# EduCards Frontend Development Guide

Hi Drin! This guide will help you understand the backend architecture and build the frontend for our EduCards application. As the backend is now complete, you can focus on creating an intuitive and responsive user interface.

## Table of Contents



## 1. Project Overview

EduCards is an educational application that uses AI to generate flashcards from uploaded PDF documents. The key features are:

- **User Management**: Teachers can upload documents, students can study with flashcards
- **Document Handling**: PDF uploads and processing
- **AI Integration**: Automatic flashcard generation from document content
- **Study System**: Interactive flashcard study sessions
- **Leaderboard**: Competitive element showing student rankings

The backend is built with Django and Django REST Framework, connected to a PostgreSQL database. You can build the frontend using any framework (React, Vue, Angular, etc.) as it will communicate with the backend via REST APIs.

## 2. Backend Structure

The backend follows a modular structure with these main apps:

- **accounts**: User management (registration, login, profiles)
- **documents**: Document upload and management
- **flashcards**: AI-powered flashcard generation and organization

- **study**: Study session tracking and progress
- **leaderboard**: Student rankings and competition

## 3. API Endpoints

Here are the main API endpoints you'll need to interact with:

### Authentication

- `POST /api/accounts/register/`: Register a new user (teacher or student)
  - Payload: `{username, email, password, password_confirm, is_teacher}`
  - Returns: User data with token
- `POST /api/accounts/login/`: Login and get authentication token
  - Payload: `{username, password}`
  - Returns: `{token, user_id, username, is_teacher, is_student}`
- `GET /api/accounts/profile/`: Get current user profile

### Documents (Teachers)

- `GET /api/documents/`: List all documents (teachers see their own, students see public)
- `POST /api/documents/upload/`: Upload a new document (PDF)
  - Requires: Multipart form with `file` and `title`
- `POST /api/documents/{id}/process/`: Process document to generate flashcards
  - Payload: `{num_cards, difficulty}`

### Flashcards

- `GET /api/flashcards/`: List all available flashcards
- `GET /api/flashcards/document/{document_id}/`: Get flashcards for a specific document

### Study (Students)

- `POST /api/study/start/`: Start a new study session
  - Payload: `{flashcard_ids: [1, 2, 3, ...]}`
  - Returns: Session info with flashcards
- `POST /api/study/{session_id}/answer/`: Record an answer
  - Payload: `{flashcard_id, is_correct, time_taken}`

- `POST /api/study/{session_id}/end/`: End a study session
- `GET /api/study/history/`: Get user's study history

### Leaderboard

- `GET /api/leaderboard/`: Get the current leaderboard
- `GET /api/leaderboard/export/`: Export leaderboard data (CSV format)

## 4. Authentication

The backend uses token-based authentication:

1. After login/registration, you'll receive a token
2. Include this token in subsequent requests:

   ```
   Authorization: Token YOUR_TOKEN_HERE
   ```

3. Store the token securely (localStorage or similar)
4. Include logic to handle token expiration and renewal

## 5. Frontend Requirements

The frontend should include these key components:

### For All Users

- **Login/Registration**: Forms with teacher/student selection
- **Navigation**: Different menus for teachers vs students

### For Teachers

- **Document Upload**: Form for uploading PDFs with metadata
- **Document Management**: List, view, and delete documents
- **Flashcard Generation**: Interface to process documents and generate flashcards
- **Flashcard Management**: View, edit, delete flashcards

### For Students

- **Available Documents**: Browse documents with flashcards
- **Flashcard Browser**: View and filter available flashcards
- **Study Interface**: Interactive flashcard study with timing and scoring
- **Progress Tracking**: View study history and performance

- **Leaderboard**: View rankings and export options

## 6. Getting Started

Here's a suggested approach to building the frontend:

1. **Choose a Framework**: React is recommended for its component-based architecture
2. **Setup Authentication**: Implement login/registration first
3. **Create User Dashboard**: Different views for teachers/students
4. **Implement Document Management**: For teachers
5. **Build Flashcard Study System**: For students
6. **Add Leaderboard**: Show rankings and progress

### Suggested Folder Structure (React example)

```
src/
├── api/              # API service modules
├── components/       # Reusable UI components
├── context/          # React context (auth, etc.)
├── pages/            # Main page components
│   ├── auth/         # Login/Registration
│   ├── teacher/      # Teacher-specific pages
│   └── student/      # Student-specific pages
├── utils/            # Helper functions
└── App.js            # Main application
```

## 7. Working with AI Assistance

Since you might be using AI tools like Claude to help with development, here are some tips:

1. **Start with Architecture**: Ask AI to help design the component structure
2. **Component Creation**: Request specific components with clear requirements
3. **API Integration**: Ask for code examples to connect to specific endpoints
4. **State Management**: Get help implementing user authentication state
5. **Styling**: Request CSS/styling assistance for components
6. **Error Handling**: Get advice on handling API errors and edge cases

When working with Claude:

- Provide specific requirements for components

- Ask for code explanations when needed
- Request step-by-step guides for complex functionality
- Break down large tasks into smaller requests

## Example Requests for AI

Here are some example prompts you might use:

- "Create a React component for user registration with teacher/student toggle"
- "Help me implement token-based authentication with React context"
- "Design a flashcard study interface with flipping cards and scoring"
- "Write code to upload PDF files and handle the API response"
- "Create a leaderboard component that sorts by score and shows rank"

## Need Help?

If you have any questions about the backend or how to integrate with it, feel free to reach out! Good luck building the frontend, and I'm looking forward to seeing what you create.

---

*Note: This guide was created to help you understand the backend and get started with frontend development. Feel free to adapt your approach based on your preferences and experience.*