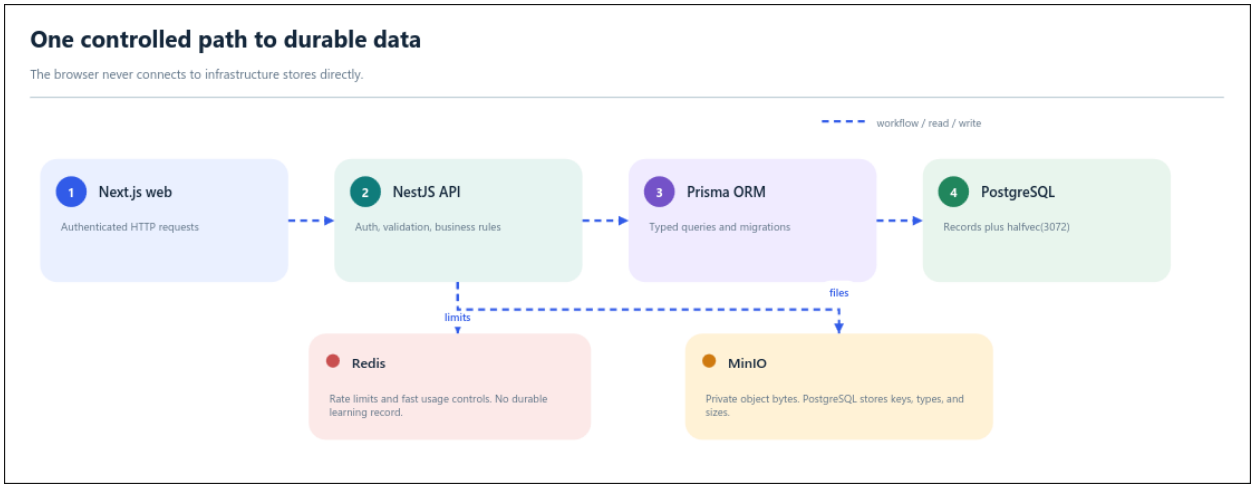
SYSTEM Architecture

The proposed request flow is Browser → Flask Routes/Controllers → Services → Repositories → PostgreSQL. The AI flow adds document processing and retrieval components: PDF Upload → Validation → Private Storage → Page Extraction → Chunking → Embedding/Indexing → Authorized Retrieval → Grounded Prompt → OpenAI API → Saved Answer + Sources.



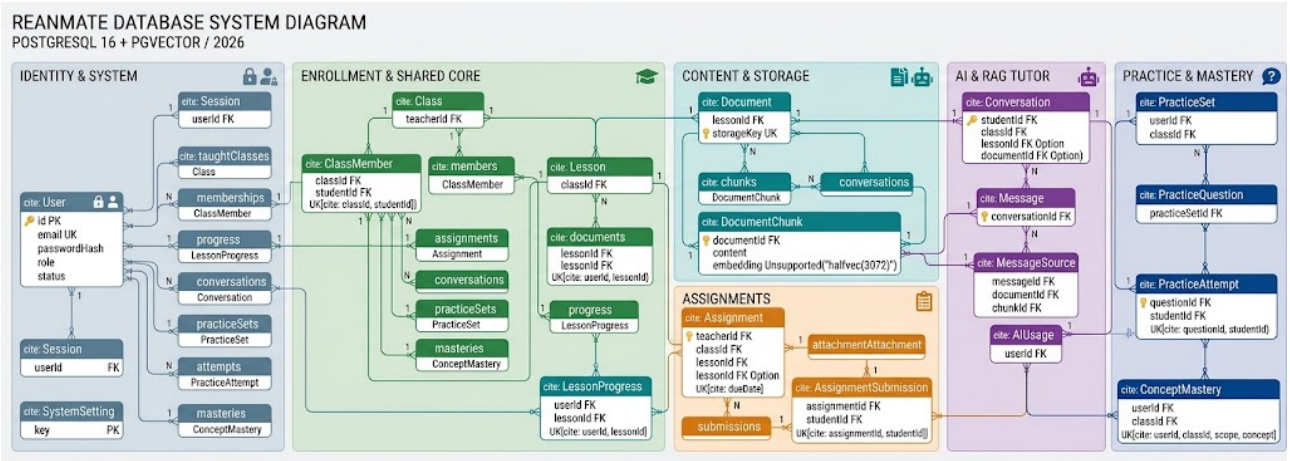
Database Design

The API owns every data access decision. PostgreSQL is durable state; Redis and MinlO solve separate operation problems.

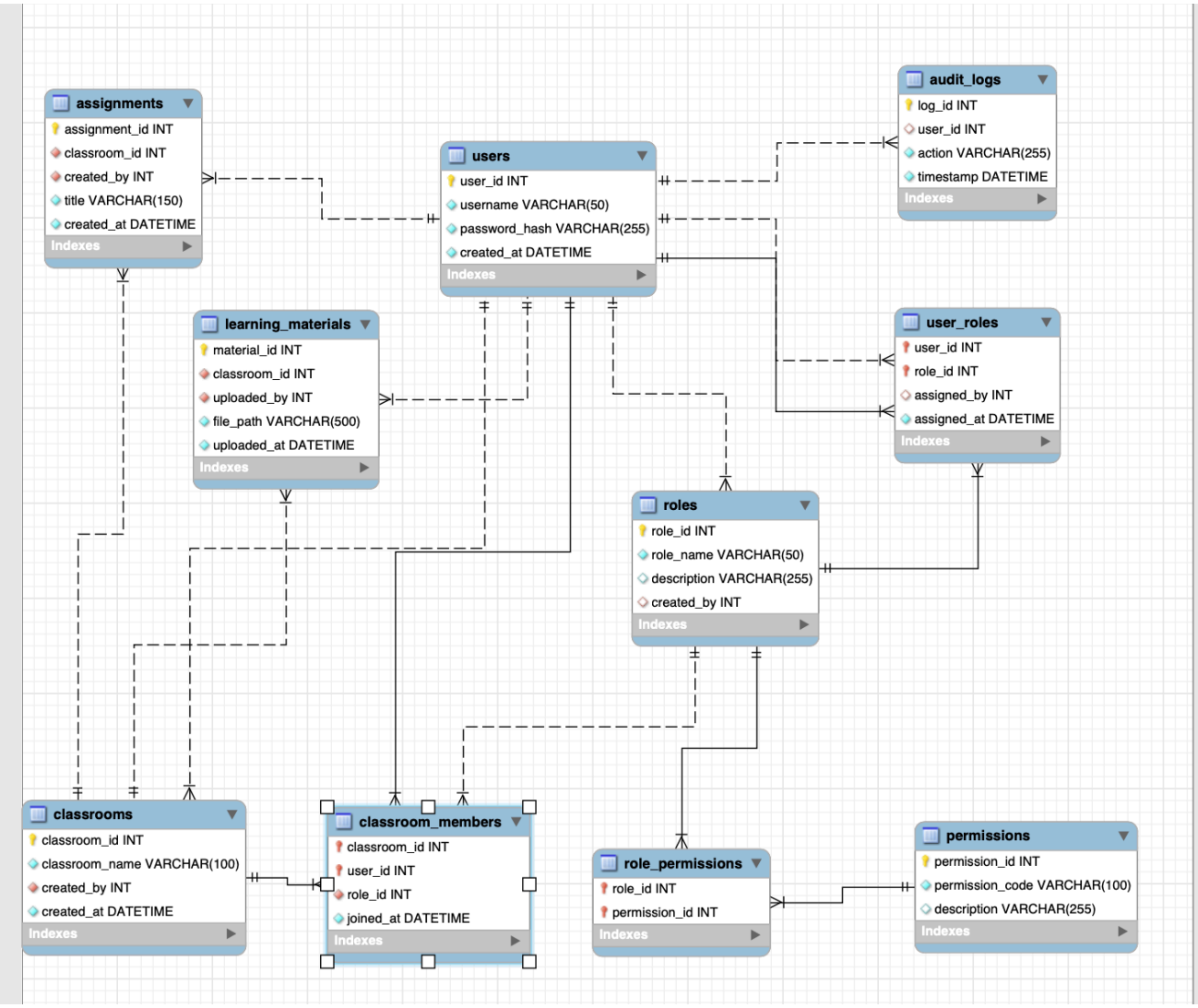


Why this design:

- DATABASE_URL and storage credentials stay in the backend environment only.
- File bytes are not databases column and relational rows keep private object metadata
- Postgre remains authoritative even when Redis or MinlO is temporarily unviable.



ERD Diagram



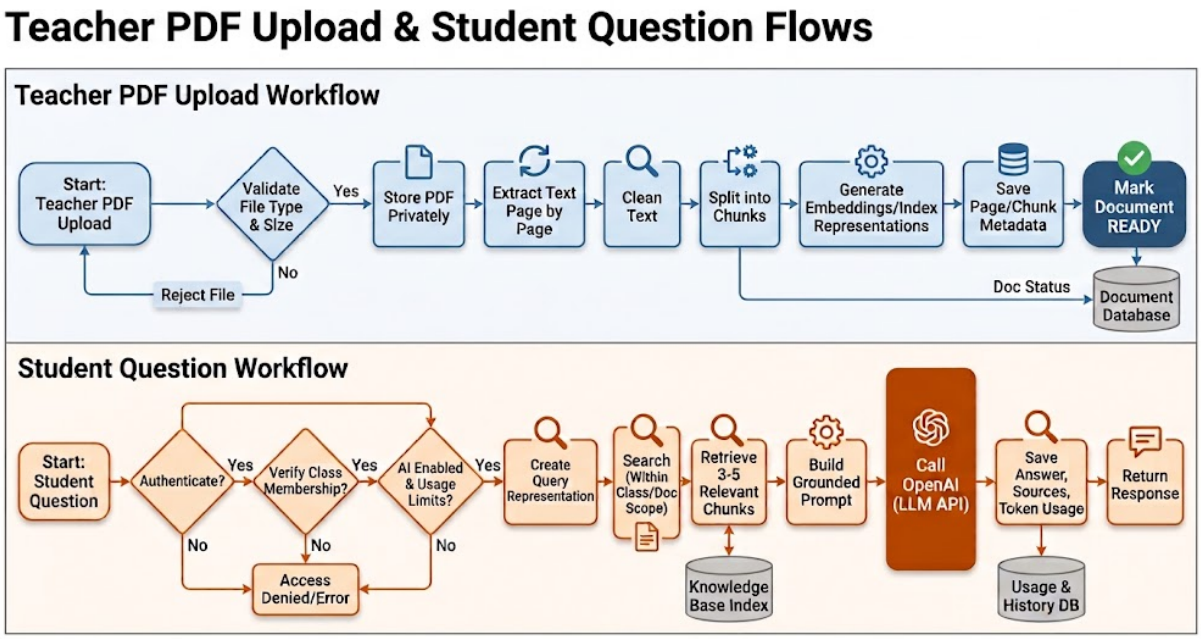
Main Relation

- One teacher can own many classes.
- A class can contain many students through ClassMember.
- A class can contain many lessons.
- A lesson can have one or more associated documents depending on final implementation.
- A document can contain many page-aware document chunks.
- A class can contain many assignments.
- An assignment can have progress records for many students.
- A conversation belongs to a user and can be scoped to a class/lesson/document.

- A conversation contains many messages.
- An AI message can have multiple source references.
- AI Usage records connect usage to a user and optionally a class.

Implementation Plan

User Interface → Student Experience → Teacher Experience. → Admin Experience →AI Processing Workflow



AI Response Rule

- Teacher material is the primary evidence.
- If the material is insufficient, the system clearly says so.
- The AI must never fabricate a source page.
- Explanations should be readable and appropriate for students.
- Khmer, English, and mixed-language questions are supported.
- Useful English technical terms may be preserved in Khmer explanations.
- For assessed assignments, the tutor should prefer explanations, hints, examples, and step-by-step guidance rather than automatically providing final answers.

API / Route Plan

Area	Example Endpoint
Authentication	POST /auth/register, POST /auth/login
Classes	GET/POST /classes, POST /classes/join
Students	GET /classes/<id>/students
Lessons	GET/POST /classes/<id>/lessons
Documents	POST /lessons/<id>/documents
Assignments	GET/POST /classes/<id>/assignments, PATCH /assignments/<id>
AI	POST /ai/ask, POST /ai/practice
Admin	GET /admin/ai-usage

Security Design

- Hash passwords securely.
- Keep the OpenAI API key on the backend only.
- Apply server-side role and ownership guards.
- Validate resource access on every protected request to prevent IDOR.
- Use private/signed document URLs where applicable.
- Validate PDF MIME type and maximum file size.
- Rate-limit authentication, upload, and AI endpoints.
- Log administrative actions and AI errors without logging secrets.
- Keep classes private by default.

AI Usage Controls

Limit	Demo Setting
Total demo users	10
AI Tutor	30 requests/user/day
Burst rate	3 AI requests/minute/user
Monthly user ceiling	500 AI requests/user
AI lesson	2 sessions/day if enabled
Practice generation	3 sets/day

Testing

Function Testing

Test	Expected Result
Teacher creates a class	Class is created and a join code is available.
Student joins valid class	Student becomes a class member.
Student uses disabled/archived code	Join is rejected.
Teacher uploads valid PDF	PDF is accepted and processing begins.
Teacher uploads invalid/oversized file	Upload is rejected with a clear error.
Student opens published lesson	Authorized student can view the PDF.
Teacher publishes assignment	Assignment appears to authorized students.
Student asks grounded AI question	Answer is returned with a source document/ page.
Khmer/English/mixed question	Tutor responds appropriately in requested language.
Admin views AI usage	Usage and estimated cost are visible.

Security Testing

- Student attempts to access another class by changing a URL ID.
- Student attempts to access another class document directly.
- Teacher attempts to edit another teacher's class.
- User attempts to change their role through a frontend request.
- AI endpoint is called above its rate limit.
- Vector retrieval attempts to cross a class boundary.

- Non-PDF and oversized uploads are submitted.

AI Quality Testing

- Question is directly answered in the PDF.
- Question is partially answered in the PDF.
- Question is absent from the PDF and the AI should state that the material is insufficient.
- Khmer-only question.
- English-only question.
- Mixed Khmer/English question.
- Student makes an incorrect assumption and the tutor corrects it using the source material.
- Source page opens the correct PDF location.

Development Methodology and Timeline

Development Phase

Phase	Main Work	Acceptance Evidence
0 – Foundation	Repository, Flask frontend/backend setup, MySQL, environment, lint/test baseline.	Application and database run together.
1 – Authentication	Register/login/logout, roles, protected routes.	Role-based access works.
2 – Private Classes	Create class, join code, join/leave, student list, archive.	Class flow works without AI.
3 – Lessons & PDFs	Lessons, PDF upload/storage, validation, viewer, processing status.	Student securely opens teacher PDF.
4 – Assignments	Teacher create/publish; student dashboard and assignment page.	Deadlines and visibility work.
5 – Document AI Processing	Page extraction, chunking, embeddings/indexing.	Relevant chunks are retrieved.
6 – RAG + OpenAI	Grounded prompt, API call, citations, no-answer behavior.	AI answer includes correct source.
7 – Study Workspace	PDF + AI split workspace, quick actions, language modes.	Approved learning UI works.
8 – Practice	Generate/save practice questions and explanations.	Practice can be reused.
9 – Usage Controls	Rate limits, token/cost tracking, admin controls.	AI usage remains controlled.
10 – QA & Demo	Security, responsive testing, ten-user accounts, real lesson validation.	Reliable demonstration.

Proposed Weekly Timeline

Week	Task	Deliverable
1	Finalize scope, team roles, requirements, and proposal writing	Approved proposal & project direction
2	OOP class design, architecture, ERD, and Flask foundation	Class diagram, ERD & project skeleton
3	Authentication, roles, private classes, and membership	Working login, access control & class module
4	Lessons, PDF upload, and Assignments modules	Lesson, PDF & Assignment modules
5	PDF processing, chunk retrieval, RAG, and OpenAI integration	Searchable document chunks & Grounded AI Tutor

6	Study Workspace development and UI refinement	Student workspace
7	Practice features and usage controls	Practice tools & limits
8	Security, functional testing, and bug fixing	Test evidence & bug fixes
9	Final bug fixing and demo preparation	Stable MVP
10	Final report writing and defense PPTX preparation	Final submission

Risk, Ethics, and Privacy

Risk ad Mitigating

Risk	Impact	Mitigation
AI API cost increases	High	Rate limits, daily/monthly caps, usage tracking, global AI control.
AI gives unsupported information	High	RAG grounding, source citations, insufficient-material behavior.
Cross-class data exposure	Critical	Membership guards, ownership checks, retrieval filters, security tests.
PDF processing failure	Medium	File validation, processing status, error handling.
Scope becomes too large	High	Keep MVP exclusions and prioritize core proof.
Team member availability	Medium	Clear responsibilities and shared Git activity.
Security mistakes	High	Server-side authorization and dedicated security testing.

Privacy:

- Classes are private by default.
- Documents are not public URLs.
- Student conversations are not exposed to other students.
- Teacher analytics should prefer aggregate signals rather than reading private chats.
- Administrative access to user content should be minimal and audited.
- Technical logs should avoid secrets and unnecessary private content.

Ethical AI Use

ReanMate is intended to support learning rather than replace the student's work. For marked assignments, the tutor should prefer explanations, hints, examples, and step-by-step guidance rather

than automatically providing a final answer. The system should also communicate uncertainty when the teacher's material does not contain enough information to answer.

Conclusion

ReanMate proposes a focused AI-assisted learning platform that combines private classes, teacher-controlled PDF materials, assignments, and a document-grounded AI tutor. The design directly supports the Advanced OOP course requirements by separating domain objects, services, repositories, routes, and database access. It also provides a practical R&D problem: how to make AI assistance useful while maintaining private class boundaries, source traceability, and controlled usage.

The MVP is deliberately limited to approximately ten users so that the team can prioritize a reliable end-to-end proof. The core demonstration is complete when a teacher creates a private class, uploads a real lesson, creates an assignment, a student joins the class, studies the PDF, asks a useful question in Khmer or English, receives a grounded answer with the correct source page, and completes the learning flow without manual database intervention.

Future Work

- Flashcards and improved AI lesson mode.
- Richer assignment submissions and a stronger progress page.
- Announcements and due-date notifications.
- DOCX/PPTX support.
- Mock exams and weak-topic detection.
- Teacher review workflow.
- School/organization accounts and institution analytics.
- AI voice questions, OCR, lecture transcription, personalized tutoring, study plans, and AI-generated diagrams.

The product plan recommends delaying payment systems, school administration, additional AI providers, and other broad LMS features until the core student and teacher value is proven.

Reference

[1] S. Sokcheat, “Advanced Object-Oriented Programming,” lecture notes, University of Technology and Entrepreneurship, Phnom Penh, Cambodia, 2026.

[2] GeeksforGeeks, “Object-Oriented Programming (OOPs) Concept in Java,” GeeksforGeeks, 2026. [Online]. Available: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

Power Point Slide

PROBLEM AND MOTIVATION

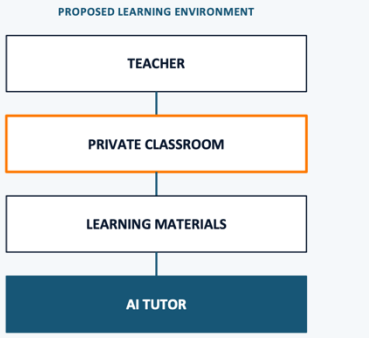
[01] Students may find learning materials difficult to understand without additional guidance.

[02] Teachers need a private space to organize classrooms and share materials.

[03] Students need a convenient way to ask questions about their lessons.

[04] Generic AI tools may not answer directly from the teacher's learning materials.

ReanMate's motivation: combine private classroom materials with an AI tutor in one focused learning environment.



A shared context for teaching, studying, and asking lesson-related questions.

OBJECTIVES

GENERAL OBJECTIVE

Build ReanMate as an object-oriented learning environment.

Design and develop a web application that combines private classrooms, learning materials, and AI-assisted tutoring.

[01] **Secure Access**
Implement authentication, sessions, Teacher/Student roles, and role-based permissions.

[02] **Private Learning Spaces**
Enable teachers to create classrooms and upload PDFs for enrolled students.

[03] **Contextual Support**
Let students access classroom materials and ask the AI tutor lesson-related questions.

[04] **OOP & Accountability**
Apply OOP principles and maintain audit logs for important system activities.

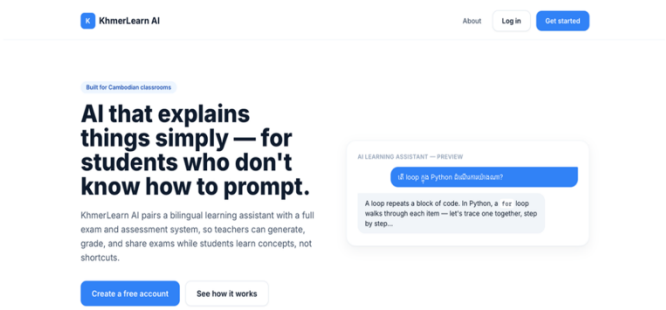
UX/UI

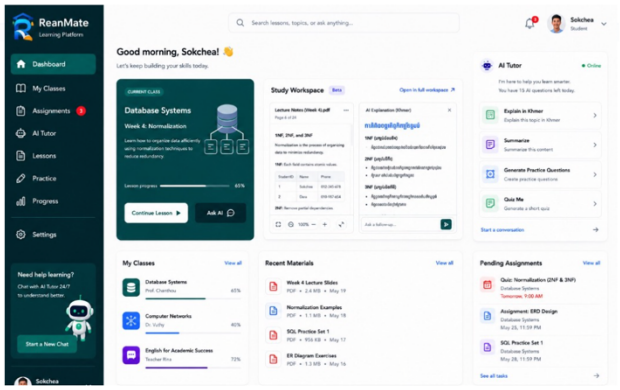
UX: Simple navigation, clear user flow, easy learning experience

UI: Clean, responsive, Khmer/English support

Key Screens: Login, Dashboard, Classroom, PDF Workspace, AI Tutor

Design Goal: Simple, accessible, student-friendly





SCOPE KEEPS THE SEMESTER PROJECT REALISTIC

INCLUDED IN THE PROPOSAL

- Web application
- Teacher & Student accounts
- PDF learning materials
- Classroom management
- Permission management
- SQLite3 database
- Private classrooms
- Authentication & RBAC
- AI tutor
- Assignment management
- Audit logging
- OOP architecture

Boundary principle: deliver the core private-learning workflow first, then consider optional extensions.

NOT INCLUDED

- Mobile application
- Large-scale enterprise deployment
- Advanced analytics
- Complex third-party identity providers
- Features outside the core learning/classroom system

THREE ROLES, THREE ACCESS PERSPECTIVES

[01] CORE ROLE

Teacher

I want to create a private classroom and upload learning materials so my students can study using the resources I provide.

[02] CORE ROLE

Student

I want to join my private classroom, access materials, and ask the AI tutor questions so I can better understand my lessons.

[03] CONDITIONAL ROLE

Admin

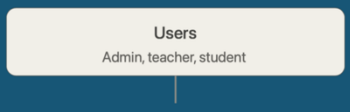
If included, I want to manage users and system permissions so the platform remains secure and organized.

TEACHER → CLASSROOM + MATERIALS

STUDENT → LEARNING + QUESTIONS

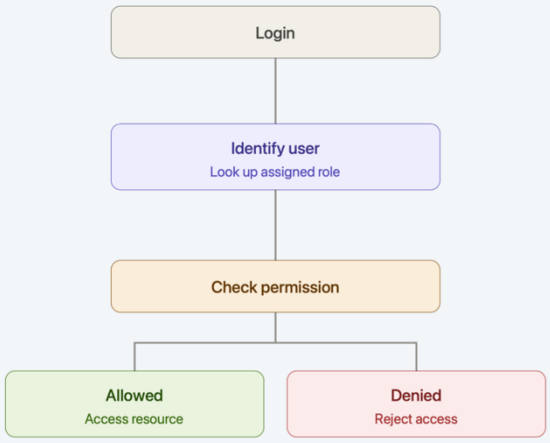
ADMIN → PLATFORM GOVERNANCE

RBAC



Roles
- Each user is assigned a role.
- Each role has specific permissions.

Permissions
- Create
- Read
- Update
- Delete
- Manage classroom
- Upload materials
- Manage assignments
- Ask AI tutor



PRIORITIZE THE CORE LEARNING WORKFLOW

MUST-HAVE FEATURES

- Registration, login & authentication
- Private classroom creation
- Classroom membership
- Teacher/Student role management
- PDF upload
- Learning-material access
- AI tutor
- Permission checking & sessions

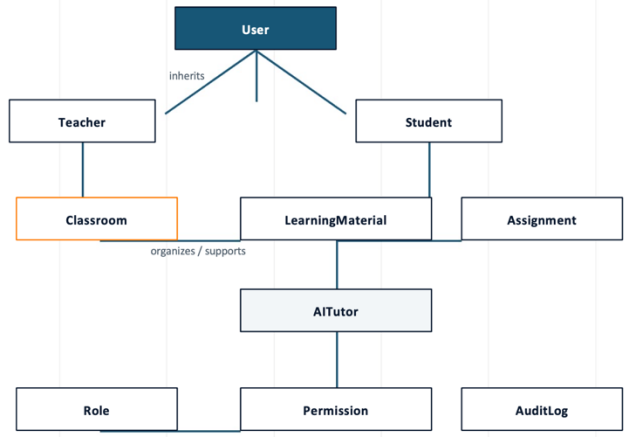
Delivery principle: establish secure classroom access before adding optional learning enhancements.

NICE-TO-HAVE EXTENSIONS

- Assignment management
- Audit logging
- Advanced classroom management
- Learning progress features
- Additional AI learning functions

OOP MAKES REANMATE MODULAR

CONCEPTUAL CLASS STRUCTURE



OOP PRINCIPLES IN THE DESIGN

[01] Encapsulation

Keep user, classroom, assignment, and permission data with related operations.

[02] Inheritance

Teacher and **Student** inherit common properties and behavior from **User**.

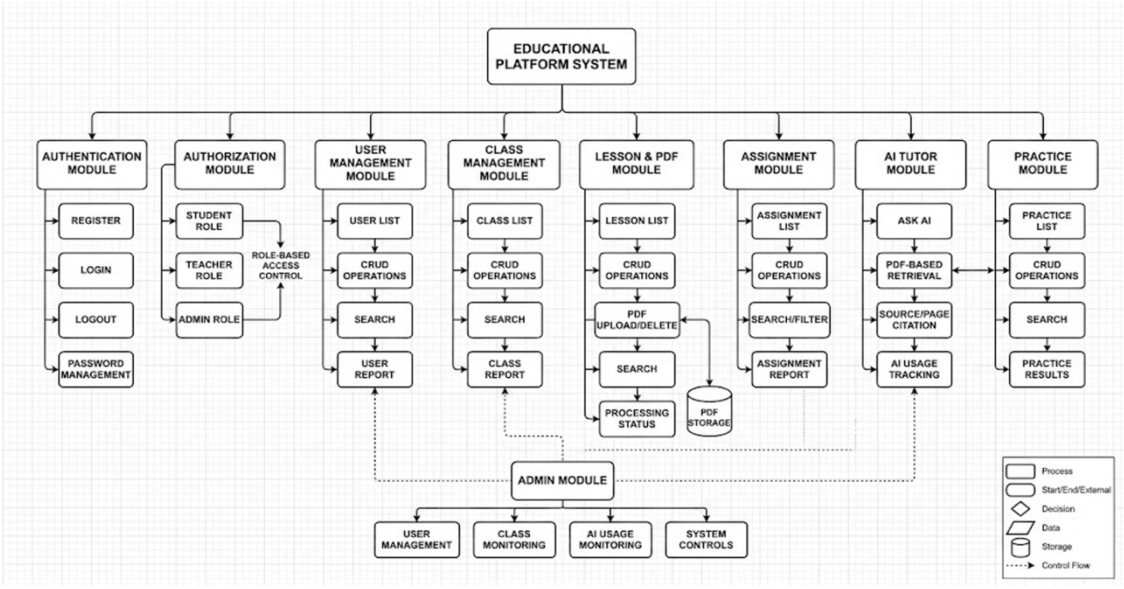
[03] Polymorphism

Different user types can provide different behaviors or permissions.

[04] Abstraction

Hide database, authentication, and AI implementation details behind classes or interfaces.

ARCHITECTURE
PROPOSED SYSTEM
FEATURE



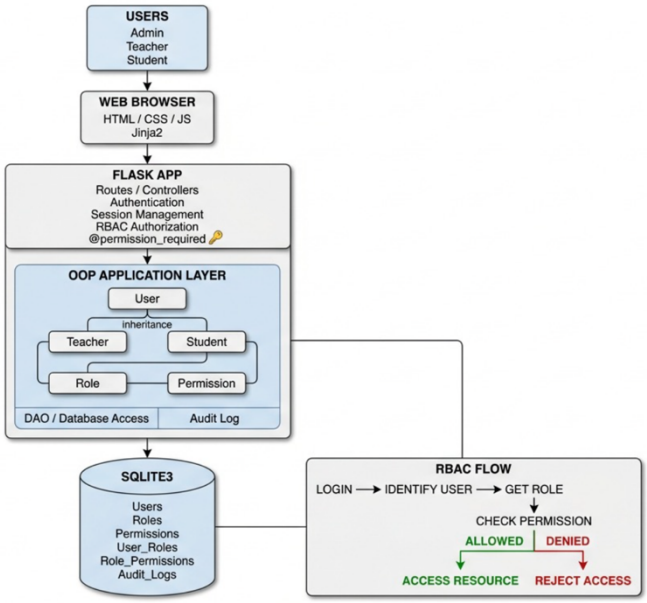
LAYERED ARCHITECTURE CONNECTS LEARNING TO DATA

WEB BROWSER	HTML / CSS / JavaScript / Jinja2 Interface for teachers and students.
FLASK APPLICATION	Routes · Auth · Sessions · RBAC Handles requests, identity, and permission checks.
OOP APPLICATION	User · Classroom · Material · Assignment · AITutor Domain classes organize ReanMate behavior.
PERSISTENCE	DAO / Database Access → SQLite3 Stores accounts, classrooms, materials, and activity records.

REANMATE LEARNING FLOW

TEACHER creates classroom → UPLOAD learning material → STUDENT joins classroom → OPEN material → ASK AI tutor

Architecture Diagram



Mysql PROVIDES A FOCUSED DATA MODEL

IDENTITY & AUTHORIZATION

Users
PK user_id
account data

Roles
PK role_id
role definitions

Permissions
PK permission_id
allowed actions

User_Roles
FK user_id, role_id
many-to-many link

Role_Permissions
FK role_id, permission_id
many-to-many link

Audit_Logs
PK/FK log_id / user_id
important activities

Users → Roles → Permissions

PRIVATE LEARNING DOMAIN

Classrooms
PK classroom_id
private spaces

Classroom_Members
FK classroom_id, user_id
membership link

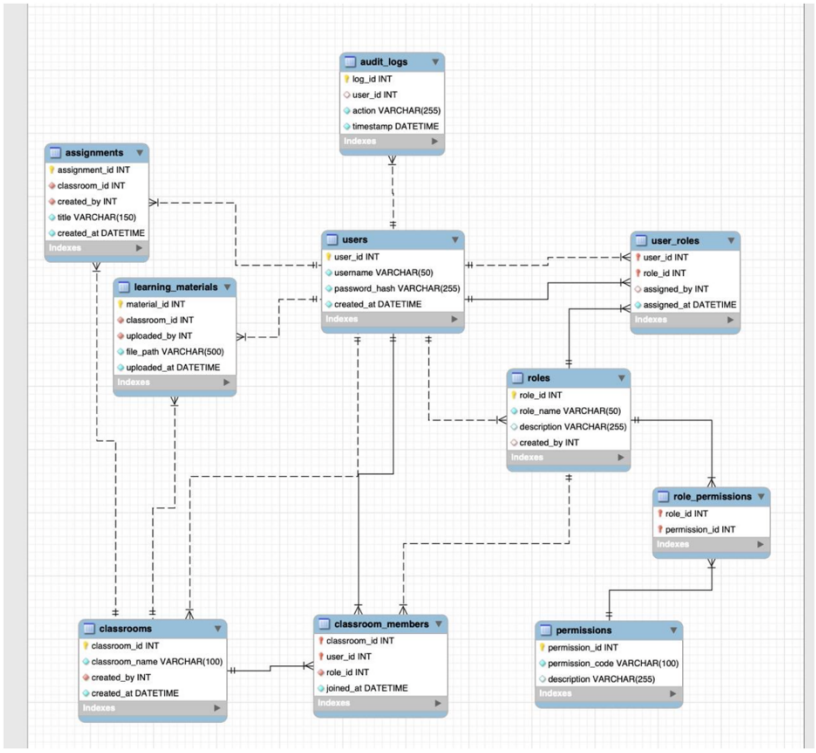
Learning_Materials
FK classroom_id
uploaded PDFs

Assignments
FK classroom_id
learning activities

Users → Classroom_Members → Classrooms → Materials / Assignments

DATABASE: SQLite3 DESIGN CHECKS: Primary keys · Foreign keys · Relationships · Important constraints

ERD Diagram



A FOUR-PHASE PLAN GUIDES DEVELOPMENT

01 RESEARCH & REQUIREMENTS

- Study OOP concepts
- Study RBAC
- Identify ReanMate requirements
- Define scope

02 SYSTEM DESIGN

- Use case & class diagram
- Architecture & ERD
- Database design
- OOP structure

03 DEVELOPMENT

- Flask + SQLite3 foundation
- Auth, roles & classrooms
- Materials & AI tutor
- Permission system

04 INTEGRATION & REFINEMENT

- Connect components
- Improve UI
- Fix issues
- Prepare demo & docs

EDITABLE TEAM RESPONSIBILITIES	[Member 1]	[Member 2]	[Member 3]	[Member 4]
	Requirements / design	Backend / database	Frontend / UI	Testing / documentation

EXPECTED OUTPUTS DEFINE THE DELIVERABLE

LEARNING PLATFORM OUTPUTS

- ReanMate web application
- Private classroom system
- Teacher and Student accounts
- Learning-material management
- AI tutor

TECHNICAL AND SUBMISSION OUTPUTS

- OOP-based architecture
- RBAC and permission system
- SQLite3 database
- Authentication and session management
- Documentation, source code, and final demonstration

PROJECT STATUS Expected deliverables only — the system is currently in the planning/design stage.