Flashcard Web Application - Project Report

Project Overview

This project is a full-stack web application for creating and studying flashcards. The application features both frontend and backend components, allowing users to create, edit, delete, and study flashcards with an intuitive interface. The application has been enhanced with additional features based on user feedback.

Technical Architecture

Backend (Server-side)

- **Framework**: Flask (Python)
- **Database**: SQLite (for easy setup and portability)
- **API Design**: RESTful API endpoints for CRUD operations
- **Data Models**: Flashcard model with question, answer, category, learning status, and metadata
- **Database Migration**: Flask-Migrate for database schema changes

Frontend (Client-side)

- **Technologies**: HTML, CSS, JavaScript (vanilla)
- **Design**: Responsive layout that works on both desktop and mobile devices
- **Interface**: Modern UI with navigation bar and elegant homepage
- **Features**: Card creation, editing, deletion, and enhanced study mode

Key Features

1. **Elegant Homepage with Navigation**
2. Welcome section with app introduction
3. Feature highlights
4. User statistics (total cards, categories, cards reviewed)
5. Navigation bar for easy access to different sections

6. **Flashcard Management**

7. Create new flashcards with questions and answers
8. Organize flashcards by categories
9. Edit existing flashcards
10. Delete unwanted flashcards

11. Filter flashcards by category (with improved filtering)

12. **Enhanced Study Mode**

13. **Study by Category**: Select specific categories to study
14. **Learning Status Tracking**: Mark cards as "Know" or "Still Learning"
15. Flip cards to reveal answers
16. Navigate through cards sequentially
17. Track review progress

18. Randomized card order for effective studying

19. **Responsive Design**

20. Adapts to different screen sizes
21. Mobile-friendly interface
22. Consistent experience across devices

Technical Implementation Details

Backend Implementation

The backend is built with Flask and provides a RESTful API for the frontend to interact with. Key components include:

1. **Data Model (`src/models/flashcard.py`)**
2. Defines the structure for flashcard data
3. Includes fields for question, answer, category, difficulty, and learning status
4. Tracks review count and last reviewed timestamp

5. Provides methods for serialization (to_dict)

6. **API Routes (`src/routes/flashcard.py`)**

7. GET `/api/flashcards` - Retrieve all flashcards or filter by category
8. GET `/api/flashcards/<id>` - Retrieve a specific flashcard
9. POST `/api/flashcards` - Create a new flashcard

10. PUT `/api/flashcards/<id>` - Update an existing flashcard (including learning status)
11. DELETE `/api/flashcards/<id>` - Delete a flashcard
12. POST `/api/flashcards/<id>/review` - Mark a flashcard as reviewed
13. **Main Application (`src/main.py`)**
14. Configures the Flask application
15. Sets up database connection (SQLite)
16. Initializes Flask-Migrate for database migrations
17. Registers API blueprints
18. Serves static files

Frontend Implementation

The frontend is built with vanilla HTML, CSS, and JavaScript, providing a clean and intuitive user interface:

1. **HTML Structure (`src/static/index.html`)**
2. Defines the page layout with navigation
3. Implements multiple "pages" within a single HTML file
4. Includes modals for card creation and confirmation dialogs
5. Features a study setup section for category selection
6. Includes "Know" and "Still Learning" buttons in study mode
7. **CSS Styling (`src/static/css/styles.css`)**
8. Implements responsive design
9. Defines animations for card flipping
10. Creates a modern, clean aesthetic
11. Ensures consistent styling across components
12. Styles for learning status indicators
13. **JavaScript Logic (`src/static/js/app.js`)**
14. Handles user interactions
15. Manages API communication
16. Implements enhanced study mode functionality
17. Controls navigation between pages
18. Updates UI based on data changes
19. Provides robust category filtering

20. Tracks and updates learning status
21. Ensures accurate review count updates

Recent Enhancements

The application has been updated with the following improvements based on user feedback:

1. **Study Mode by Category**
2. Added a dedicated study setup screen
3. Implemented category selection before starting study mode
4. Allows users to focus on specific subjects or topics
5. Maintains the same category dropdown options as the main view
6. **Learning Status Tracking**
7. Added "Know" and "Still Learning" buttons in study mode
8. Implemented backend storage of learning status
9. Visual indicators for learning status on cards
10. Persists learning status between sessions
11. **Fixed Review Count Bug**
12. Corrected the issue where the "Cards Reviewed" count wasn't updating
13. Implemented real-time stats updates after reviewing cards
14. Ensured proper synchronization between frontend and backend
15. Added immediate UI feedback when cards are reviewed
16. **Improved User Experience**
17. Enhanced navigation between application sections
18. Added visual feedback for user actions
19. Improved error handling and notifications
20. Optimized performance for smoother interactions

Setup and Usage Instructions

Prerequisites

- Python 3.6 or higher
- pip (Python package manager)

Installation Steps

1. Extract the project files from the zip archive
2. Create a virtual environment: `python -m venv venv`
3. Activate the virtual environment:
4. On Windows: `venv\Scripts\activate`
5. On macOS/Linux: `source venv/bin/activate`
6. Install dependencies: `pip install -r requirements.txt`

Running the Application

1. Start the Flask server: `python -m src.main`
2. Access the application in a web browser at: `http://localhost:5000`

Using the Application

1. **Creating Flashcards**:
2. Navigate to "My Cards" section
3. Click "New Card" button
4. Fill in question, answer, and optional category
5. Click "Save"
6. **Studying Flashcards**:
7. Navigate to "Study" section
8. Select a category to study (or "All Categories")
9. Click "Start Studying"
10. Click on cards to flip between question and answer
11. After viewing the answer, mark your knowledge:
    - Click "Know" if you knew the answer
    - Click "Still Learning" if you need more practice
12. Use "Previous" and "Next" buttons to navigate
13. **Managing Flashcards**:
14. Use edit and delete buttons on each card
15. Filter cards by category using the dropdown
16. Select "All Categories" to view all flashcards

Project Structure

```
flashcard_app/
├── requirements.txt       # Python dependencies
├── src/                   # Main application directory
│   ├── __init__.py        # Python package marker
│   ├── main.py            # Application entry point
│   ├── models/            # Database models
│   │   ├── __init__.py
│   │   ├── flashcard.py   # Flashcard data model
│   │   └── user.py        # User model and database setup
│   ├── routes/            # API endpoints
│   │   ├── __init__.py
│   │   ├── flashcard.py   # Flashcard API routes
│   │   └── user.py        # User API routes
│   └── static/            # Frontend assets
│       ├── css/
│       │   └── styles.css # Application styles
│       ├── js/
│       │   └── app.js     # Frontend JavaScript
│       └── index.html     # Main HTML entry point
└── README.md              # Project documentation
```

Conclusion

This flashcard web application successfully implements both frontend and backend components as required. The application provides a complete solution for creating and studying flashcards with an intuitive user interface and robust backend functionality. The recent enhancements have significantly improved the user experience, particularly for studying and tracking learning progress.

The code is organized, well-documented, and follows best practices for web development. The application can be accessed directly through the HTML entry point (`src/static/index.html`) when the Flask server is running, meeting the requirement for direct access to the webpage.