Application Design — CampusConnect

Date: 2025-11-17

Contents

- Architecture Overview
- Domain Model
- Database Schema
- API Design
- Services
- Security
- Events & Messaging
- Diagrams
- Testing Strategy

Architecture Overview

Client apps connect to a backend API exposing Q&A, search, and notifications. Background workers handle delivery and indexing.

Domain Model

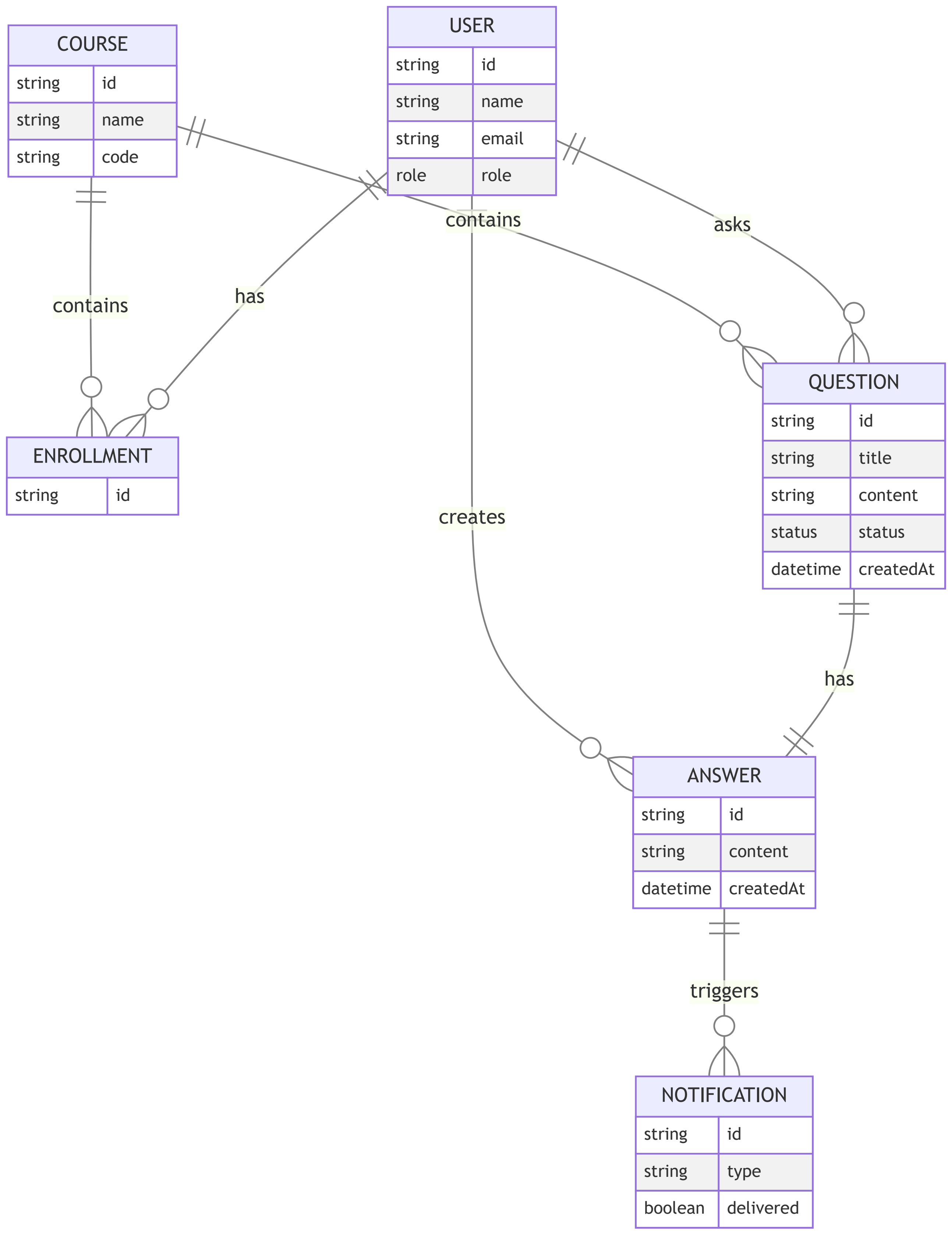
Core Entities

- User, Course, Enrollment
- Question, Answer (single official), Vote
- Tag, QuestionTag, Attachment
- Subscription, Notification
- AuditLog

Invariants

- One official Answer per Question
- One vote per User per Question
- Question belongs to a single Course

Database Schema



API Design

```
POST /api/auth/login
GET  /api/courses
GET  /api/courses/{courseId}/questions?page&size&query
POST /api/courses/{courseId}/questions
GET  /api/questions/{id}
POST /api/questions/{id}/vote { value: 1|-1 }
POST /api/questions/{id}/answer { content, official }
PATCH /api/answers/{id} { content }
GET  /api/dashboard/stats
```

Services

- AuthService, CourseService
- QuestionService, AnswerService, VoteService
- NotificationService, SearchService, AuditService

Security

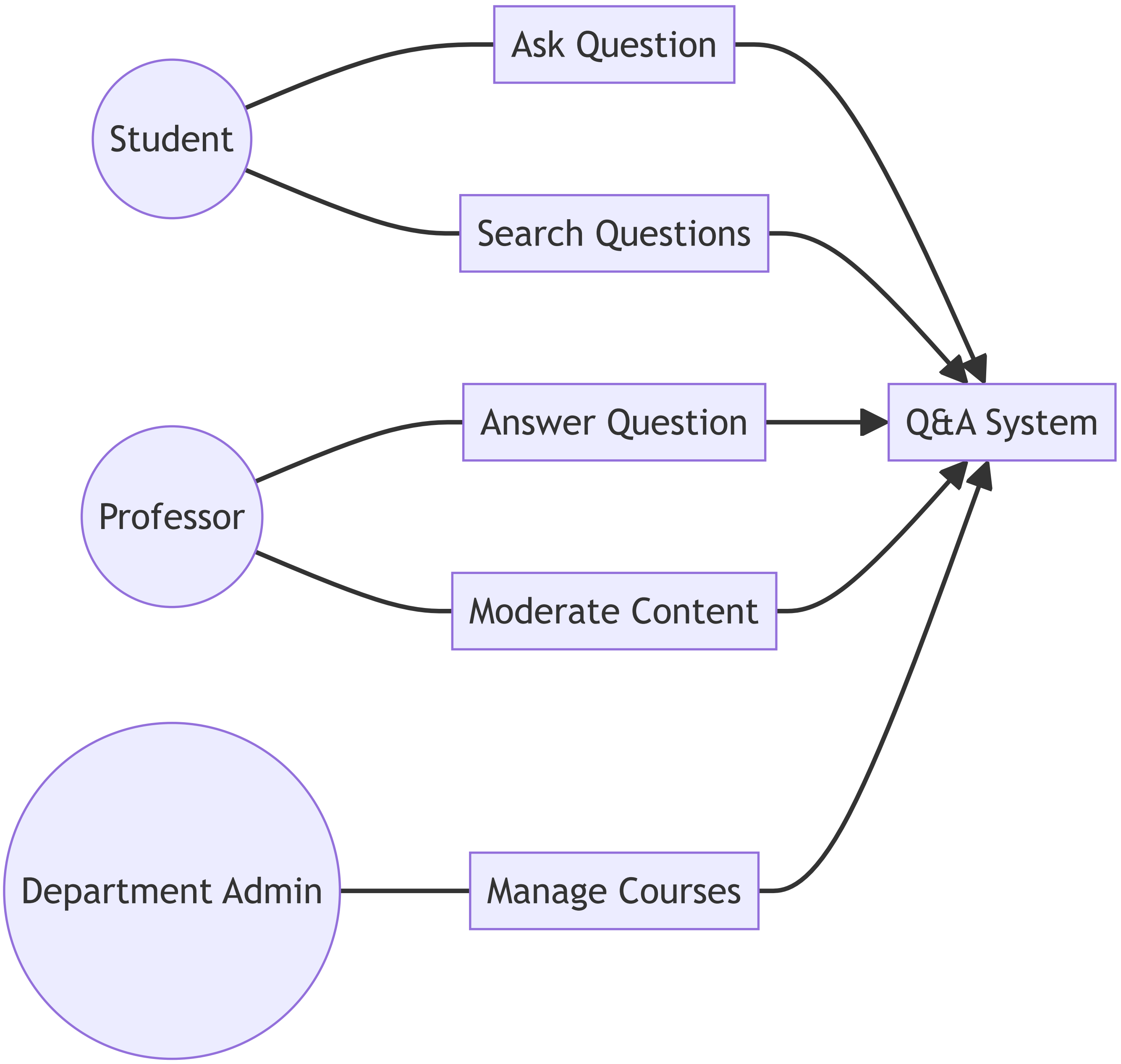
- JWT bearer auth with role checks
- Course-scoped access based on enrollment
- Input validation and audit logging

Events & Messaging

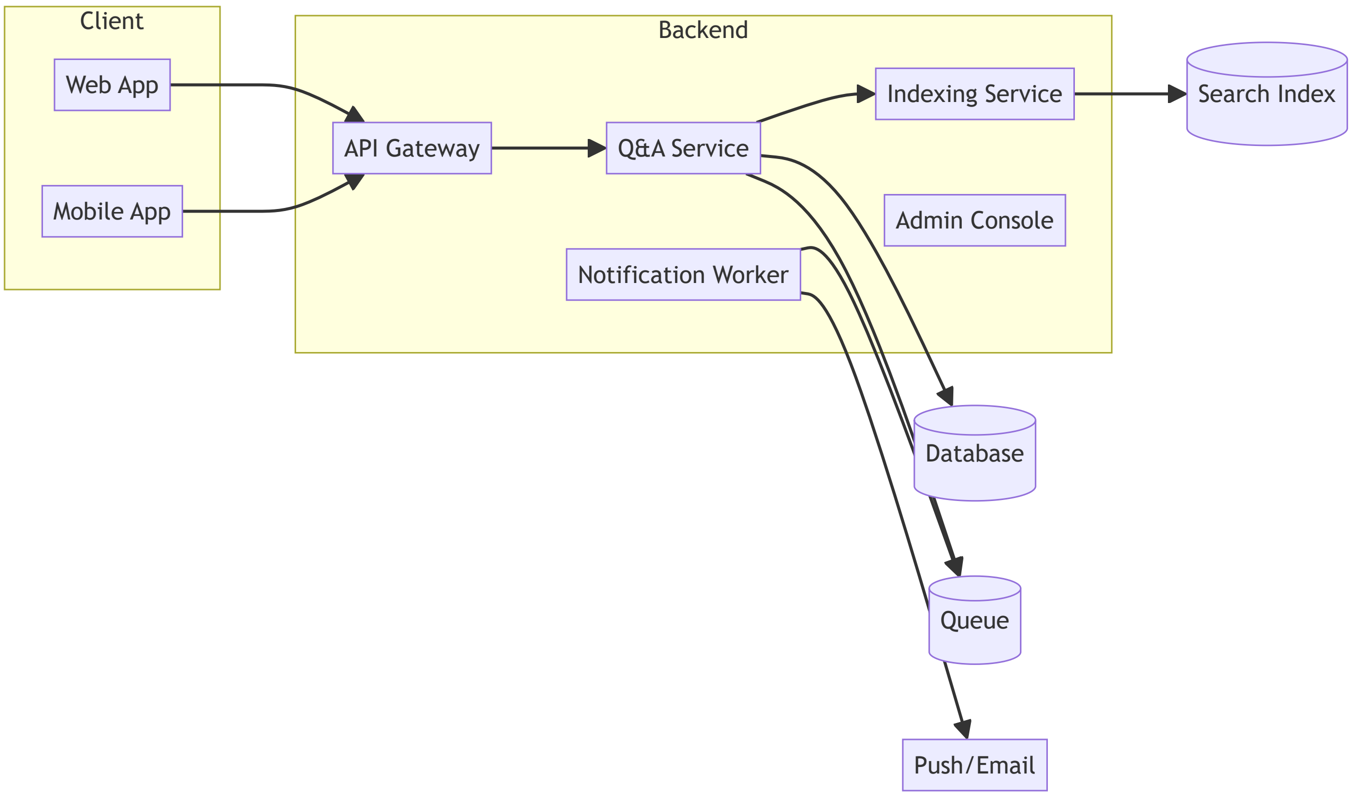
```
question.created { questionId, courseId }
answer.created { questionId, answerId, official }
question.voted { questionId, delta }
```

Diagrams

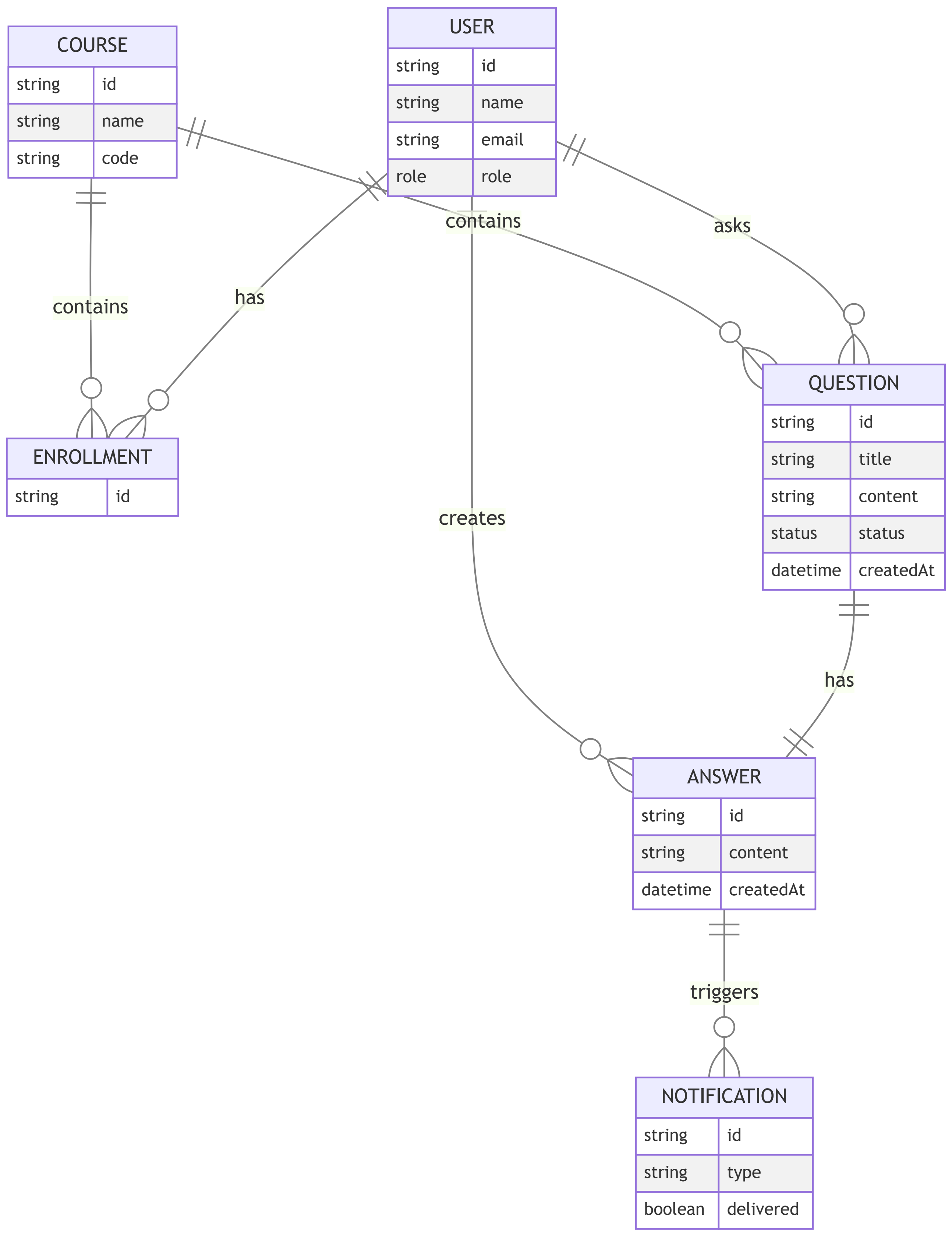
Use Case



Component

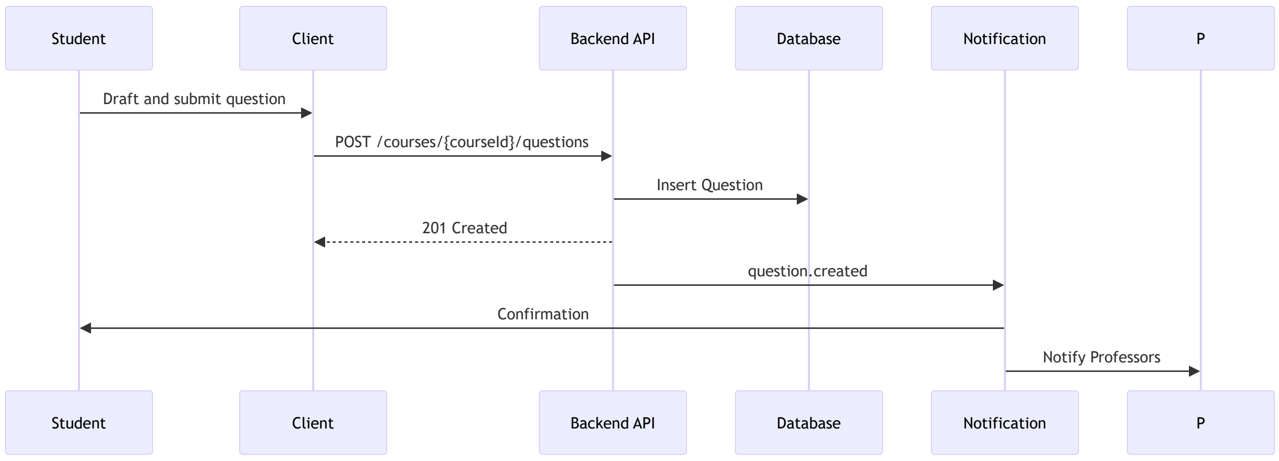

Client
Web App
Mobile App
Backend
API Gateway
Q&A Service
Indexing Service
Search Index
Admin Console
Notification Worker
Database
Queue
Push/Email

Database ER

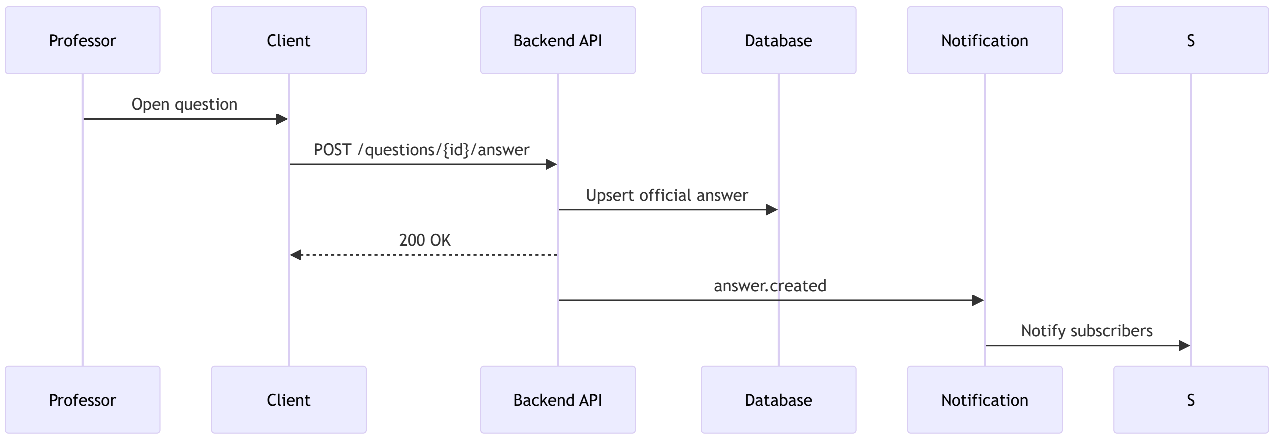


COURSE
string | id
string | name
string | code
USER
string | id
string | name
string | email
role | role
ENROLLMENT
string | id
QUESTION
string | id
string | title
string | content
status | status
datetime | createdAt
ANSWER
string | id
string | content
datetime | createdAt
NOTIFICATION
string | id
string | type
boolean | delivered
contains
has
contains
asks
creates
has
triggers

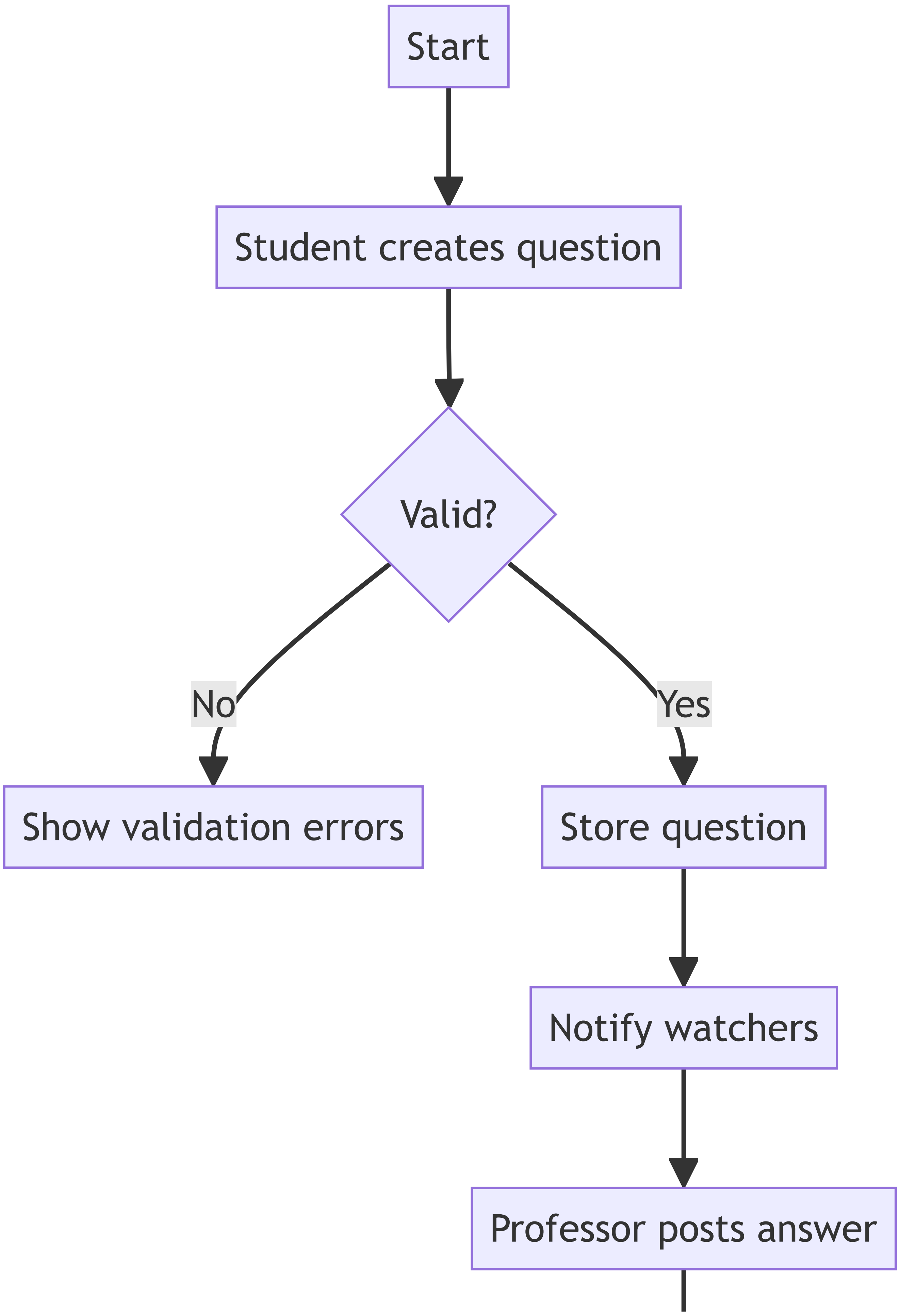
Sequence: Ask Question

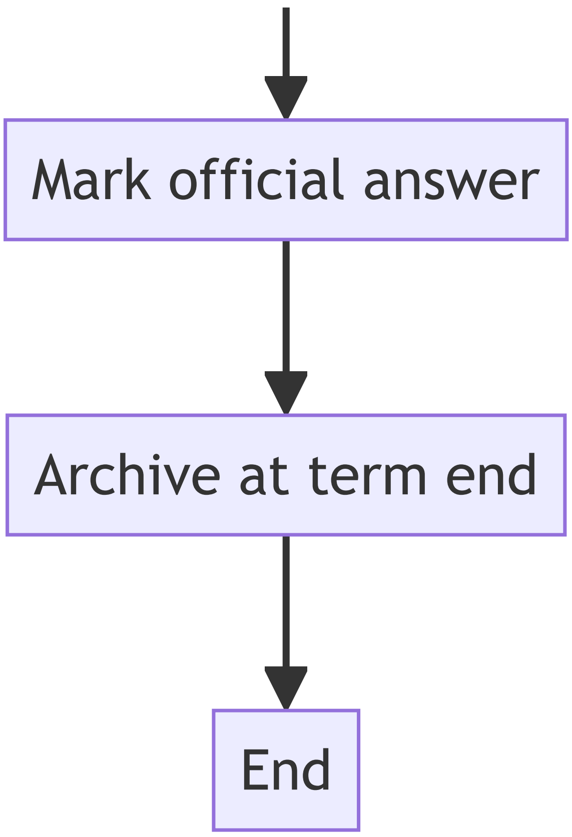


Sequence: Professor Answers

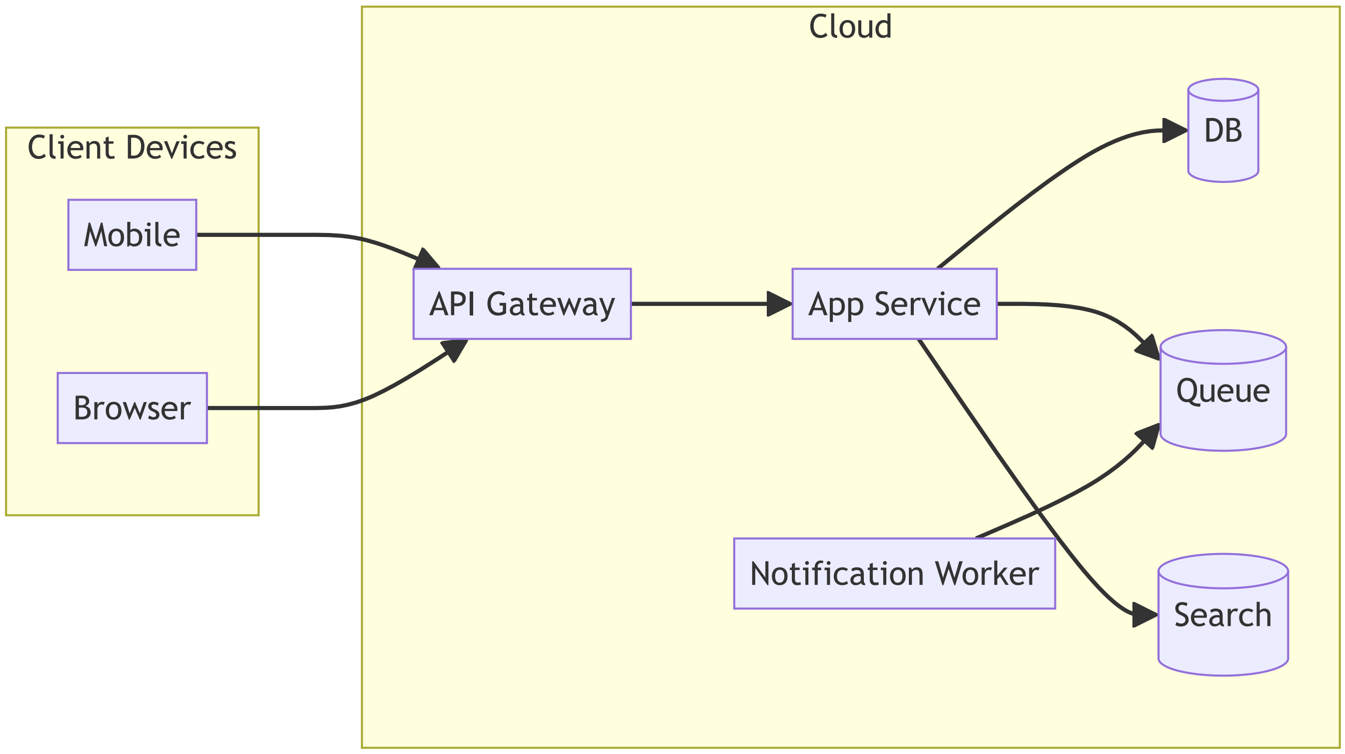


Activity


Start
Student creates question
Valid?
No
Yes
Show validation errors
Store question
Notify watchers
Professor posts answer



Deployment



Testing Strategy

- Unit tests for services and invariants
- Integration tests with a real DB (Testcontainers)

- Contract tests for REST API
- E2E for core Q&A flows