EduCards: Simplified Database Design

Overview

This document outlines a simplified database design for the EduCards application. The design focuses on essential tables and relationships to support the core functionality: teachers uploading PDFs, AI generating flashcards, and students studying and competing on a leaderboard.

Database Tables

1. User

Stores information about teachers and students.

Field	Type	Description
id	Integer (PK)	Unique identifier
username	String	User's login name
password	String	Hashed password
email	String	User's email address
is_teacher	Boolean	True if user is a teacher
is_student	Boolean	True if user is a student
date_joined	DateTime	When the user account was created

2. Document

Stores information about uploaded PDF files.

Field	Type	Description
id	Integer (PK)	Unique identifier
title	String	Document title

file_path	String	Path to the stored PDF file
upload_date	DateTime	When the document was uploaded
teacher_id	Integer (FK)	Teacher who uploaded the document

3. Flashcard

Stores flashcards generated from documents.

Field	Type	Description
id	Integer (PK)	Unique identifier
document_id	Integer (FK)	Document this card was created from
question	Text	Flashcard question
answer	Text	Flashcard answer
difficulty	String	Easy, Medium, or Hard

4. StudySession

Tracks student study sessions.

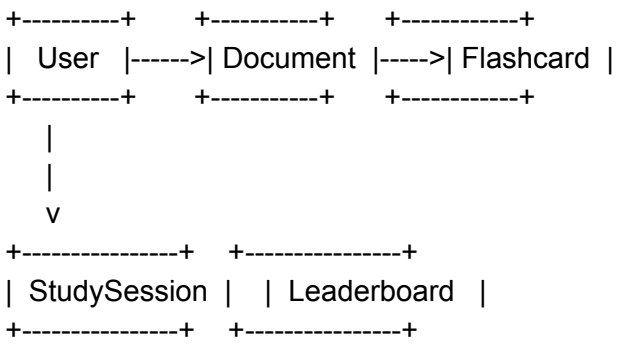
Field	Type	Description
id	Integer (PK)	Unique identifier
student_id	Integer (FK)	Student in the study session
start_time	DateTime	When the session started
end_time	DateTime	When the session ended
score	Integer	Points earned during the session
correct_count	Integer	Number of correct answers
total_count	Integer	Total number of flashcards reviewed

5. Leaderboard

Stores student rankings.

Field	Type	Description
id	Integer (PK)	Unique identifier
student_id	Integer (FK)	Student on the leaderboard
total_score	Integer	Student's cumulative score
rank	Integer	Student's position in the ranking
last_updated	DateTime	When the ranking was last updated

Entity Relationship Diagram



Django Models Implementation

```
# models.py
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.utils import timezone

class User(AbstractUser):
    is_teacher = models.BooleanField(default=False)
    is_student = models.BooleanField(default=False)

    def __str__(self):
        return self.username

class Document(models.Model):
    title = models.CharField(max_length=255)
    file_path = models.FileField(upload_to='documents/')
```

```
    upload_date = models.DateTimeField(default=timezone.now)
    teacher = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='documents')

    def __str__(self):
        return self.title

class Flashcard(models.Model):
    DIFFICULTY_CHOICES = [
        ('easy', 'Easy'),
        ('medium', 'Medium'),
        ('hard', 'Hard'),
    ]

    document = models.ForeignKey(Document, on_delete=models.CASCADE,
related_name='flashcards')
    question = models.TextField()
    answer = models.TextField()
    difficulty = models.CharField(max_length=10, choices=DIFFICULTY_CHOICES,
default='medium')

    def __str__(self):
        return f"Flashcard {self.id}: {self.question[:30]}..."

class StudySession(models.Model):
    student = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='study_sessions')
    start_time = models.DateTimeField(default=timezone.now)
    end_time = models.DateTimeField(null=True, blank=True)
    score = models.IntegerField(default=0)
    correct_count = models.IntegerField(default=0)
    total_count = models.IntegerField(default=0)

    def __str__(self):
        return f"Session {self.id} by {self.student.username}"

    @property
    def accuracy(self):
        if self.total_count > 0:
            return (self.correct_count / self.total_count) * 100
        return 0

class Leaderboard(models.Model):
```

```
    student = models.OneToOneField(User, on_delete=models.CASCADE,
related_name='leaderboard')
    total_score = models.IntegerField(default=0)
    rank = models.IntegerField(null=True, blank=True)
    last_updated = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"Rank {self.rank}: {self.student.username}"

    class Meta:
        ordering = ['rank']
```

Key Relationships

1. **Teacher** → **Documents**: One teacher can upload many documents
2. **Document** → **Flashcards**: One document generates many flashcards
3. **Student** → **Study Sessions**: One student can have many study sessions
4. **Student** → **Leaderboard**: One student has one leaderboard entry

Database Configuration (PostgreSQL)

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'educards_db',
        'USER': 'postgres',
        'PASSWORD': 'your_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```