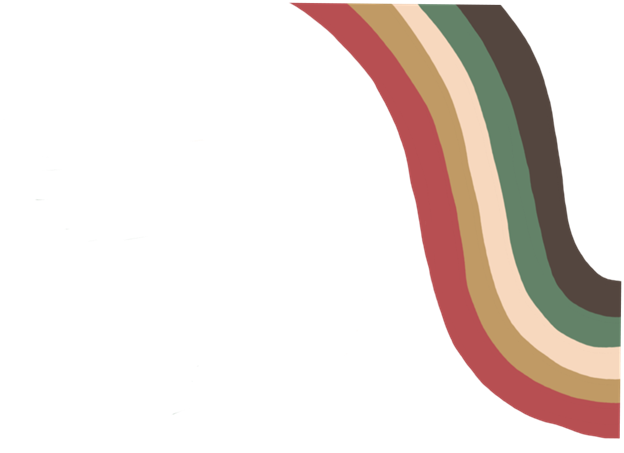
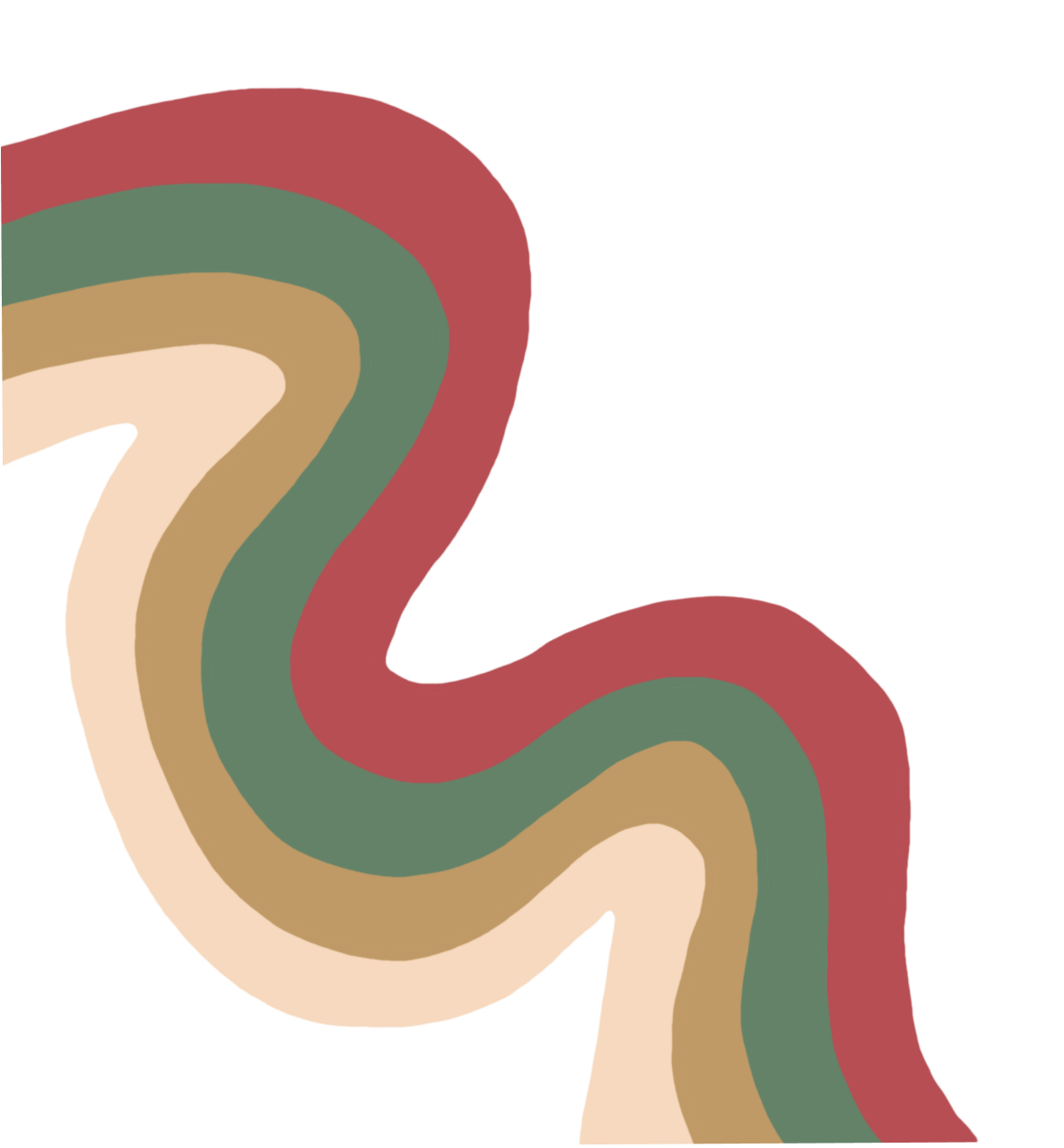
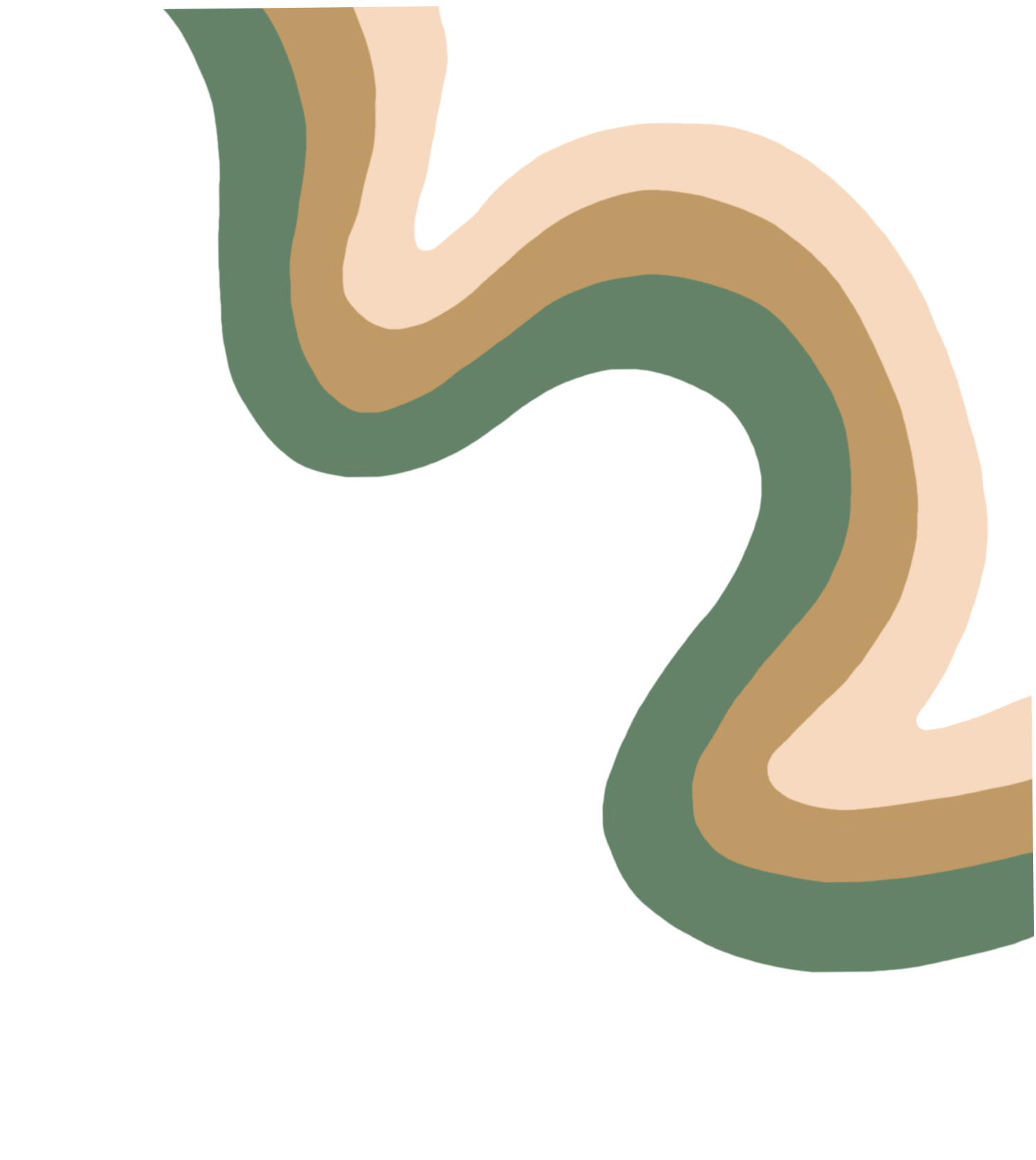
#### **Milestone 5: Front-End Development for Mobile Application:**

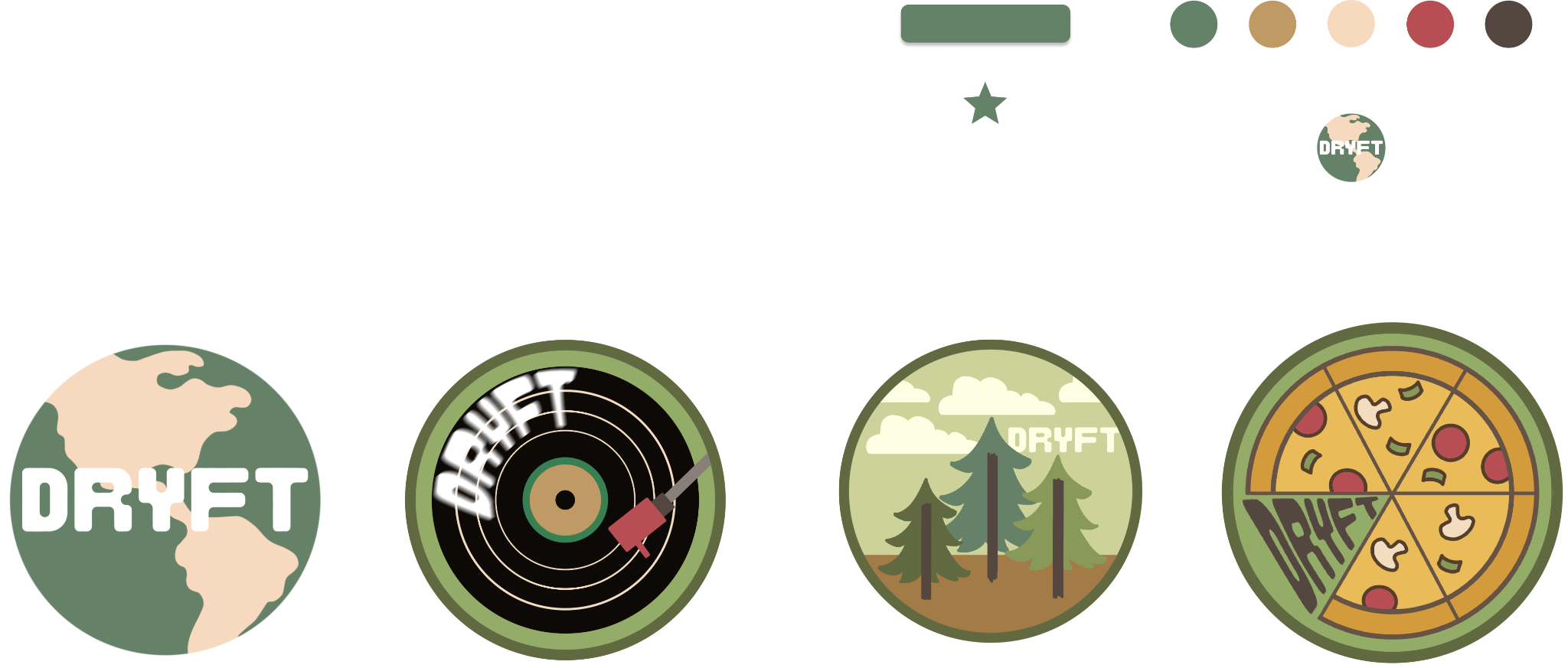
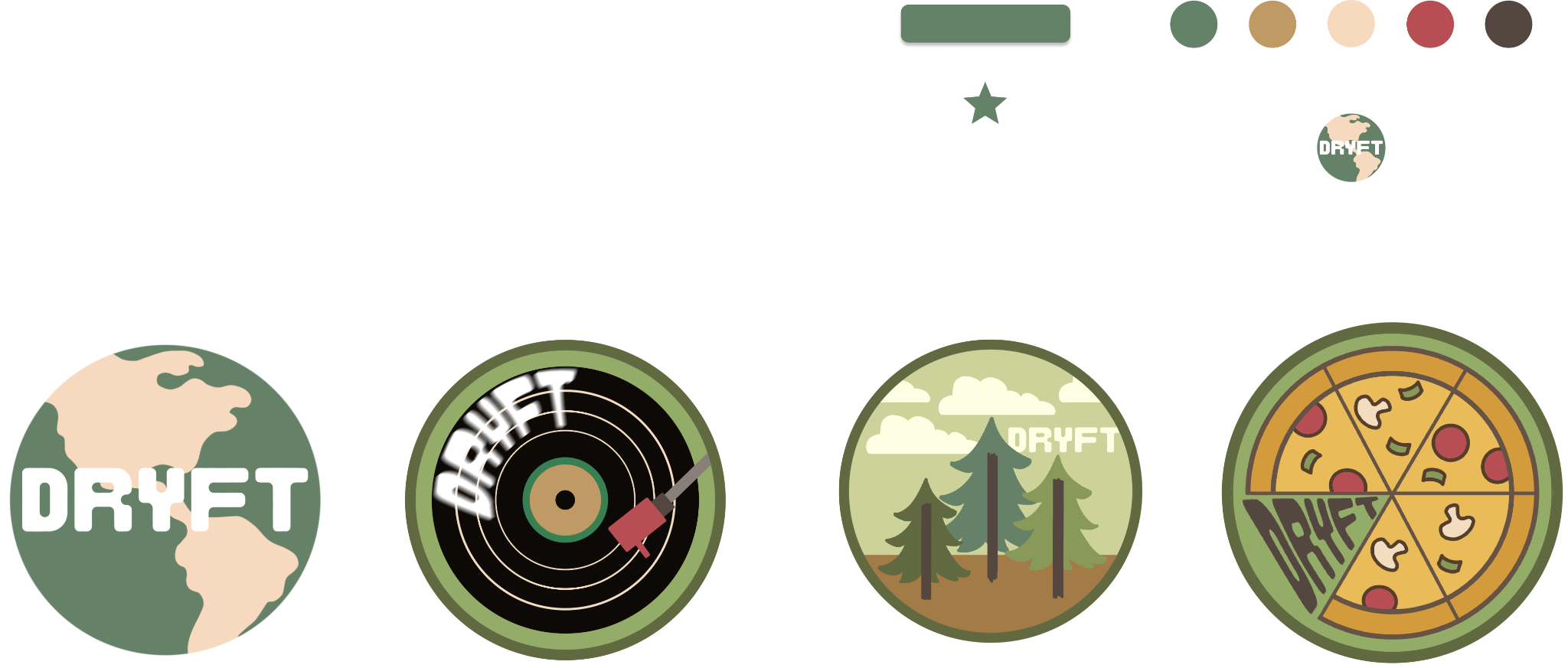
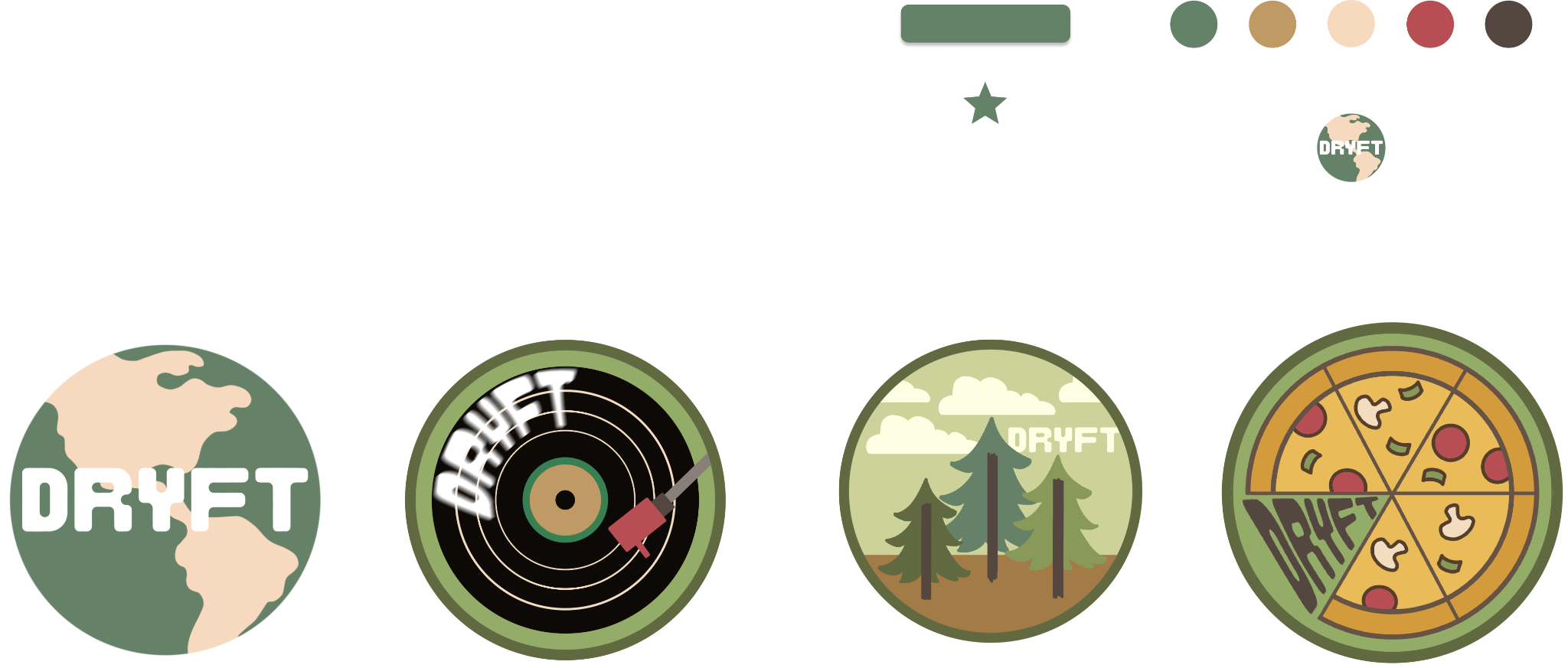
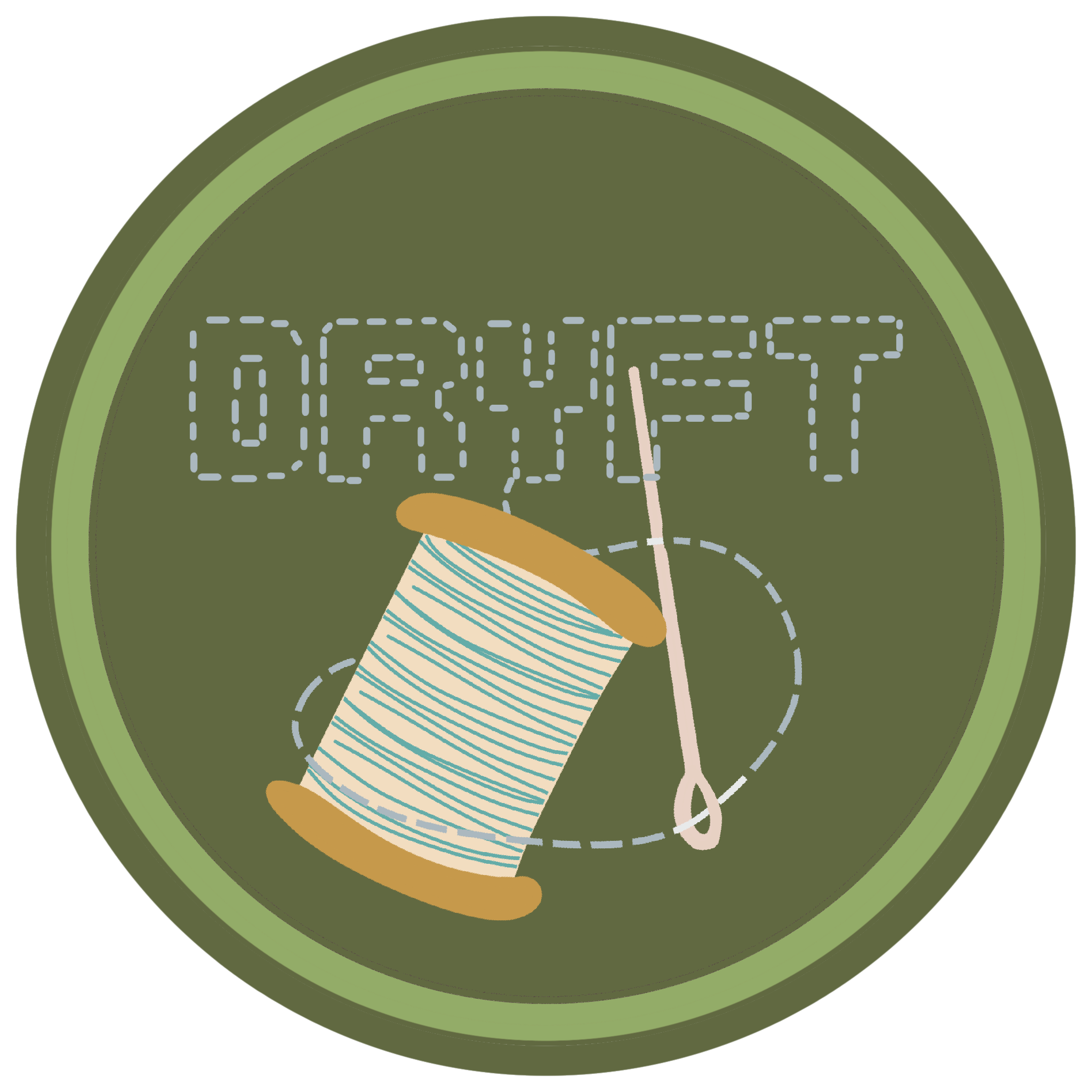
**Dryft**

**Deliverables:**

* HTML, CSS, and JavaScript code for the user interface of the mobile application

**Documentation**:

* **HALEY** - *Executive summary: A brief overview of the front-end development work completed for the website:*
  + - The front-end development for Dryft is underway. As of now, we are currently focused on the client-side development of the web design, which entails our latest project: the implementation of the interactive elements of our site. For instance, the buttons, login forms, and navigation menus are looking great which will lead us to accessible engagement for the user. The app itself also seems to be loading and running quickly and efficiently, which is excellent. Additionally, we are working on the front end of the site to make sure the platform is operating as a website and *also* as a mobile app. We have really seemed to hone in on an inclusive, modern, and seamless experience for our users. We are as ready as ever to move into our next stages of development.
* **LAUREN** - *Design assets: A list of the design assets (such as images, icons, and fonts) used in the website user interface*
  + - The design assets for Dryft were chosen to align with the app’s mission of promoting travel and supporting small businesses while creating a visually inviting experience. We chose an earthy, retro-inspired color palette with warm greens, browns, and muted reds to evoke a sense of adventure and authenticity, tying the design to themes of exploration and connection with local culture. The logo, featuring a stylized globe with bold, blocky text, reinforces the travel aspect of the app. Wavy, organic patterns inspired by vintage travel aesthetics create a sense of movement, symbolizing the journey users take as they discover new places. Icons, such as a vinyl record, a forest landscape, and a pizza, represent different experiences users can collect, making the app feel interactive and rewarding. By integrating these natural, travel-inspired elements, Dryft’s design fosters a sense of nostalgia and curiosity, encouraging users to explore and engage with local businesses in a meaningful way.
    - Images:

* **NICOLE** - *User interface design: A description of the user interface design, including any user flows or interactions*
  + - Dryft’s user interface is built around a streamlined, single-page architecture with multiple “page” sections that are toggled via JavaScript, ensuring smooth transitions without constant page reloads. A fixed bottom navigation bar provides direct access to Home, Feed, Tokens, Map, and Profile, while a top back button supports intuitive, step-by-step navigation. Users begin on a clean Landing page, move through Login or Signup flows, and then arrive at the Home screen, which features a search bar, local recommendations, and quick links to other sections. Throughout the app, button-based interactions, scrollable carousels, and visual cues guide users seamlessly from browsing tokens and favoriting posts to editing their profile or scanning new tokens, creating a cohesive and efficient mobile-friendly experience.`
* **MAGGIE** - *Technical implementation: A description of the technical implementation of the user interface, including details on the HTML, CSS, and JavaScript code used*
  + HTML: Our main use for HTML was structure. The app is structured using multiple pages, organized throughout the code using <div> elements. To handle transitions between pages, we used multiple instances of the showpage() function, connected to buttons and other user interactions. CSS: Our main use for CSS was style. Our main goal with our CSS styling was to replicate the styling choices we made when creating the hi-fi prototype. We tried to use as many of the same fonts, icons, and colors as we could. JS: Our main use for Javascript was interactivity. Javascript was mainly used to define and structure our functions. One of the main issues we ran into while trying to replicate certain functions we used in our Figma prototype was that as a low/no code platform, there are interactions that are easier to execute in a “drag and drop” kind of setting, as opposed to actually coding them.
* **ADONIA -** *Testing and debugging: A description of the testing and debugging process for the user interface, including any issues encountered and how they were resolved*
  + The testing process incorporated a combination of usability testing to ensure a smooth and accessible user experience. The interface was tested on two different browsers (Chrome and Edge) to ensure consistency. Initially, the elements such as images and buttons, shifted on different screen sizes. To resolve this we Implemented position: relative inside .container in CSS and ensured proper alignment using left, top, and transform properties. The goal was to make the interface match the High-Fidelity Prototype as much as possible. During the development of Dryft, debugging played a critical role in ensuring the app functioned smoothly across different devices and user interfaces. However, the front-end development testing and debugging initial coding stages still required much more collaboration and lacked the identical appearance to the prototype. The resolution was to develop a seamless mobile app integration of code that would be faster to implement individually and more time-efficient.

It was discovered that JavaScript’s navigateTo() function was not properly linked in the updated lines of code. We fixed this by ensuring the script was correctly referenced in the HTML and added console.log() statements in the terminal for debugging. Several issues were identified and resolved during the debugging process. One key issue was that images and buttons shifted unpredictably on different screen sizes for the interface. Which was tested with the right-click inspection tool for toggle size (mobile or web) toolbar. Another challenge involved the “Get Started” and “Login buttons”, which did not redirect users as expected was traced to JavaScript’s navigateTo() function not being properly linked.

In addition, throughout the testing phase of the interface, the GitHub repository played a crucial role in collaboration. It allowed our group to track issues, commit fixes, and ensure version control, making it easier to identify and resolve bugs efficiently. Initially in the beginning stages, regular commits and issue tracking helped streamline the debugging process and maintain code consistency across all team members. We encountered and resolved several key issues throughout the process. Debugging with the DevTools console revealed that event listeners were not properly attached to form elements. This was fixed by ensuring validation scripts executed at the right time.

Overall, form validation errors were not displaying the correct feedback messages. A significant challenge arose with our GitHub repository not showing commit changes or images in the browser. Initially, team members’ commits were not appearing in the repository. After investigating, we found that some commits were being pushed to the wrong branch. We resolved this by ensuring everyone was working on the correct branch and merging changes properly. Local changes were not syncing due to authentication issues. This was fixed by reconfiguring Git credentials. Due to these repository issues, our team ultimately decided to submit the updated coded project by file upload and a walkthrough video instead of using GitHub. This ensured that work was properly included and that we had a complete, functional version of Dryft for submission.