Day 1

Bootcamper Pain Points

1. Created mind mindmaps to identify this week's core pain points of bootcampers.

Pain Points

Needs

User Stories

As a Bootcamper, I want a software tool that can help me stay organized and focused on my course, access my information and avoid feeling overwhelmed.

As a Bootcamper, I want a software tool that can connect me with personalized learning recommendations, so that I can improve the areas that I need to learned.

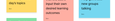
As a Bootcamper, I want a software tool that can track my progress and achievements, so that I can stay motivated and see how much I've learned.

As a Bootcamper, I want a software tool that can help me find study partners, so that I can get help and support when I need it.

As a Bootcamper, I want a software tool that can provide me with real-world coding challenges, so that I can apply what I've learned and get practical experience.

As a Bootcamper, I want a software tool that can help me set and track my goals, so that I can get the most out of the program and achieve my goals.

Project Ideas



User Personas

Problem Statement

1. Select a single problem and develop a problem statement based on past research.

Problem Statement

The goal of this research is to develop a software tool that addresses the challenge of knowledge retention and keeping up with the pace of learning experienced by people participating in coding bootcamps.

Despite the growing popularity of coding bootcamps as a means of learning programming skills, many participants struggle to retain knowledge and keep up with the fast-paced learning environment.

This leads to frustration, low self-confidence, and ultimately hinders their ability to successfully complete the program and achieve their learning goals.

Therefore, the problem to be addressed is: How can a software tool be developed to assist bootcamp participants in retaining knowledge and keep up with the pace of learning in coding bootcamps?

The findings of this research will contribute to improving the effectiveness and success rates of coding bootcamps, ultimately benefiting participants and the tech industry as a whole.

Ideation

1. Come up with creative solutions to the problem and map out your minimum viable product features (MVP).

Dreamer	Realist	Critic
1. A platform that let people exchange flashcards with peers. Ideally a gamifiable app	1. Web app that allows people to make their own flashcards. The only data to be deployed, we would need to store user data (the cards).	1. Time-consumption. Limit functionality for MVP.
2. Randomly generated coding challenges (codewars etc).	2. Accessibility would need to be developed to either import flashcards from existing tooling solutions or generate them (random).	2. How does this take into account people's age approach other ages?
3. User libraries can be uploaded and shared with other users.	3. Shared libraries can be made available for back end dev.	3. How is this app to be used? Desktop, mobile etc.
4.		4. Do the proposed features fully address the identified problem?

User Flow Diagram

MVP Feature Set - Primary

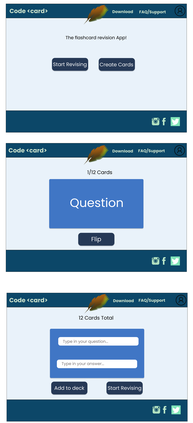
```
Body - Header - Menu

Logs
- Login - Not functional.
- Flashcards Library - Hardcoded with a few examples for demo
- Explore Flashcards - Not functional
- Settings - Not functional

Body - Main

Page - Create Flashcard
- Input Text - input question
- Input Text - input answer
- Button - Add to Flashcard Library

Page - Start Revising
- Flashcard generated on load (useEffect)
- Flashcard Flips on tap to reveal answer - JS Animation

(Flashcards could be generated with randomised colour/style settings)

Footer

- Socials - Simple links to social landings
- Copyright - use Date/time to set responsive copyright
```

MVP Feature Set - Secondary

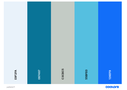
```
Body - Header - Menu

Logs
Budget Menu
- Login - Added
- Flashcard Library - Called from API
- Explore Flashcards - Called from API
- Settings - Simple settings for changing theme, font etc.

Body - Main

Page - Create Flashcard
- Input Text - input question
- Input Text - input answer
- Button - Add to Flashcard Library
- Button - Save cards from Library
- Button - Delete cards from Library
- Button - Back to home

Page - Start Revising
- Flashcard generated on load (useEffect)
- Flashcard Flips on tap to reveal answer - JS Animation
- Double tap leads home flashcard
- Back to home

(Flashcards could be generated with randomised colour/style settings)

Footer

- Socials - Simple links to social landings
- Copyright - use Date/time to set responsive copyright
```

Low-res Wireframe

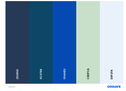
High-res Wireframe



Colour Scheme (old)

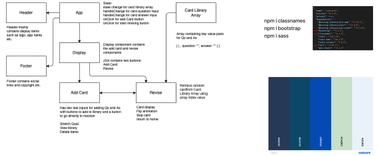


Colour Scheme Final



(Adheres to AAA standard)

Component Tree



Deprecated:
1. Header, Footer replaced with nav button, nav 3.1
2. Header, Footer no used in flip
3. Flip button replaced with 'Next Card' button
4. currently NO NEXT to move onto next card

npm i classnames
npm i bootstrap
npm i sass

#054A83 was discarded after trying it out

Things we need to include:
1. The sass-page (card file)
2. Add a card to the card library
3. which creates a new pushfor
4. New logo
5. made header/footer work

```
$color: #ffffff;
$bg-color: #000000;
$card-bg-color: #1a1a1a;
$card-text-color: #ffffff;
$card-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
$card-hover-shadow: 0 10px 15px rgba(0, 0, 0, 0.2);

// Mixins
@mixin flex($dir: 'row', $wrap: nowrap, $jc: flex-start, $ai: stretch, $ac: stretch) {
  display: flex;
  flex-direction: #{$dir};
  flex-wrap: #{$wrap};
  justify-content: #{$jc};
  align-items: #{$ai};
  align-content: #{$ac};
}

// Flashcard container
.flashcard-container {
  height: 100vh;
  @include flex(column, nowrap, center, center, stretch);
}

// Flashcard
.flashcard {
  width: 300px;
  height: 200px;
  perspective: 1000px;
  cursor: pointer;
  transition: transform 0.6s cubic-bezier(0.4, 0.0, 0.2, 1);
  transform-style: preserve-3d;
  &:hover {
    transform: scale(1.05);
    box-shadow: 0 10px 15px rgba(0, 0, 0, 0.2);
  }
  &.flipped {
    transform: rotateY(180deg);
  }
}

// Flip card inner
.flashcard-inner {
  position: relative;
  width: 100%;
  height: 100%;
  transition: transform 0.6s cubic-bezier(0.4, 0.0, 0.2, 1);
  transform-style: preserve-3d;
}

// Flip card sides
.flashcard-front, .flashcard-back {
  position: absolute;
  width: 100%;
  height: 100%;
  backface-visibility: hidden;
  -webkit-backface-visibility: hidden;
  @include flex(column, nowrap, center, center, stretch);
  padding: 20px;
  box-sizing: border-box;
  text-center;
  overflow: hidden;
  text-overflow: ellipsis;
  -webkit-line-clamp: 3;
  -webkit-box-orient: vertical;
}

// Flip card front
.flashcard-front {
  z-index: 2;
  transform: rotateY(0deg);
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

// Flip card back
.flashcard-back {
  z-index: 1;
  transform: rotateY(180deg);
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
```

styles.scss file above, essentially a second .scss file.
1. Mixins are reusable code blocks
2. require dependencies needed to make it work.
3. e.g. includes sprites/icons/ui elements/ui style sheets (.scss)
4. Mixins are like functions
5. referenced to make the code work for our needs (like users needed)

what makes styles.scss files different from styles.css

The main difference between a `styles.scss` file and a `styles.scss` file is that `styles.scss` is a Sass file while `styles.scss` is a regular CSS file.

Sass (short for Syntactically Awesome Stylesheets) is a CSS preprocessor that adds features to CSS like variables, functions, nesting, and mixins, among others. Sass files use a different syntax than regular CSS, with some added features that make it easier to write and maintain stylesheets.

In contrast, CSS is a style sheet language used for describing the presentation of a document written in HTML or XML. It does not have any preprocessor features like variables or mixins, and its syntax is simpler than Sass.

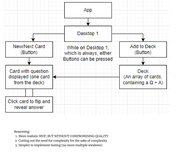
When you write styles in a `styles.scss` file, you can use Sass-specific features to make your code more modular, reusable, and maintainable. You can also use variables to define values that you can reuse throughout your stylesheet, making it easier to update your styles in the future.

Once you've written your Sass code, you need to compile it to CSS using a Sass compiler. The output of the Sass compiler is a regular `styles.scss` file that you can include in your HTML document, just like any other CSS file.

Overall, the main difference between a `styles.scss` file and a `styles.scss` file is that Sass files provide more advanced features and a more organized syntax that can help you write better CSS code.

Current product as of Thursday

Updated Userflow



Reasoning:
1. More realistic MVP, BUT WITHOUT COMPROMISING QUALITY
2. Coding out the need for complexity for the sake of complexity
3. Simple to implement testing (no more multiple windows)