Flash Carma

by Adrian West, Brandon Sitz,

Joseph Ramos-Garcia, and Josh Moore

## Introduction

What is Flash Carma? To answer this question, we must first acknowledge the problems with traditional flashcards. Traditional flashcards are exhaustive and time-consuming to create. Obtaining the materials needed to create them also adds an unnecessary expense to the individual who needs them. Furthermore, while traditional flashcards do stimulate a user’s memory, they offer very little- if anything at all- in terms of entertainment value. Flash Carma is the solution to all of these problems. Flash Carma is a web-based application that utilizes digital flashcards. These digital flashcards offer the same level of customizability as traditional flashcards; however, instead of relying on the user’s writing speed, they rely on a user’s typing speed. With traditional flashcards, the user has a limited amount of materials and is forced to purchase more when their supply runs out. However, with Flash Carma, there is no additional cost for the user to create more flashcards, and creating them is as simple as clicking a button. Flash Carma also provides an abundance of entertaining features that make studying an enjoyable experience.

## Technology

Firebase technologies were used to provide us with user authentication, a database to store the user’s data, and a means to host the site itself. Since Visual Studio Code provided numerous extensions that benefited the production of Flash Carma, we decided to use it as our coding environment. One of the most useful extensions was the Chakra UI extension, which provided numerous resources for creating the pages of Flash Carma. React.js was used as a framework for the application, and Node JS was used as a package manager. Our logo was created using the GNU Image Manipulator, and the streak icon was created using Microsoft Word. Discord was our means of communication, while GitHub was used primarily for version control as well as a means for us to access each other’s developments and divide the work.

Firebase was challenging to use throughout the development of the application. The documentation provided needs to be more adequate, especially when using the React.js framework. We created a separate application specifically for testing out Firebase functionality and watched many tutorials on integrating it with React. After working with Firebase for a while, we started to get the hang of how it worked and were able to eliminate the number of errors we were getting when we first started. We initially proposed Firebase because it came with many of the features that we would need to build a full-stack application, and in the end, it worked out for us. Firebase worked out as a small application with a small number of users. However, it may not be viable as a large application due to the costs associated with a larger user base.

Many group members did not have any experience with React or Chakra UI. Initially, getting up to speed with React was difficult, but over time we were able to learn the fundamentals and main concepts. Compared to vanilla JavaScript, React reduced the amount of code we needed to write and came with many useful libraries. The group didn’t need much time to get accustomed to using Chakra UI. It came with many useful components that made designing and styling the application's front end easier.

## Design

Upon entering the URL for the application, the user is met with the Login screen. The Login screen prompts the user for an email address and password and provides respective text boxes to enter the information. Once the user’s information has been entered, they can click the “Log In” button and an authentication request will be sent to firebase to verify that they are a registered user. The user will be directed to the Homepage -provided that the information entered corresponds with an already existing profile. If the information entered does not correspond to an existing profile, the user will be notified that either their email or password is incorrect. If the user is new and does not yet have a profile, they can click the “Sign Up” link and will be taken to the Sign-Up page. The Sign-Up page prompts the user to enter a username, email address, and password, which will all be used to create the user’s new profile. Each of these prompts is accompanied by its own textbox for the user to enter the information. The user is also prompted to enter their password a second time in an additional textbox, which will be compared to the first entry of their password to ensure the characters making up the password are what the user intended. Once the information has been entered, the user can click the “Sign Up” button to create their profile. After the user’s profile has been created, they will be directed to the Login page, where they can now log in to their existing profile.

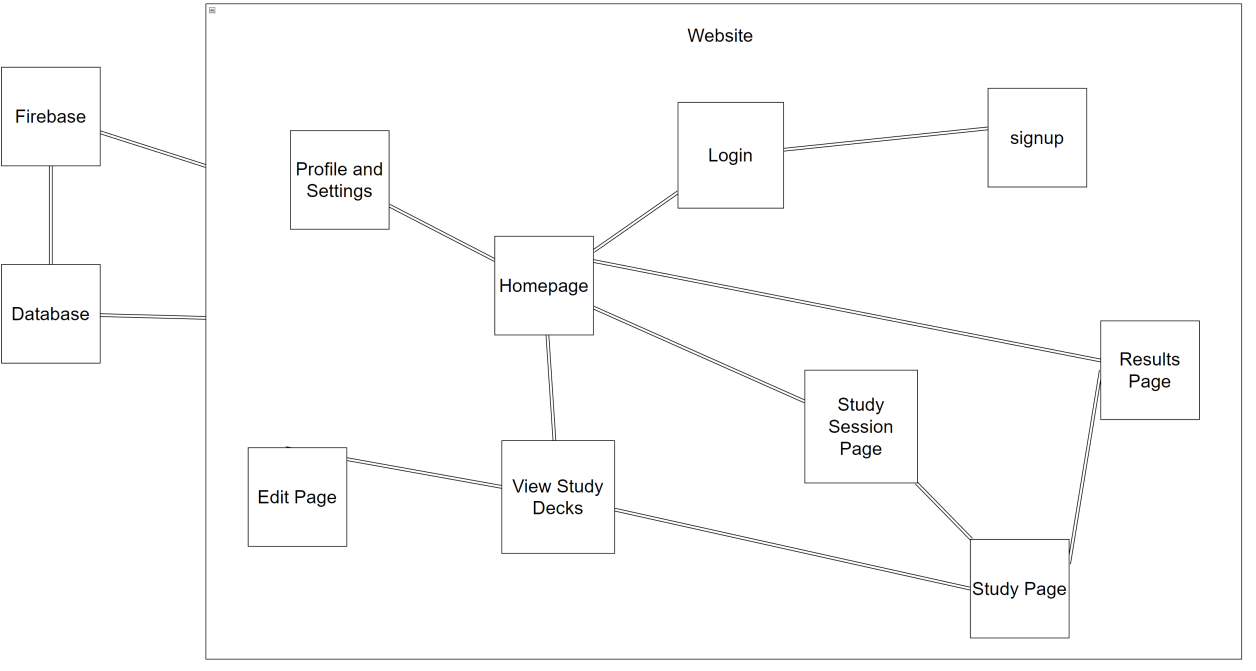
Once they reach the Homepage, which is shown in *Figure 4*, the user gains access to the navigation bar. The navigation bar provides links to various pages within the application. The first link is represented by the Flash Carma logo and will direct the user to the Homepage when clicked. The second link, labeled “View Study Decks”, will direct the user to the View Study Decks page, where they can view all of their existing study decks in a grid layout and create new study decks. The third link is labeled “Study Session” and will direct the user to a screen that presents the user with all of their existing study decks- in a grid layout that is similar to the View Study Decks page- allowing them to choose which one they wish to study. The next two items in the navigation bar are on the far right of the user’s screen. The first item is a message that says “Hello, [User’s username]”. The second is a profile icon that, if clicked, displays a drop-down menu. The first item in the drop-down menu is a link to the Settings page labeled “Settings”. The second item is a link labeled “Logout” that will log the user out of their profile. The final item in the drop-down menu is a button that switches the user’s theme between Light Mode and Dark Mode. The Settings page displays the user’s current username and provides fields should the user wish to change their username or password. If it has been a significant amount of time since the user entered their email and password, they may have to reauthenticate. To assist in this matter, there is a reauthenticate button located on the Settings page that directs them to the Login page and allows them to enter their email address and password. The navigation bar is available to the user at any time and is present on all pages with the exception of the Login and Sign-Up pages. As the user completes study sessions in Flash Carma, they are awarded points depending on the number of questions they answered correctly. On the Homepage, the user can see their cumulative score from all their past study sessions. Additionally, the Homepage displays the four most recently created study decks.

The View Study Decks page- shown in *Figure 5*- presents all the user’s existing study decks in a grid layout. The user’s study decks- organized in a way where the most recently created deck is first and the least recently created deck is last- will begin on the second element in the grid and continue to the final element in the grid. The first element in the grid is a square button- shaped like the study decks in the grid- with a “+” symbol on it. This button is used to create new study decks. Each row in the grid contains up to five of the user’s study decks with the exception of the first row, which contains the button to create a new study deck as well as up to four of the user’s study decks. When the button to create a study deck is clicked, the user will be presented with a pop-up that prompts the user to enter a name for the new study deck. After entering a name and clicking “Okay”, an empty study deck with the entered name will appear in the grid. If the user clicks on one of the study decks, a drop-down menu will display. The drop-down menu will give the user the option to “Edit”, “Study”, or “Delete” the selected deck. If the user selects the option to delete the study deck, they will be given a prompt asking if they wish to delete the chosen study deck. If the user selects the option to “Edit” the study deck, they will be redirected to the Edit page. The Edit page will allow the user to add, edit, and delete flashcards from the selected study deck. If the user wishes to edit one of their existing flashcards- such as one of the flashcards in their study deck containing a typing error or some circumstance that requires a change in the text on the flashcard- they may modify the text on the flashcard by clicking said text. Once clicked, the user is presented with the flashcard’s text in an editable text box. After the user has finished modifying the text, they simply click outside of the text box, and the changes to the flashcard will be saved. In addition, should the user wish to edit the title of their study deck, they can do so by clicking the text for the study deck’s title on the Edit page. Similar to the flashcards, the text for the study deck’s title will be presented in an editable text box. Once they have finished modifying the title, the user simply hits the “enter” key, and their changes to the title will be saved. If the user would like to add a new flashcard, they would simply need to enter the text for the front and back of the flashcard in the respective text boxes, which are presented towards the top of the Edit page, and click the button labeled “Create a new card” as can be seen in *Figure 6*. Once the button has been clicked, the flashcard will be created in the database and appear in the current study deck. If the user wishes to study a particular study deck, they can do so using either one of two methods. The first method the user can use is to navigate to the View Study Decks page, click on their preferred deck, and click the button labeled “Study” to be taken into a study session with that particular study deck. Alternatively, the user can click the “Study Session” link in the navigation bar, which will direct them to a page that presents all their currently existing study decks in a grid layout, and click on their preferred study deck to be taken into a study session with that particular deck.

The Study Session page, which is depicted in *Figure 7*, displays the first flashcard in the study deck to the user in the center of the page. If the user clicks on the flashcard, the text will switch between the text on the front of the flashcard and the text on the back of the flashcard. The user’s current score is displayed directly above the flashcard at any given time during the study session. A correct choice button that is labeled with a check mark and an incorrect choice button that is labeled with an “X” are displayed below the flashcard. If the user guessed correctly, they will select the correct choice button, and their score will be incremented. If the user guessed incorrectly, they will select the incorrect choice button, and their score will not change. Once either of these buttons are clicked, the next flashcard in the study deck will be displayed. Additionally, the number of the current flashcard out of the total number of flashcards is displayed towards the top of the page, above the title of the study deck. The user also has the option to end the current study session at any time by clicking the button labeled “End Session” at the top right of the page. If the user clicks this button, the study session will end, and the user will be directed to the Results page, which- as the name suggests- displays the results of their study session.

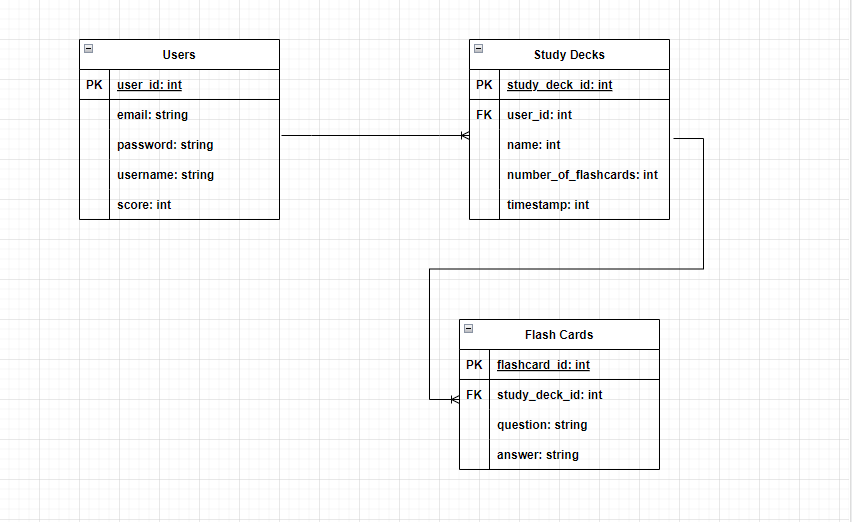
At the top of the Results page is the word “Results”, and beneath that the user is shown how much of the study deck they completed, what their correct answer streak was at the end of the study session, and their score for the study session. Just beneath that information are two buttons: one labeled “Replay” that will add the user’s score to their overall score on the Homepage and restart the study session, and one labeled “Home” that will direct the user to the Homepage. If at any point the user attempts to begin a study session with a study deck that has no flashcards in it, they will be directed to a page that displays a message explaining that there are no flashcards in their selected study deck. There is a button below, which links to the Edit page, so the user can fill in the empty deck.

## Block Diagram

****

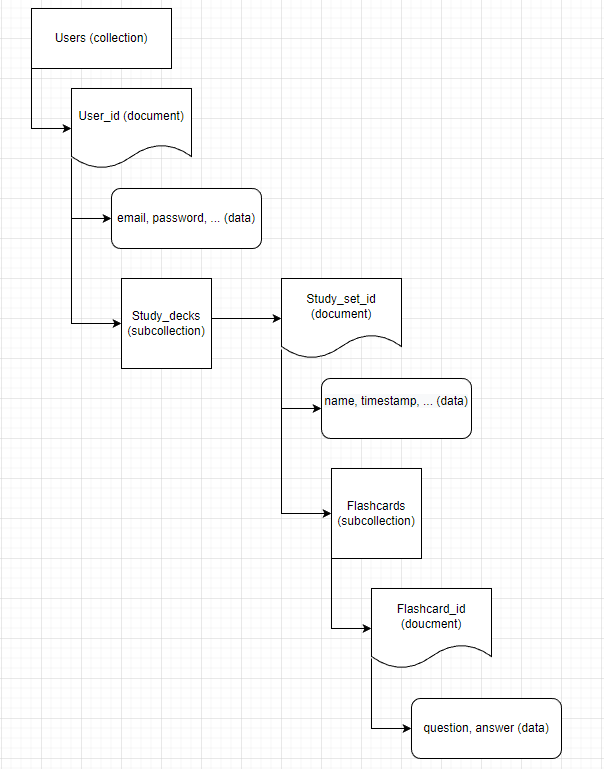
***Figure 1****: Flash Carma Block Diagram*

## Storage Documentation

Entity Relationship Diagram (ERD) 

***Figure 2****: Entity Relationship Diagram*

Data Model



***Figure 3****: Database Data Model*

## 

## Screen Captures

## 

## ***Figure 4:*** *Screen capture of the Flash Carma Homepage.*

## 

## 

## ***Figure 5:*** *Screen capture of the View Study Decks page.*

## 

## ***Figure 6:*** *Screen capture of the Edit page.*

## 

## ***Figure 7:*** *Screen capture of the Study Session page.*

## 

## How to build / deploy

* Clone the repository
  + Navigate to the GitHub repository at https://github.com/Brandon3522/flash\_carma
  + Select the button "Code" and copy the HTTPS link provided.
  + Open your favorite IDE (Integrated Developer Environment) and go to the source control.
  + Select the button to clone a repository and paste the HTTPS from GitHub into the input.
  + Select the folder that you would like to store the repository on your local device.
  + Next, you need to make sure that Node.js is installed on your machine.
  + If you do not have Node.js on your machine. Go to https://nodejs.org/en/ and follow the instructions to install the most recent version of Node.js.
  + Once Node.js is installed. Open your terminal and enter "npm install". Running this command will install all libraries and packages associated with the project.
* Create a firebase project and register your app
  + Navigate to the firebase console at https://console.firebase.google.com/.
  + Sign in to your google account.
  + Enter the project name.
  + Click through the prompts on the site.
  + Click "Create project" to create your project.
  + Once the process is complete, you will be taken to an overview page of your new Firebase project.
  + Click "Register your app" under the "Web" icon to set up a web-based application.
  + Enter your application’s nickname.
  + Click the checkmark to set up Firebase Hosting for the application.
  + Click "Register app".
  + Add Firebase SDK
    - Navigate back to your IDE terminal and run the command "npm install firebase".
    - Copy the provided code sample and create a new file called "firebase.js" in the source directory of the project.
    - Click the "Next" button.
  + Install Firebase CLI (Command Line Interface)
    - In the terminal run the command "npm install -g firebase tools".
    - Click the "Next" button.
* Firebase authentication
  + Navigate to the firebase console at https://console.firebase.google.com/ and select your project.
  + Click on "Build" and then "Authentication".
  + Click "Email/Password" and enable this feature.
  + Click "Save".
  + Firebase authentication is now ready.
* Firestore database
  + Click on "Build" and then "Firestore database".
  + Click "Create database".
  + A pop up will display. Click the toggle that says, "Start in test mode".
  + Select a location where your cloud firestore data will be stored.
  + Congrats! Your Firestore database is now ready.
* Deploy / hosting
  + Navigate to the terminal in your IDE.
  + Run the command "firebase login" to login into your Firebase account.
  + Run the command "firebase init" to initialize your application with Firebase.
  + Select the option for "hosting" in the firebase CLI.
  + Enter "build" as your public directory.
  + Select "yes" when prompted to configure the application as a single-page application.
  + Select "yes" or "no" when prompted to set up automatic builds and deploy with GitHub.
  + Your application is now initialized with Firebase.
  + Navigate back to the terminal and enter "npm run build" to create a production ready version of your application.
  + Finally, run "firebase deploy" in the terminal.
  + Awesome! Your application is now being hosted at the provided URL.

## Known Bugs

When navigating between certain pages in dark mode, a brief white flash appears for a split second. When the application is in light mode, the white flash is not visible due to the background color of index.html defaulting to white. After doing some research, we discovered the color mode switcher component, provided by Chakra UI, is the root of the issue. When the color mode switcher is toggled to dark mode, it fails to change the background color of index.html to the dark mode variation, which results in a brief white flash. A potential solution to the issue is to change the background color of index.html based on the color mode value of light or dark.

Another known issue is the responsiveness of the application. The nav bar, login, sign-up, and settings pages are all responsive when it comes to resizing the browser window but need to be optimized for mobile devices. Additionally, the rest of the pages need to be more responsive. A solution to this issue would be to add styling to the components based on the size of the user's screen or window. We would also restructure the layout of certain components depending on the user's screen or window size. Let's say the user navigates to the application's home page and resizes their browser window. Once the window reaches a specific size, we would change the layout of the study decks from a grid to a column layout so that the study decks do not extend past the viewable window.

## Future Works

Over time, there were plenty of ideas that we had, which we ended up needing to put off to the side, whether because of technical issues or time constraints. A big one on the navbar is the search bar, which would be a good thing to have for users with many different decks, as it would grow to be cumbersome to sift through 20+ decks. Similarly, we were also thinking of having a search feature for the editor, so the user can find a specific card by its question or answer. Furthermore, we were planning on having the decks have a toggle support between alphabetical and reverse-alphabetical display orders for the deck list, as it's cleaner and more consistent with how it's displayed. We also plan on adding a toggle to the settings page, which enables the user to be able to change how the study decks are displayed between the standard grid layout and a list layout.

Another aspect of the program we would like to develop is the account/settings information. We would like to add the option to have a forgotten password option, as there is still the possibility that the user could end up locked out of their account, and that functionality could help them out. Furthermore, having the user confirm password when changing their password is another aspect we considered for more account security. Furthermore, we plan to add an email verification to secure the account and a 3rd party sign in for more options for the user.

When it comes to studying the decks, we plan on having a rotation animation which happens whenever the user clicks on a card to see the other side. We also plan on adding a streak symbol, which appears behind the streak counter, and increases in intensity as the user increases streak. In the future, there will be an option labeled “Leitner system” that the user can toggle on and off from the Settings page. This option will assist the user in their studying schedule by pushing notifications to the user, reminding them to study certain amounts on certain days of the week. Furthermore, we would also like to establish a connection between accounts, where the user can access a page containing decks from other users, which they have enabled others to see. On this page, the user would be able to search for different topics, and also be able to access a user’s page, where any decks they’ve enabled for public viewing would be visible. They will be able to study a user’s deck from there.