Web X Mini Project Report

Topic : Flashcards app for learning

Name: Mahi Jodhani(21)

Class:D15A

Contents

---

1. Introduction

Learning and retaining information effectively is essential in both academic and professional contexts. Traditional study methods often lack interactivity and personalization, leading to reduced engagement and efficiency. This project introduces a smart Flashcards App for Learning with a modern, user-friendly web interface and intelligent learning features. Users can create and manage flashcard decks directly from the homepage, with full control to add, update, or delete decks. Upon selecting a deck, users can enter a dedicated page where they can create and review individual flashcards.

The app offers two dynamic study modes:

- **Study Mode,** where users can practice by viewing the question and toggling the answer visibility manually to test their recall.
- **Quiz Mode,** where users must input their answers to each question, receiving instant feedback on whether their response is correct.

The frontend is built with a responsive framework, while the backend is developed using Flask. All data, including decks and flashcards, is securely stored and managed using MongoDB, ensuring scalability and real-time data handling.

Key Features :

- **Deck Management :** Create, update, and delete flashcard decks.
- **Flashcard Creation :** Add, edit, and remove cards within each deck.
- **Study Mode :** View questions with a toggle to show/hide answers.
- **Quiz Mode :** Input answers and get instant correct/incorrect feedback.
- **Responsive Frontend :** Clean, modern UI built with React.
- **Flask Backend :** Handles API and logic efficiently.
- **MongoDB Storage :** Secure, scalable data storage for decks and cards.

2. Problem Statement

- Traditional study methods (e.g., paper flashcards) lack interactivity and flexibility.
- Existing digital flashcard tools often have limited customization and poor user experience.
- Learners need an efficient way to organize study material into decks and cards.
- There is a lack of engaging modes to actively recall and test knowledge.
- No centralized system to create, manage, and review flashcards with instant feedback.
- Need for a responsive and user-friendly web app that supports real-time data management.

3. Objectives of Project

- Develop a web-based flashcard application for effective learning and revision.
- Allow users to create, update, and delete decks to organize flashcards by topic.
- Enable users to add, edit, and remove flashcards within each deck.
- Provide an interactive Study Mode with toggleable answers for self-paced learning.
- Implement a Quiz Mode to test knowledge with user-input answers and instant feedback.
- Ensure a responsive and user-friendly interface using modern frontend technologies (e.g., React, Flask).
- Use Flask for backend services to handle logic and communication with the database.
- Store all user data securely and efficiently using MongoDB.
- Lay the groundwork for future enhancements like user authentication and progress tracking.

4. Requirements & Technology Stack

Functional Requirements :

- Users can create, update, and delete flashcard decks.
- Users can add, edit, and delete flashcards within a deck.
- Two learning modes:
    - Study Mode – View question and toggle answer.
    - Quiz Mode – Input answer and receive instant correctness feedback.
- Data should be stored and retrieved securely and efficiently.
- Responsive and intuitive UI for seamless interaction.

Non-Functional Requirements :

- User-friendly and responsive design.
- Fast performance and smooth navigation.
- Scalable database structure for handling large sets of flashcards.
- Code modularity for future enhancements (e.g., login system, analytics).

Technology Stack

Frontend

- React.js – For building a dynamic and responsive UI.
- HTML5 & CSS3 – For layout and styling.
- JavaScript (ES6+) – For interactive features and state management.

Backend

- Flask (Python) – For API development and handling backend logic.
- Flask-Restful – Optional: For building RESTful APIs more efficiently.

Database

- MongoDB – NoSQL database to store decks, flashcards, and related data.

5. Proposed System

The proposed system is a web-based Flashcards Learning Application designed to improve the learning experience through interactive and organized study methods. It allows users to create and manage custom decks of flashcards, and engage with the content using two distinct modes: Study Mode and Quiz Mode.

Key Components

1. **User Interface (Frontend)**
    - Built using React.js for a responsive and dynamic experience.
    - Homepage to view, create, update, and delete decks.
    - Deck page to manage and view flashcards.
    - Easy navigation between study and quiz modes.

2. **Flashcard Functionalities**
    - **Study Mode:** Displays one question at a time with a toggle to show/hide the answer.
    - **Quiz Mode:** Prompts the user to input an answer and gives real-time feedback.

3. **Backend System**
    - Developed with Flask to handle API endpoints for CRUD operations.
    - Secure and modular REST API design for interaction with frontend.

4. **Database**
    - MongoDB stores user data, decks, and flashcards in a flexible, scalable structure.

Benefits of the Proposed System :

- Enhanced learning with active recall and self-testing.
- Easy management of flashcards and decks for organized study.
- Responsive and intuitive UI accessible across devices.
- Scalable and extensible architecture for future features like login, analytics, or spaced repetition.

6. Proposed Design

The design of the Flashcards App is centered around simplicity, scalability, and user engagement. It follows a modular, component-based structure on the frontend and a RESTful architecture on the backend.

1. Frontend Design

- Homepage (Dashboard)
    - Displays all existing decks.
    - Options to Create, Update, and Delete decks.

- Deck Page
    - Lists flashcards under the selected deck.
    - Toggle between:
        - Create Mode – Add/Edit/Delete flashcards.
        - Study Mode – Show one flashcard at a time with "Show/Hide Answer" button.
        - Quiz Mode – Show question, user inputs answer, feedback is given.

- Components Breakdown
    - DeckList, DeckCard, Flashcard, StudyView, QuizView, FormModal, etc.

2. Backend Design

- RESTful API Endpoints
    - GET /decks – Retrieve all decks.
    - POST /decks – Create a new deck.
    - PUT /decks/<id> – Update deck name.
    - DELETE /decks/<id> – Delete a deck.
    - GET /decks/<id>/cards – Get all cards in a deck.
    - POST /decks/<id>/cards – Add a new flashcard.
    - PUT /cards/<card_id> – Edit a flashcard.
    - DELETE /cards/<card_id> – Remove a flashcard.

- Modular Flask App Structure
    - routes/, models/, controllers/, app.py

3. Database Design

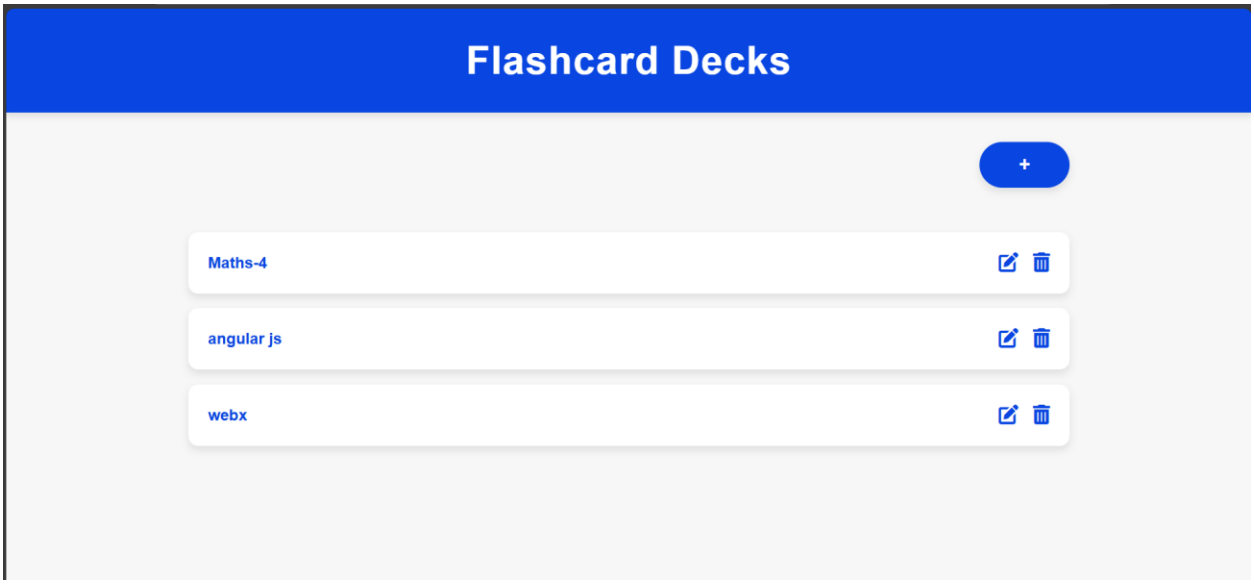
- Collections
  - decks
    - Fields: _id, name, created_at, updated_at
  - flashcards
    - Fields: _id, deck_id, question, answer, created_at, updated_at

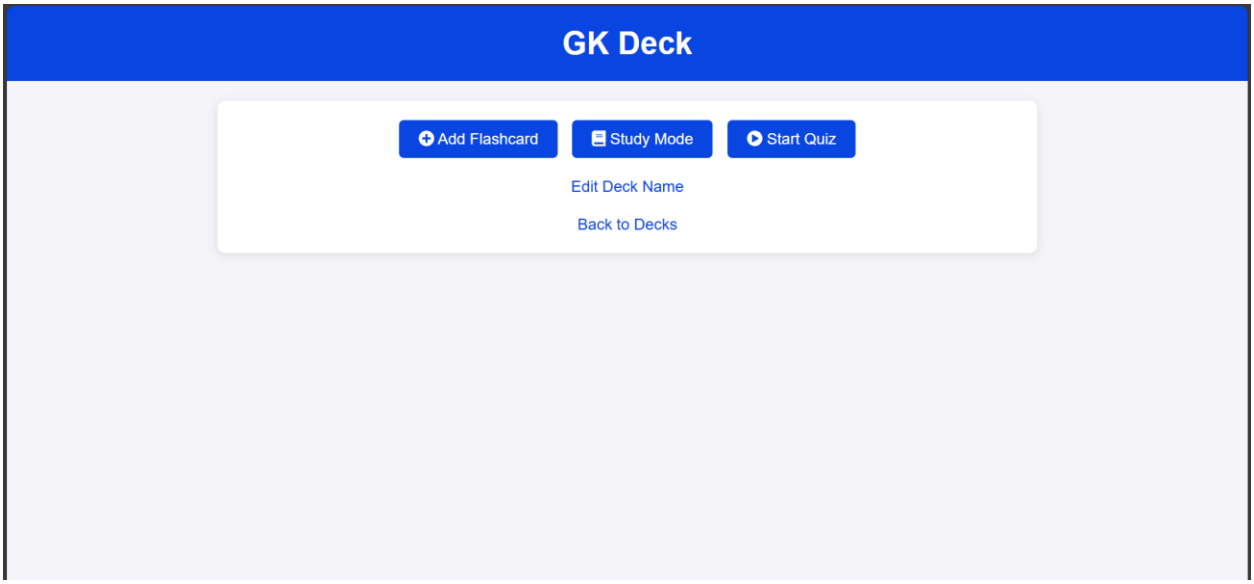
- Relationships
  - One-to-many: One deck → many flashcards

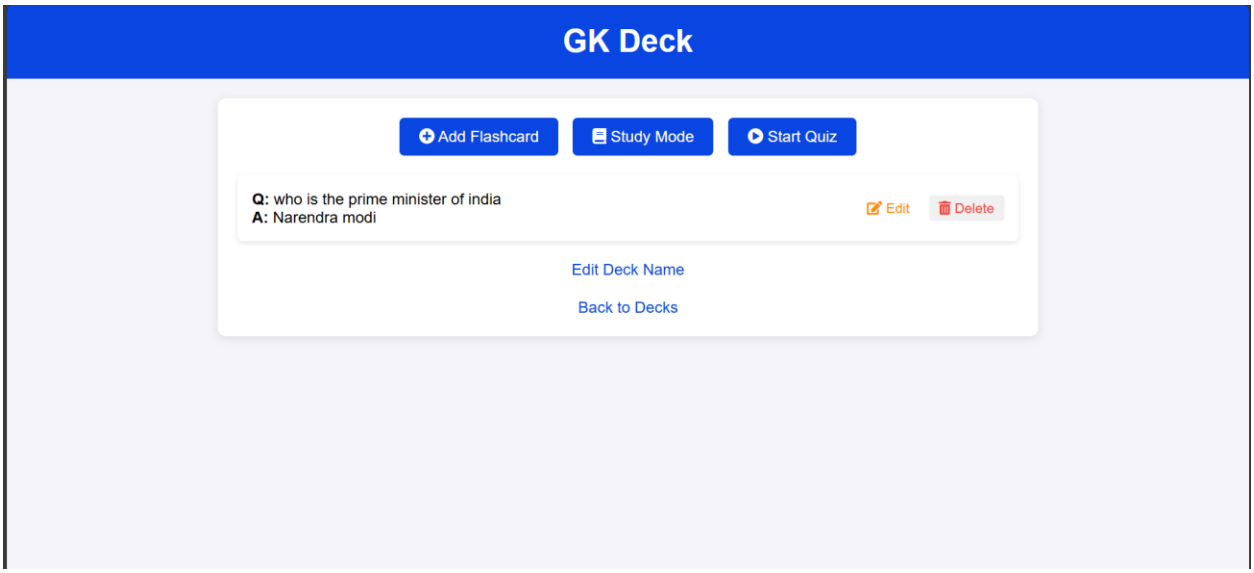
4. User Interaction Flow

1. User opens the app and sees the dashboard (all decks).
2. User selects or creates a deck.
3. Inside a deck, user can:
   - Add/edit flashcards (Create Mode)
   - Review flashcards (Study Mode)
   - Test themselves (Quiz Mode)

7. Implementation





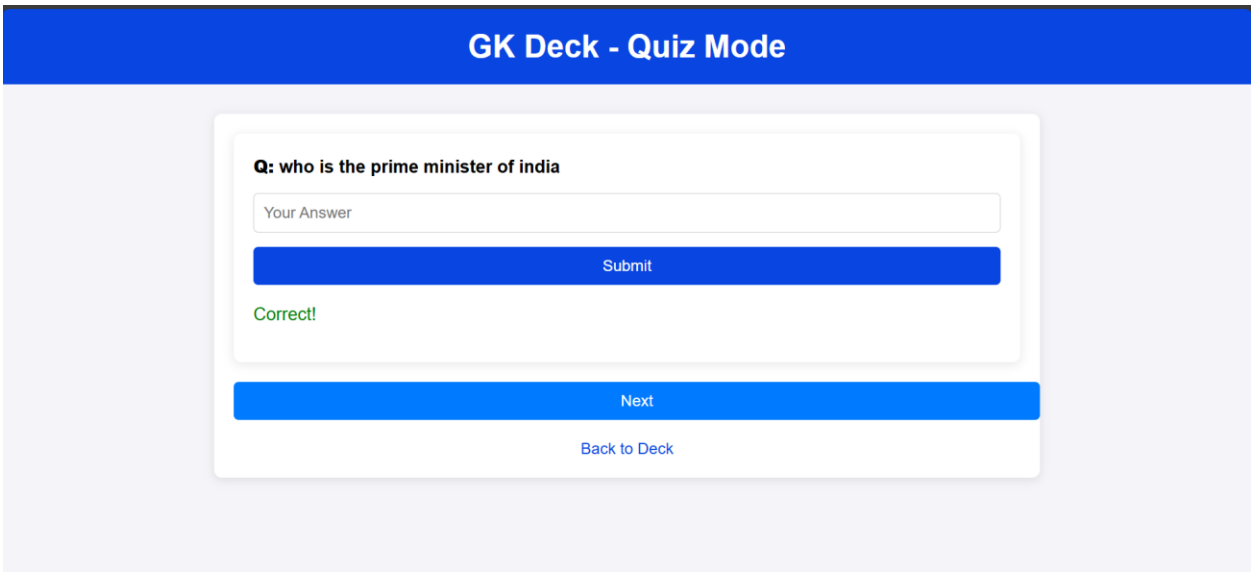


Study Mode for GK

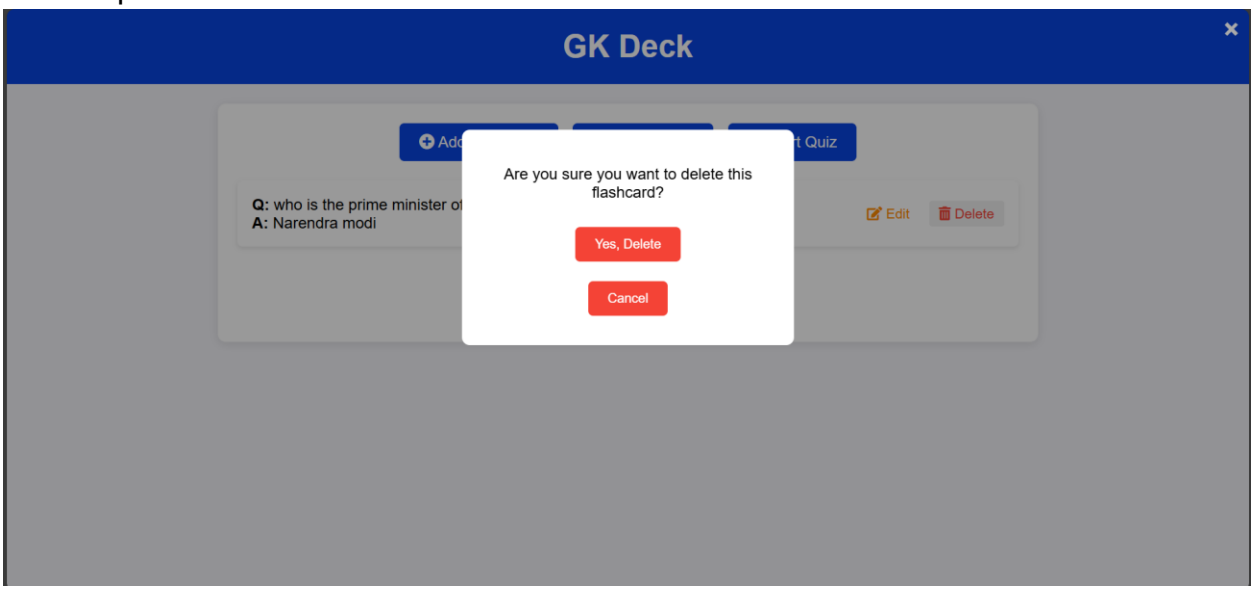
Q: who is the prime minister of india

Show Answer

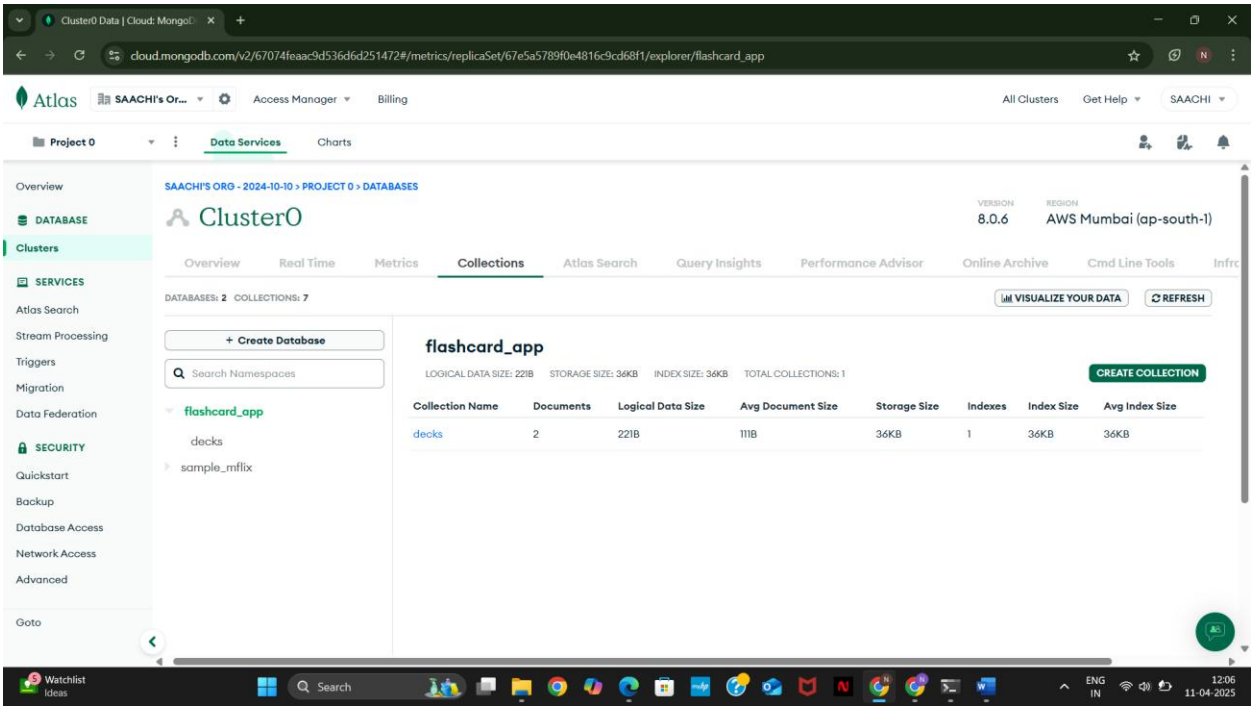
Back to Deck

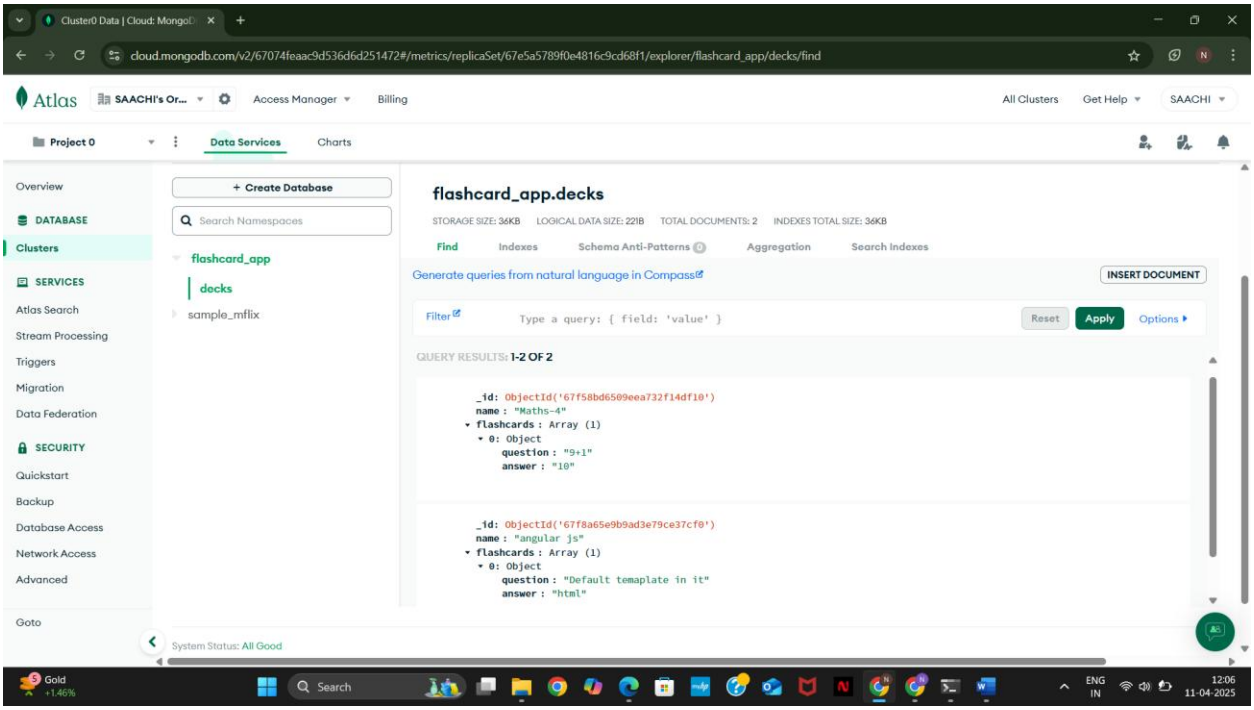


delete operation for deck



MONGO DB : DATA





8. Result Analysis

The Flashcards App successfully meets its objectives by providing a user-friendly, interactive platform for effective learning. Users were able to seamlessly create, update, and delete decks and flashcards, with the app performing smoothly across devices thanks to its responsive React-based frontend. The core features, including the Study Mode and

Quiz Mode, functioned as intended, offering users the ability to toggle answers and receive immediate feedback, which enhanced their learning experience. The backend, built with Flask and powered by MongoDB, efficiently handled CRUD operations and data management, ensuring quick and reliable responses. The app's performance was optimal, with minimal load times and smooth transitions between actions.

9. Conclusion

In conclusion, the Flashcards App effectively enhances learning through its interactive Study and Quiz modes, promoting active recall and self-assessment. Its robust architecture—built with React, Flask, and MongoDB ensures smooth performance and efficient data handling. The app improves engagement with instant feedback and offers strong potential for future upgrades, such as user authentication and detailed progress tracking.