**Flashcard Quiz Game - Project Requirements**

**1. Project Overview**

A quiz-style flashcard application where users select a deck (topic/category) and answer multiple-choice questions. The system tracks scores and provides immediate feedback. Built with Node.js, Express, MySQL backend and React.js frontend.

**2. Technical Stack**

**Backend**

* **Runtime**: Node.js (v16+)
* **Framework**: Express.js
* **Database**: MySQL (v8.0+)
* **ORM/Query Builder**: mysql2 or Sequelize
* **Authentication**: JWT (optional for user sessions)
* **Validation**: express-validator or joi
* **Environment**: dotenv for configuration

**Frontend**

* **Framework**: React.js (v18+)
* **State Management**: React Hooks (useState, useEffect) or Context API
* **HTTP Client**: Axios or Fetch API
* **Styling**: CSS Modules, Tailwind CSS, or styled-components
* **Routing**: React Router (if multi-page)

**Development Tools**

* **API Testing**: Postman or Thunder Client
* **Version Control**: Git/GitHub
* **Package Manager**: npm or yarn

**3. Database Schema**

**Tables**

**decks**

CREATE TABLE decks (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

description TEXT,

category VARCHAR(50),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

**cards**

CREATE TABLE cards (

id INT PRIMARY KEY AUTO\_INCREMENT,

deck\_id INT NOT NULL,

question TEXT NOT NULL,

option\_a VARCHAR(255) NOT NULL,

option\_b VARCHAR(255) NOT NULL,

option\_c VARCHAR(255) NOT NULL,

option\_d VARCHAR(255) NOT NULL,

correct\_answer ENUM('A', 'B', 'C', 'D') NOT NULL,

explanation TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (deck\_id) REFERENCES decks(id) ON DELETE CASCADE

);

**users (Optional - for user tracking)**

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**game\_sessions (Optional - for score tracking)**

CREATE TABLE game\_sessions (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

deck\_id INT NOT NULL,

score INT NOT NULL,

total\_questions INT NOT NULL,

completed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE SET NULL,

FOREIGN KEY (deck\_id) REFERENCES decks(id) ON DELETE CASCADE

);

**4. Backend API Endpoints**

**Deck Management**

**GET** /api/decks

* Description: Retrieve all available decks
* Response: Array of deck objects with id, name, description, category, card count

**GET** /api/decks/:id

* Description: Get specific deck details
* Response: Deck object with metadata

**POST** /api/decks (Admin)

* Description: Create a new deck
* Body: { name, description, category }
* Response: Created deck object

**PUT** /api/decks/:id (Admin)

* Description: Update deck information
* Body: { name, description, category }
* Response: Updated deck object

**DELETE** /api/decks/:id (Admin)

* Description: Delete a deck and its cards
* Response: Success message

**Card Management**

**GET** /api/decks/:deckId/cards

* Description: Get all cards for a specific deck (without correct answers for quiz mode)
* Response: Array of card objects with question and options

**GET** /api/cards/:id

* Description: Get specific card details
* Response: Card object

**POST** /api/decks/:deckId/cards (Admin)

* Description: Add a new card to a deck
* Body: { question, option\_a, option\_b, option\_c, option\_d, correct\_answer, explanation }
* Response: Created card object

**PUT** /api/cards/:id (Admin)

* Description: Update a card
* Body: Same as POST
* Response: Updated card object

**DELETE** /api/cards/:id (Admin)

* Description: Delete a card
* Response: Success message

**Quiz/Game Flow**

**POST** /api/quiz/start

* Description: Start a quiz session
* Body: { deck\_id, user\_id (optional) }
* Response: { session\_id, deck\_info, first\_card }

**POST** /api/quiz/answer

* Description: Submit an answer and get result
* Body: { card\_id, selected\_answer }
* Response: { is\_correct, correct\_answer, explanation, next\_card }

**POST** /api/quiz/submit

* Description: Submit final score
* Body: { deck\_id, score, total\_questions, user\_id (optional) }
* Response: { session\_id, final\_score, percentage }

**GET** /api/quiz/leaderboard/:deckId (Optional)

* Description: Get high scores for a deck
* Response: Array of top scores

**User Management (Optional)**

**POST** /api/users/register

* Description: Create new user account
* Body: { username, email, password }
* Response: User object (without password)

**POST** /api/users/login

* Description: Authenticate user
* Body: { email, password }
* Response: { token, user }

**GET** /api/users/:id/history

* Description: Get user's quiz history
* Response: Array of past game sessions

**5. Frontend Requirements**

**Pages/Views**

**1. Home/Deck Selection Page**

* Display all available decks as cards/tiles
* Show deck name, description, category, and number of cards
* Search/filter by category
* Click to start quiz with selected deck

**2. Quiz Game Page**

* Display current question number (e.g., "Question 3 of 10")
* Show question text clearly
* Display 4 multiple choice options (A, B, C, D)
* Highlight selected answer
* "Submit Answer" button
* Show correct/incorrect feedback after submission
* Display explanation (if available)
* "Next Question" button
* Progress bar or indicator

**3. Results Page**

* Display final score (e.g., "8/10")
* Show percentage
* List all questions with user's answers and correct answers
* "Play Again" button
* "Choose Another Deck" button
* Option to save score (if user system implemented)

**4. Admin Dashboard (Optional)**

* Manage decks (CRUD operations)
* Manage cards within decks (CRUD operations)
* View statistics

**Key Features**

1. **Responsive Design**: Works on desktop, tablet, and mobile
2. **Loading States**: Show spinners during API calls
3. **Error Handling**: Display user-friendly error messages
4. **Validation**: Ensure answer is selected before submission
5. **Animations**: Smooth transitions between questions
6. **Accessibility**: Keyboard navigation, ARIA labels

**6. Core Features & User Stories**

**MVP (Minimum Viable Product)**

**As a user, I can:**

1. View all available quiz decks
2. Select a deck to start a quiz
3. Answer multiple-choice questions one at a time
4. Receive immediate feedback (correct/incorrect)
5. See my final score at the end
6. Review all questions and answers after completion
7. Restart the same quiz or choose a different deck

**Phase 2 Features**

**As a user, I can:**

1. Create an account and log in
2. Save my quiz scores
3. View my quiz history
4. See a leaderboard for each deck
5. Filter decks by category
6. See statistics (average score, best score, etc.)

**As an admin, I can:**

1. Create, edit, and delete decks
2. Add, edit, and delete cards within decks
3. View user statistics

**7. Optional Enhancements**

1. **Timer Mode**: Add time limits per question or entire quiz
2. **Difficulty Levels**: Tag cards with difficulty (Easy/Medium/Hard)
3. **Study Mode**: Show answer immediately without scoring
4. **Random Order**: Shuffle questions and answer options
5. **Hints System**: Allow users to use hints (reduce points)
6. **Favorites**: Let users bookmark favorite decks
7. **Social Features**: Share scores on social media
8. **Progress Tracking**: Track learning progress over time
9. **Multi-language Support**: i18n implementation
10. **Dark Mode**: Theme switcher