EduCards Backend Requirements Specification

1. Project Overview

EduCards is an educational platform that leverages artificial intelligence to generate flashcards from uploaded PDF documents. The system enables teachers to upload educational materials and automatically generates study flashcards to help students learn, while creating a competitive environment with a leaderboard system.

2. Technical Requirements

2.1 Technology Stack

- **Programming Language**: Python 3.9+
- **Web Framework**: Django 4.2+
- **REST API**: Django REST Framework 3.14+
- **Database**: PostgreSQL 14+
- **AI Integration**: OpenAI API (GPT-4 or GPT-3.5-Turbo)
- **Containerization**: Docker and Docker Compose (optional)
- **Version Control**: Git

2.2 Required Packages

Django>=4.2.0,<5.0.0
djangorestframework>=3.14.0
psycopg2-binary>=2.9.5
Pillow>=9.0.0
PyPDF2>=3.0.0
openai>=1.0.0
python-dotenv>=1.0.0
drf-yasg>=1.21.0
django-cors-headers>=4.0.0
django-filter>=23.0
xlsxwriter>=3.0.0
celery>=5.3.0
redis>=4.5.0
gunicorn>=21.0.0
whitenoise>=6.4.0

2.3 Hosting Requirements

- **Web Server**: Nginx or Apache
- **Application Server**: Gunicorn or uWSGI
- **Database**: PostgreSQL 14+
- **Python Runtime**: Python 3.9+
- **Storage**: Sufficient for document storage (depends on expected usage)
- **Memory**: Sufficient for handling concurrent users (depends on expected traffic)
- **HTTPS**: Secure communication via SSL/TLS (provided by Azure/AWS hosting)

3. Functional Requirements

3.1 User Management

- **User Registration**: Support registration for both teachers and students
- **Authentication**: Token-based authentication system
- **User Roles**: Distinct roles for teachers and students with different permissions
- **Profile Management**: Allow users to update profile information and preferences

3.2 Document Management

- **Upload System**: Allow teachers to upload PDF documents
- **Document Storage**: Secure storage of uploaded documents
- **Document Browsing**: List available documents with filtering options
- **Access Control**: Teachers can make documents public or private
- **Document Management**: Allow teachers to update and delete their documents

3.3 AI Integration

- **Text Extraction**: Extract text content from uploaded PDF documents
- **Flashcard Generation**: Use OpenAI API to generate flashcards from document content
- **Generation Parameters**: Allow customization of quantity and difficulty of flashcards
- **API Usage Tracking**: Track and log OpenAI API calls for monitoring purposes

3.4 Flashcard System

- **Flashcard Creation**: Generate question-answer pairs from document content
- **Flashcard Sets**: Group flashcards into sets for organized studying
- **Difficulty Levels**: Support different difficulty levels (easy, medium, hard)
- **Customization**: Allow teachers to edit AI-generated flashcards
- **Search & Filter**: Enable searching and filtering flashcards by various criteria

3.5 Study Session Management

- **Session Tracking**: Record student study sessions with start and end times
- **Performance Metrics**: Track correct/incorrect answers and time spent
- **Progress Tracking**: Monitor student progress across multiple sessions
- **Score Calculation**: Calculate scores based on accuracy and completion time
- **Statistics**: Generate comprehensive statistics for each study session

3.6 Leaderboard System

- **Ranking Algorithm**: Calculate student rankings based on performance metrics
- **Global Leaderboard**: Display top-performing students across the platform
- **Course-specific Leaderboards**: Display rankings within specific courses/subjects
- **Filtering Options**: Allow filtering the leaderboard by time period or subject
- **Export Functionality**: Support exporting leaderboard data in multiple formats (CSV, JSON, Excel)

4. API Endpoints

The backend will expose a RESTful API with the following endpoint groups:

4.1 Authentication Endpoints

- `POST /api/accounts/register/`: Register a new user
- `POST /api/accounts/login/`: Authenticate and receive token
- `GET /api/accounts/profile/`: Retrieve user profile
- `PUT /api/accounts/profile/update/`: Update user profile

4.2 Document Endpoints

- `GET /api/documents/`: List accessible documents
- `POST /api/documents/upload/`: Upload a new document
- `GET /api/documents/<id>/`: Retrieve document details
- `DELETE /api/documents/<id>/delete/`: Delete a document
- `POST /api/documents/<id>/process/`: Process document with AI

4.3 Flashcard Endpoints

- `GET /api/flashcards/`: List available flashcards
- `GET /api/flashcards/<id>/`: Retrieve flashcard details
- `GET /api/flashcards/sets/`: List flashcard sets
- `POST /api/flashcards/sets/create/`: Create a new flashcard set
- `GET /api/flashcards/sets/<id>/`: Retrieve flashcard set details
- `GET /api/flashcards/sets/<id>/cards/`: Get cards in a set

4.4 Study Session Endpoints

- `GET /api/study/sessions/`: List user's study sessions
- `POST /api/study/sessions/start/`: Start a new study session
- `GET /api/study/sessions/<id>/`: Retrieve session details
- `POST /api/study/sessions/<id>/end/`: End a study session
- `POST /api/study/sessions/<id>/record/`: Record an answer
- `GET /api/study/statistics/`: Get user statistics

4.5 Leaderboard Endpoints

- `GET /api/leaderboard/`: Get the current leaderboard
- `GET /api/leaderboard/export/`: Export leaderboard data
- `GET /api/leaderboard/user/<id>/`: Get user rank information
- `POST /api/leaderboard/update/`: Recalculate leaderboard (admin only)

5. Database Schema

The database will include the following core tables:

5.1 User Table

- User authentication and profile information
- Extended with teacher/student role fields

5.2 Document Table

- Document metadata (title, description)
- File path and upload information
- Teacher reference (foreign key to User)

5.3 Flashcard Table

- Question and answer content
- Difficulty level
- Document reference (foreign key to Document)
- AI-generated flag

5.4 FlashcardSet Table

- Set name and description
- Creator reference (foreign key to User)

- Many-to-many relationship with Flashcard

5.5 StudySession Table

- Student reference (foreign key to User)
- Start time and end time
- Score and performance metrics
- FlashcardSet reference

5.6 SessionDetail Table

- Session reference (foreign key to StudySession)
- Flashcard reference (foreign key to Flashcard)
- Answer correctness and time taken

5.7 Leaderboard Table

- Student reference (foreign key to User)
- Performance metrics and ranking
- Last update timestamp

5.8 AIProcessingLog Table

- Document reference (foreign key to Document)
- Processing timestamp and duration
- API tokens used and success status

6. Security Requirements

- **Authentication**: Token-based authentication for all API endpoints
- **Authorization**: Role-based access control (teachers vs. students)
- **Data Protection**: Secure storage of user credentials and uploaded documents
- **API Security**: Secure handling of OpenAI API keys
- **Input Validation**: Thorough validation of all user inputs
- **Error Handling**: Secure error responses that don't expose sensitive information

7. Performance Requirements

- **Response Time**: API response time under 500ms for most operations
- **Scalability**: Support for at least 500 concurrent users
- **File Upload**: Support for PDF documents up to 20MB
- **AI Processing**: Complete flashcard generation within 30 seconds for average documents

- **Data Export**: Generate and download exports within 5 seconds

8. Environment Configuration

The application should support different environment configurations:

- **Development**: Local development setup with debug information
- **Testing**: Isolated environment for running automated tests
- **Staging**: Pre-production environment for final testing
- **Production**: Optimized for performance and security

Environment variables should be used for sensitive configuration (database credentials, API keys, etc.)

9. Deployment Requirements

- **Containerization**: Docker setup for consistent deployment
- **Environment Variables**: Configuration via environment variables
- **Database Migrations**: Automated database schema migrations
- **Static Files**: Proper handling of static files
- **Media Storage**: Secure storage for user-uploaded files
- **Logging**: Comprehensive logging system for monitoring and debugging

10. Testing Requirements

- **Unit Tests**: Test coverage for individual components
- **Integration Tests**: Tests for API endpoints and service interactions
- **Performance Tests**: Benchmark critical operations
- **Security Tests**: Verify authentication and authorization mechanisms

11. Documentation Requirements

- **API Documentation**: Comprehensive documentation of all API endpoints
- **Code Documentation**: Docstrings and comments for code maintainability
- **Setup Guide**: Instructions for setting up the development environment
- **Deployment Guide**: Instructions for deploying to production
- **User Manual**: Documentation for system administrators and end users

12. Maintenance Requirements

- **Logging**: Comprehensive logging for debugging and monitoring

- **Backup**: Regular database backup procedures
- **Updates**: Mechanism for applying updates and patches
- **Monitoring**: Performance and error monitoring
- **Scalability**: Ability to scale with increasing user load

---

Date: March 23, 2025 Version: 1.0