Flash Carma

A web application made by Adrian West, Brandon Sitz, and Josh Moore

Austin Peay State University

CSCI 4805

Table of Contents

[Abstract 2](#_Toc112358602)

[Description 2](#_Toc112358603)

[Features List 4](#_Toc112358604)

[Technology 5](#_Toc112358605)

[Server Information 6](#_Toc112358606)

[Data Sources 6](#_Toc112358607)

[Team Members Backgrounds 6](#_Toc112358608)

[Dependencies, Limitations, and Risks 7](#_Toc112358609)

[Timeline 8](#_Toc112358610)

# Abstract

Studying can be a daunting task no matter the scholastic level. Traditional index cards can also add unnecessary expenses for students. Flash Carma is a study app that seeks to solve both of these problems by utilizing digital flashcards and helping the user study their custom material in an entertaining and enjoyable manner.

The user is able to create multiple flashcards with their custom questions and answers that they can then store in a study deck. When they want to study, they simply pick a study deck and are taken through each card in the deck. Does the user need to study multiple subjects? No problem! They can create as many study decks as they want and review their subjects time and time again. As the user studies, they will gain points for each correct answer they give and be given a score at the end of the study session to help them assess their proficiency with that study deck. They can even start a streak by giving successive correct answers! With Flash Carma, the user can reduce their study expenses and enjoy their time studying.

# Description

The project goal is to create a web-based flashcard application with an optimized User Interface (UI) to aid students in their study sessions. The React JavaScript library, Tailwind, and Chakra will function as the client-side UI. The Firebase Application Programming Interface (API), Node.js, and Express framework will function as server-side tools. When the application is loaded up for the first time, the user will be prompted to log in to their existing account. If the user does not have an existing account, they will have the option to create one on the user registration page. The user will then be directed to the main page of the application. Once at the applications main page, a navigation bar will allow the user to create a new study deck consisting of flashcards, access their profile information, search for pre-existing study decks, and start a new study session with an existing study deck.

If the user selects the option to create a new study deck, they will be redirected to a new page to create them. The user can input text onto the front and back of the flashcard and save it to the current study deck. They will also have the option to delete any existing flashcards from the study deck. If the user clicks the profile button, they are redirected to a new page where they will have the option to edit any of their profile information. If the user selects the “Study Session” button, they will be prompted to select a study deck and be redirected to a new page containing the flashcards for the chosen study deck. The study session consists of a single flashcard with an animation to display the answer, a score for the current study session, a button to switch to the next flashcard, and two buttons the user can select depending on if their answer is correct or incorrect. Presumably, the user will answer the question verbally or mentally, so we will not need to know the user’s answer or create any input area for said answer. Incorporating a scoring system allows the user to keep track of their progress with a particular study deck and will enable them to compete with themselves for a better score. Once the study session is complete, the results are displayed, and the user has the option to restart the current study session, select another study deck, or exit to the main page.

Flashcards are a popular study tool that utilizes active recall and helps users test and improve their memory when learning new information. While reading from a textbook and attempting to memorize large amounts of information at once may be a useful practice for some, flashcards can greatly enhance a user’s studying experience by forcing them to retrieve their knowledge suddenly and from scratch.

# Features List

The must have features of this web application in no particular order are as follows:

* User login / authentication
* User registration
* User profile information
* Create study deck
* Create flashcards
* Delete flashcards
* Begin a new study session
* Flashcard animation to display correct answer
* Button to determine if the user’s answer is correct or incorrect
* Scoring system
* Add points for each correct answer
* Display results of study session
* Store study decks in the database for each user
* Search study decks
* The area for creating flash cards and the study area could be two separate areas that can be navigated to by using a navigation bar.

Features that we would like to have if there is time are as follows:

* Questions database
* User high score
* Light / dark mode
* A correct answer “streak” may help encourage users' mid-study session and utilize many humans’ natural competitiveness. Possibly keep a record of the user’s highest streak.
* Including a “hint” feature could prove beneficial for user satisfaction. This mechanic would allow the user to add a hint for any question when creating it. To encourage the user to not totally depend on the hints, limiting the number of hints used per study session may be beneficial.

Features that we would like to add after the semester is over are as follows:

* Mobile friendly
* User leaderboard
* Multiple choice quizzes
* Randomize multiple choice questions
* Sign in with 3rd party organization
* Print flashcards

# Technology

The technology we will be utilizing is:

* Visual Studio Code for the IDE as we believe it is the best IDE for developing web applications.
* React (JavaScript library) as everyone in the group is mostly familiar with it and it is a tried-and-true library for web development.
* Font Awesome (Font Library) because it allows to choose from a multitude of different fonts.
* ChakraUI (UI library for React) because it provides ease of development for the UI.
* Tailwind (CSS framework) because it allows us to speed up the development process by allowing us to input CSS directly into the HTML.
* Node.js because it is a requirement for server-side development.
* Express (Node.js framework) as it allows us to manage servers and routes.
* Firebase API which will be used as the backend server for database, user creation, user authentication, and hosting/deployment of the web application.
* Chrome/Chromium based browsers as those will be the browsers we deploy the application to.
* Discord for communicating between group mates.

# Server Information

The server components we will be using are the Firebase API, Node,js, and the Express framework.

# Data Sources

If we need to utilize a preset database, we will use https://opentdb.com/.

# Team Members Backgrounds

* Brandon Sitz
  + Experience with the React JavaScript library. Experience with user authentication in the Firebase API. Minimal experience with databases. No experience with server-side operations.
* Adrian West
  + Minimal experience in web development. Not familiar with React, Express, or Firebase.
* Josh Moore
  + Have studied Computer Science at Austin Peay State University (APSU) for more than four years, including many classes focused on web design and development. Experienced with HTML, CSS, JavaScript, PHP, some database tools, and client-side and server-side operations.

# Dependencies, Limitations, and Risks

Dependencies, limitations, and risks include:

* We reach the limit of the Firebase free tier
* Adrian does not have experience with developing react based applications, express, or database management with firebase.
* Brandon has minimal experience with databases and no experience with server-side operations.

Solutions to these problems include:

* Migrate to a server on Linode.
* Adrian can learn how most of this stuff works pertaining towards this project.
* Brandon can learn more about databases and server-side operations with the Firebase API and Node.js

# Timeline

|  |  |  |
| --- | --- | --- |
| Week | Task Name | Assigned To |
| Aug 29 – Sep 4 | Create UML Diagram | Adrian |
|  | General design of the UI | Brandon |
|  | Create React project and upload it to GitHub. | Josh |
| Sep 5 – 11 | Node.js setup | Adrian |
|  | Express setup | Josh |
|  | Firebase API setup | Brandon |
| Sep 12 - 18 | Home page: layout/welcome message | Adrian |
|  | Home page: project name | Brandon |
|  | Navigation bar: buttons | Adrian |
| Sep 19 - 25 | Navigation bar: links | Josh |
|  | Search bar | Josh |
|  | \*Dark mode | Brandon |
| Sep 26 – Oct 6 | Study deck page: create flash card | Adrian |
|  | Study deck page: delete flash card | Josh |
|  | Store study deck in database | Brandon |
| Oct 3 – 9 | Study session page: flashcard | Adrian |
|  | Study session page: flashcard animation | Brandon |
|  | Scoring system | Adrian |
| Oct 10 – 16 | \*High score system | Brandon |
|  | \*Correct answer streak | Adrian |
|  | \*Hint system | Josh |
| Oct 17 – 23 | Registration page | Brandon |
|  | Login page | Adrian |
|  | Store user in database | Josh |
| Oct 24 – 30 | User authentication | Brandon |
|  | Profile page | Adrian |
|  | Editing profile information and updating profile | Brandon |
| Oct 31 – Nov 6 | Finish critical operations of app | All |
|  | Any remaining features | All |
|  | Begin poster | All |
| Nov 7 – 13 | Prepare Presentation | All |
|  | Finalize poster | All |
|  | Any remaining features | All |
| Nov 14 – 20 | Project demo | All |
|  | Any remaining features | All |
|  | Begin final report | All |
| Nov 21 – 27 | Any Remaining Maintenance/Debugging for the App’s Necessary Operations | All |
|  | Work on final report | All |
|  | Any Remaining Maintenance/Debugging for the App’s Necessary Operations | All |
| Nov 28 – Dec 6 | Final report submissions | All |
|  | Project demo | All |
|  | Project finalization | All |