IU – International University

Module: DLBCSPJW01 – Project: Java and Web Development

Module Director: Prof. Dr. Holger Klus

Bachelor: Computer Science

PROJECT:

Develop a web application

Student: Tunschi Alexandru-Radu

Matriculation number: 92013549

E-mail: tunschi.alexandruradu@iubh.de

Phase 3 – Finalization phase

Simple Learning Flashcard Application (SLFA) is a MERN stack web application which users can help themselves learn and memorize things with sets of flashcards.

Concept and Content:

The core concept of this project is to create a user-friendly web application that allows users to learn sets of information efficiently. The main features of the application include user registration, login functionality, set creation, set management (edit and delete), taking quizzes, and creating/editing flashcards.

1. User Registration and Login:
   - Users start by navigating to the login page.
   - If they don't have an account, they can click on the "Register" button to create one.
   - During registration, the frontend sends an API request to the backend.
   - The backend checks if the provided email already exists in the MongoDB database.
   - If the email exists, an error response is sent to the frontend.
   - If the email doesn't exist, a new account is created, and user data is stored in MongoDB.
   - After registration, users can log in using the same process.
   - Users can change the password of their account, by clicking the “Change password” button. They are required to input the old password and twice the new password. The equivalence of both fields of the new password is verified also, if the old password is correct. If the old password is correct and both fields of the new password field have the same information, the new password will be saved.
   - When user login with email and password the api will send the data to backend, the backend will check first if the email does not exist and will send an error message to frontend, else will compare the password with database password - if the password is wrong it will send and error to frontend, else will send the success response and on the frontend user will successfully login.
2. Set Creation and Management:
   - After logging in, users can create sets of information.
   - These sets are saved in the MongoDB database.
   - For each set, users have the option to edit the title of the set, or delete it.
   - These actions trigger corresponding API calls on the backend, ensuring data consistency.
   - On click delete, the api will call against the set id, and the set will be deleted from the mongodb
   - on click edit, the api will calls and the title of the set will be updated in the mongodb
3. Quiz Feature:
   - Users can take quizzes associated with specific sets.
   - Each flashcard consists of one correct answer and three incorrect ones.
   - Quiz results are stored in MongoDB against the user's profile.

4. Flashcard Creation and Editing:
   - Users can create and edit flashcards within a set.
   - Each flashcard consists of one question and four answers, with one being correct.
   - Users can easily manage and review flashcards.

Technical Approach:

The technical approach for this project leverages Next.js for the frontend and Express.js with Node.js for the backend.

Frontend:

- Next.js was chosen for the frontend due to its server-side rendering capabilities, which enhance the user experience and SEO.
- React components were used to create the user interface, and routing is handled with Next.js's built-in routing system.
- Axios API is utilized to make HTTP requests to the backend APIs.

Backend:

- Express.js is used as the backend framework for its simplicity and robustness.
- Node.js serves as the runtime environment.
- MongoDB is used as the database to store user information, sets, quiz results, and flashcards.
- API routes are defined in Express to handle user registration, login, set creation/editing, quiz-taking, and flashcard management.
- Mongoose, an Object Data Modeling (ODM) library for MongoDB, is employed to simplify database interactions.

Lessons learned:

- To take in consideration ways for the user to exit or return to a previous screen. After designing the app according to the draft that I've made, I discovered through testing that I didn't implement ways for the user to return Home, and the user would be stuck in the Registration or Change password screens. I've added a “Home” button to the set view, and redirected the user to the login page after a successful registration or change of password.
- The importance of verifying the spelling of the folders vs. what is written in the code. I've spent 3 hours looking for an error that allowed me to create a set, put in flashcards that were displayed at the creation, quiz the set with those flashcards, but when entering again in the set, no flashcards were displayed. This was created by incorrectly creating the directory “[flashcard]”, instead of “[flashCard]” as it was in the code.

Things that can be improved: personalization for users' profiles, personalization of the sets (as in the numbers of the questions in a set, multiple answers) ratings, reviews, sharing sets between users, adding gamification elements (badges, points, rewards), social networking integration etc.