

Study Buddy Documentation

Table of Contents

Overview	3
Architecture	3
Features	3
Prerequisites	3
Installation	4
Configuration	5
API Documentation	5
File Structure	6
User Manual	7
Troubleshooting	7
Performance Optimization	8
Deployment	8
Contributing	9
License	9
Support	9
Shortcomings/Wishlist	10

Overview

The Flashcard Generator is a full-stack web application that creates AI-powered flashcards on any subject. Users can input a topic and receive automatically generated question-and-answer flashcards to aid in learning and memorization.

Architecture

Frontend (React)

- **Framework:** React with hooks (useState, useEffect)
- **Styling:** CSS with dark/light mode support
- **Features:** Interactive flashcard flipping, navigation, theme toggling

Backend (Flask)

- **Framework:** Flask with CORS support
- **AI Integration:** Ollama local LLM (Gemma3 model)
- **API:** RESTful endpoint for flashcard generation

AI Model

- **Service:** Ollama (local LLM hosting)
- **Default Model:** Gemma3 (configurable)
- **Alternative Models:** Llama3, Mistral, etc.

Features

-  **AI-Powered Generation:** Create flashcards using local LLM
-  **Dark/Light Mode:** Automatic theme detection with manual toggle
-  **Interactive Cards:** Click-to-flip flashcard interface
-  **Navigation:** Previous/Next card controls
-  **Customizable:** Adjustable number of cards (1-20)
-  **Error Handling:** Comprehensive error messages and loading states
-  **Local Storage:** Theme preference persistence

Prerequisites

Before running this application, ensure you have:

1. **Node.js** (v14 or higher)
2. **Python** (v3.8 or higher)
3. **Ollama** installed and running locally
4. **Gemma3 model** (or preferred model) pulled in Ollama

Installation

1. Ollama Setup

```
bash
# Install Ollama (visit https://ollama.ai for installation)
# Pull the Gemma3 model
ollama pull gemma3

# Start Ollama service
ollama serve
```

2. Backend Setup

```
bash
# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate

# Install dependencies
pip install flask flask-cors requests

# Run Flask server
python app.py
```

The backend will start on <http://localhost:5000>

3. Frontend Setup

```
bash
```

```
# Install dependencies
npm install
# Start React development server

npm start
```

The frontend will start on <http://localhost:3000>

Configuration

Backend Configuration

In `app.py`, you can modify:

```
python
OLLAMA_URL = "http://localhost:11434/api/generate" # Ollama API endpoint
MODEL_NAME = "gemma3" # AI model to use
```

Available Models: You can switch to other models like `llama3`, `mistral`, `codellama`, etc. Make sure to pull the model first:

```
bash
ollama pull llama3
```

Frontend Configuration

The React app automatically connects to the Flask backend at <http://localhost:5000>. If you change the backend port, update the fetch URL in the React code.

API Documentation

Generate Flashcards Endpoint

URL: `POST /api/generate-flashcards`

Request Body:

```
json
{
  "subject": "Ancient Rome",
  "numCards": 5
}
```

```
}
```

Response (Success):

```
json
```

```
{
```

```
  "flashcards": [
```

```
    {
```

```
      "question": "Who was the first Roman Emperor?",
```

```
      "answer": "Augustus (formerly Octavian) was the first Roman Emperor, ruling from 27 BC to 14 AD."
```

```
    },
```

```
    {
```

```
      "question": "What year did the Western Roman Empire fall?",
```

```
      "answer": "The Western Roman Empire fell in 476 AD when Germanic chieftain Odoacer deposed Emperor Romulus Augustulus."
```

```
    }
```

```
  ]
```

```
}
```

Response (Error):

```
json
```

```
{
```

```
  "error": "Subject is required"
```

```
}
```

Status Codes:

- 200: Success
- 400: Bad Request (missing subject)
- 500: Server Error (AI generation failed)

File Structure

```
flashcard-generator/
```

```
|— frontend/
```

```
| |— src/
```

```
| | |— App.js # Main React component
```

```
| | |— App.css # Styling
```

```
| | |— index.js # React entry point
```

```
| |— public/
```

```
| └─ package.json
| └─ backend/
|   └─ app.py # Flask server
|   └─ requirements.txt # Python dependencies
└─ README.md
```

User Manual

Creating Flashcards

1. **Enter Subject:** Type any topic (e.g., "Quantum Physics", "Spanish Vocabulary", "World War II")
2. **Set Card Count:** Use the slider to select 1-20 cards
3. **Generate:** Click "Generate Flashcards" button
4. **Study:** Navigate through cards using Previous/Next buttons
5. **Flip Cards:** Click on any card to reveal the answer

Theme Switching

- **Automatic:** App detects system dark/light mode preference
- **Manual:** Click the sun/moon icon to toggle themes
- **Persistent:** Theme choice is saved in browser storage

Troubleshooting

Common Issues

1. "Connection refused" errors

- Ensure Ollama is running: `ollama serve`
- Check if Flask backend is running on port 5000
- Verify firewall isn't blocking local connections

2. "Model not found" errors

- Pull the required model: `ollama pull gemma3`
- Verify model name matches in `app.py`

3. "No flashcards generated" errors

- Try a more specific subject

- Check Ollama logs for AI generation issues
- Ensure the model supports English (or your preferred language)

4. JSON parsing errors

- The AI sometimes returns malformed JSON
- The backend includes cleanup code for common formatting issues
- Try regenerating or using a different model

Debug Mode

Enable debug logging in Flask:

```
python
if __name__ == '__main__':
    app.run(debug=True, port=5000, host='0.0.0.0')
```

Performance Optimization

Backend Optimizations

1. **Model Caching:** Ollama keeps models in memory after first use
2. **Connection Pooling:** Consider implementing request pooling for high traffic
3. **Async Processing:** For better scalability, consider async Flask (Quart)

Frontend Optimizations

1. **Lazy Loading:** Cards are generated on-demand
2. **Local Caching:** Consider caching generated flashcards
3. **Debounced Requests:** Prevent multiple simultaneous API calls

Deployment

Local Network Deployment

To make the app accessible on your local network:

1. **Backend:** Change Flask host to `0.0.0.0`
2. **Frontend:** Update API URL to your machine's IP address
3. **Firewall:** Ensure ports 3000 and 5000 are accessible

Production Deployment

For production deployment, consider:

1. **Backend:** Use Gunicorn or uWSGI instead of Flask dev server
2. **Frontend:** Build React app (`npm run build`) and serve with nginx
3. **Security:** Add API authentication, rate limiting, input validation
4. **Monitoring:** Add logging, health checks, error tracking

Contributing

Development Setup

1. Fork the repository
2. Create a feature branch
3. Make changes with appropriate tests
4. Submit a pull request

Code Style

- **Python:** Follow PEP 8 guidelines
- **JavaScript:** Use ES6+ features, functional components
- **CSS:** Use consistent naming conventions

License

This project is open-source and available under the MIT License.

Support

For issues and questions:

1. Check the troubleshooting section
2. Review Ollama documentation
3. Create an issue in the project repository

Shortcomings/Wishlist

Shortcomings

- Currently unable to save flashcards unless having used the export .txt feature.
- The AI is ran locally which spends more resources on the user's device.
- No mobile version support.
- No login/logout or account creation feature.
- Unable to return feedback to the LLM on the quality of the flashcards given.

Wishlist

- The creation of an account along with a login/logout system where each account can have its flashcard sets stored.
- Scaling up the project in a way that allows the program to run the LLM non locally through AWS servers.
- Mobile version support that does not need the setup of the web-application
- A flashcard rating system that allows the LLM to get feedback on the quality of the flashcards generated.