High Level Design Document (HLD)

REACT DRUM KIT

By: Shubham Singh

Document Version Control

Date Issued	Version	Description	Author
28-01-2023	1.0	Initial HLD	Shubham Singh
30-01-2023	1.1	Final Draft	Shubham Singh

Contents

Table of Contents

Abstract

The react drum kit is an interactive musical instrument that allows users to play various percussion sounds by clicking on different drum pads. The user interface is designed with a clean and minimalist style, making it easy to use and visually appealing. The responsive design ensures the drum kit works seamlessly on various devices and screen sizes. The React framework was used to create a dynamic and interactive experience, while the Tailwind.css styling library was used to create a stylish and functional interface. Overall, this drum kit provides a fun and engaging experience for users of all levels.

1. Introduction

1.1 Why this High-Level Design Document?

This high-level design document for a React drum kit helped in plan and organize project before start coding. It allows to outline the overall structure and functionality of drum kit and to identify the key components and their relationships and can be used as a reference manual for how the modules interact at a high level.

This document can also help to:

- Clarify the project requirements and scope
- Determine the components that need to build and how they fit together
- Identify potential challenges and risks
- Serve as a reference for later development stages.
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Include the design features and the architecture of the project

By having a clear and detailed design document, we can streamline the development process, ensure that the final product meets our expectations, and minimize the risk of misunderstandings or rework.

1.2 Scope

The scope of the high-level design document for a React Drum Kit includes:

- Overview of the project: The goal of this project is to create an interactive drum kit that users can play using their keyboard or mouse. The drum kit will consist of a set of drum pads that play different sounds when triggered.

- User Requirements: The drum kit must have at least 7 drum pads, each with a unique sound. The user interface must be intuitive and easy to use. The drum kit must also be responsive and work well on different devices and screen sizes.

- System Architecture: The system architecture will consist of a React component that represents the drum kit, and child components for each drum pad. The components will communicate with each other using props and state. The audio files will be loaded using the JSON file.

- Component Description: The drum kit component will be the main container for the drum pads. The drum pad components will each contain a button and an audio element that plays the sound. The audio files will be stored in a separate folder.

- Technologies: The technologies and tools used to build the drum kit will include React, JSX, Tailwindcss, JavaScript, and the Web Audio API. No additional libraries or frameworks will be used.

- Data Flow: The user will trigger a drum pad by clicking on it or pressing a corresponding keyboard key. The drum pad component will then play the audio file and update its own state to indicate that it is active. The drum kit component will pass down the necessary props to each drum pad component.

- User Interaction: Users will be able to play the drum kit by clicking on the drum pads or by pressing the corresponding keyboard keys. The drum kit will display a visual indicator when a drum pad is triggered, such as a highlight or animation.

- Error handling: The drum kit will include error handling to handle any issues that may arise, such as missing audio files or invalid user inputs. The drum kit will display an error message if an error occurs.

- Deployment: The drum kit will be deployed to a web hosting platform, such as Netlify. The project will be tested on different devices and browsers to ensure that it works correctly. The code will be version controlled using Git and stored on GitHub.

1.3 Definitions

Term	Description
JSON	JavaScript Object Notation
CRA	Create React App
HTML	Hyper Text Markup Language
JS	JavaScript
JSX	JavaScript XML
VCS	Version Control System

2. General Description

2.1 Product Perspective

The react drum kit is an interactive musical instrument that allows users to play various percussion sounds by clicking on different drum pads.

2.2 Problem statement

Build a drum kit app in which players can make a corresponding drum noise by pressing each key.

- To design a page where users can play sound with choice by pressing the keyboard key or clicking the mouse.
- User can on/off the drum sound or can change the volume of drum.
- Another page for the instructions to how to play the drum kit.
- Modal for YouTube video demo if available.

2.3 Proposed Solution

The solution proposed here is to build a single-page web application using React, Tailwindcss and JavaScript. The main component will represent the drum kit and child components will be created for each drum pad. Users will be able to play the drum kit by clicking on the drum pads or pressing the corresponding keyboard keys. The components will communicate using props and state, and the audio files will be stored and loaded locally.

2.4 Further Improvements

These are some additional improvements that can be made to the React drum kit:

- Customizable drum pads: Users should be able to customize the drum pads by changing the sounds or adding new ones.
- User profiles: The drum kit should allow users to create their own profiles and save their customizations.
- The drum kit should allow users to record their drum performances and export the recordings as audio files.

2.5 Project requirements

1. **Functionality Requirements:**
   - At least 8 drum pads, each with a unique sound.
   - Responsive design that works on different devices and screen sizes.
   - Intuitive user interface that allows users to play the drum pads by clicking or pressing the corresponding keyboard keys.
   - Visual indicator that shows when a drum pad is triggered.
   - Indicator when drum is on or off.
   - Error handling while trying to play drum when it’s off or invalid user inputs.

2. **Technical Requirements:**
   - Built using React, HTML, CSS, and JavaScript.
   - Audio files stored in a separate folder and loaded dynamically.
   - Deployed to a web hosting platform, such as Netlify.
   - Version controlled using Git and stored on GitHub.

3. **User Experience Requirements:**
   - Simple and intuitive user interface.
   - Mobile compatibility and responsive design.
   - Tutorials page on how to play the drum for beginners.

2.6 Tools Used

- Code editor: Visual Studio Code.
- Node.js and npm: Node.js as JavaScript runtime & npm package manager for Node.js.
- React: JavaScript library for building user interfaces.
- React-Router-DOM: client-side routing
- Tailwind CSS: styling the website
- Web browser: Google Chrome.
- Git: version control system
- GitHub: to manage code.

2.7 Constraints

The constraints for the React Drum Kit include:

- Browser Compatibility: The application may only work in modern browsers that support the latest JavaScript and HTML standards with support for Tailwindcss.

- Sound Quality: The quality of the drum sounds used in the application may be limited by the sound library used, or by the user's device and browser.

- Keyboard Shortcuts: The keyboard shortcuts used to trigger drum sounds may not work for all keyboard configurations, or may interfere with other keyboard shortcuts used by the user.

- User Experience: The overall user experience may be limited by the application's design, navigation, and functionality.

- Accessibility: The application may not be accessible to users with disabilities, or may not fully support users with accessibility needs.

2.8 Assumptions

It is assumed that the end-users have access to a web browser, network connectivity and web browser compatible with the technology used to build the React drum kit, such as React and JavaScript. Also it is assumed that the end-users have a basic understanding of how drum kits work and how to play them.

3. Design details

3.1 Process Flow

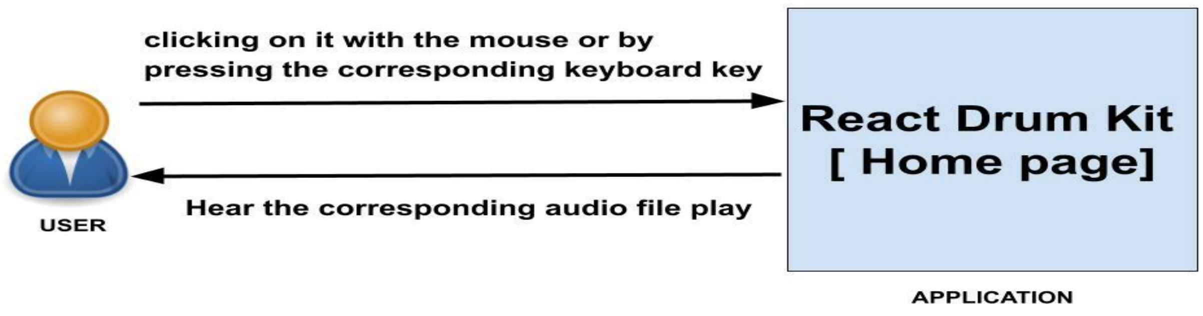

clicking on it with the mouse or by pressing the corresponding keyboard key
Hear the corresponding audio file play
USER
React Drum Kit [Home page]
APPLICATION

Figure 1: Workflow as a User [Home page]

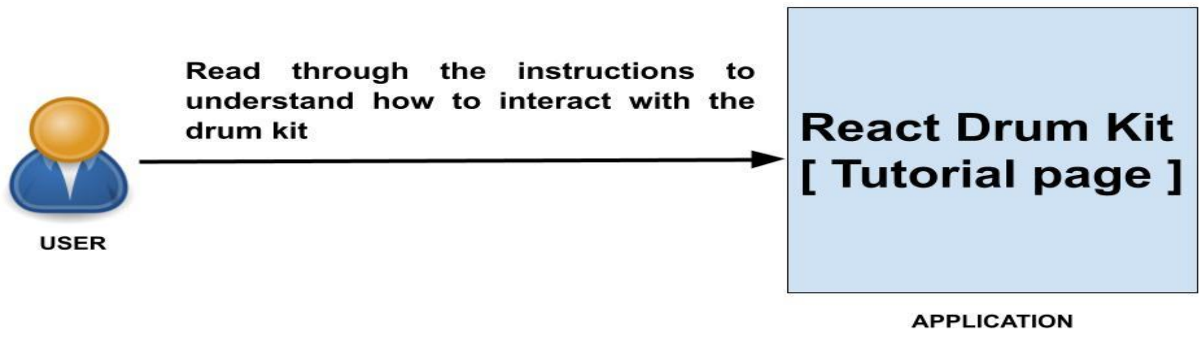
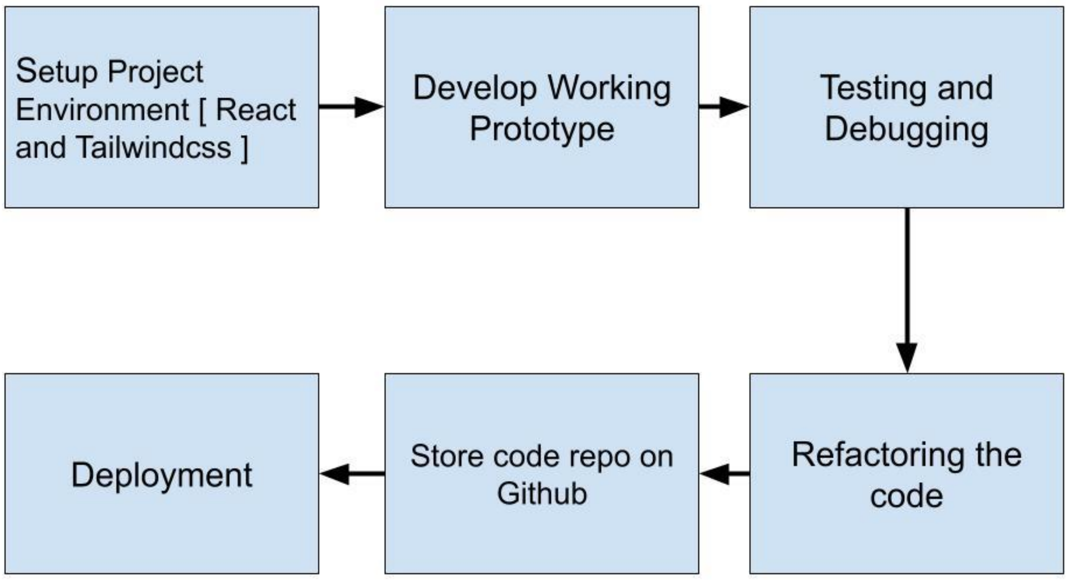

Read through the instructions to understand how to interact with the drum kit
USER
React Drum Kit [Tutorial page]
APPLICATION

Figure 1: Workflow as a User [Tutorial page]

3.2 Development Process


Setup Project Environment [React and Tailwindcss]
Develop Working Prototype
Testing and Debugging
Refactoring the code
Store code repo on Github
Deployment

3.3 Error handling

Error handling mechanism for showing alerts while pressing keys while the React drum is turned off.

4. Performance and Accessibility

Improving performance ensures that the application runs smoothly and efficiently, while making the application accessible ensures that it can be used by a wide range of users.

4.1 Reusability

The code written and the components used should have the ability to be reused withno problems.

4.2 Application compatibility

This project should be cross platform supporting i.e., should support mobile and desktops also. One way to resolve this is by adapting mobile first development while building our application.

4.3 Resource utilisation

When any task is performed, it will likely use all the processing power available until that function is finished.

4.4 Deployment

For hosting the application I used **Netlify** web services because of cost factor and easy to use.

5. Conclusion

In conclusion, the React drum kit project is a valuable and beneficial solution for users. For users, the application provides a fun and engaging experience that allows them to explore their musical creativity. The dynamic audio file loading and clear feedback features allow for a seamless and intuitive experience, while error handling ensures that the application runs smoothly and effectively