



KBPartPicker

Final System Design Document

Team Members

Rejan Quebral r_quebral@u.pacific.edu

Instructor

Dr. Canniff

Github URL: <https://bit.ly/3omhoks>

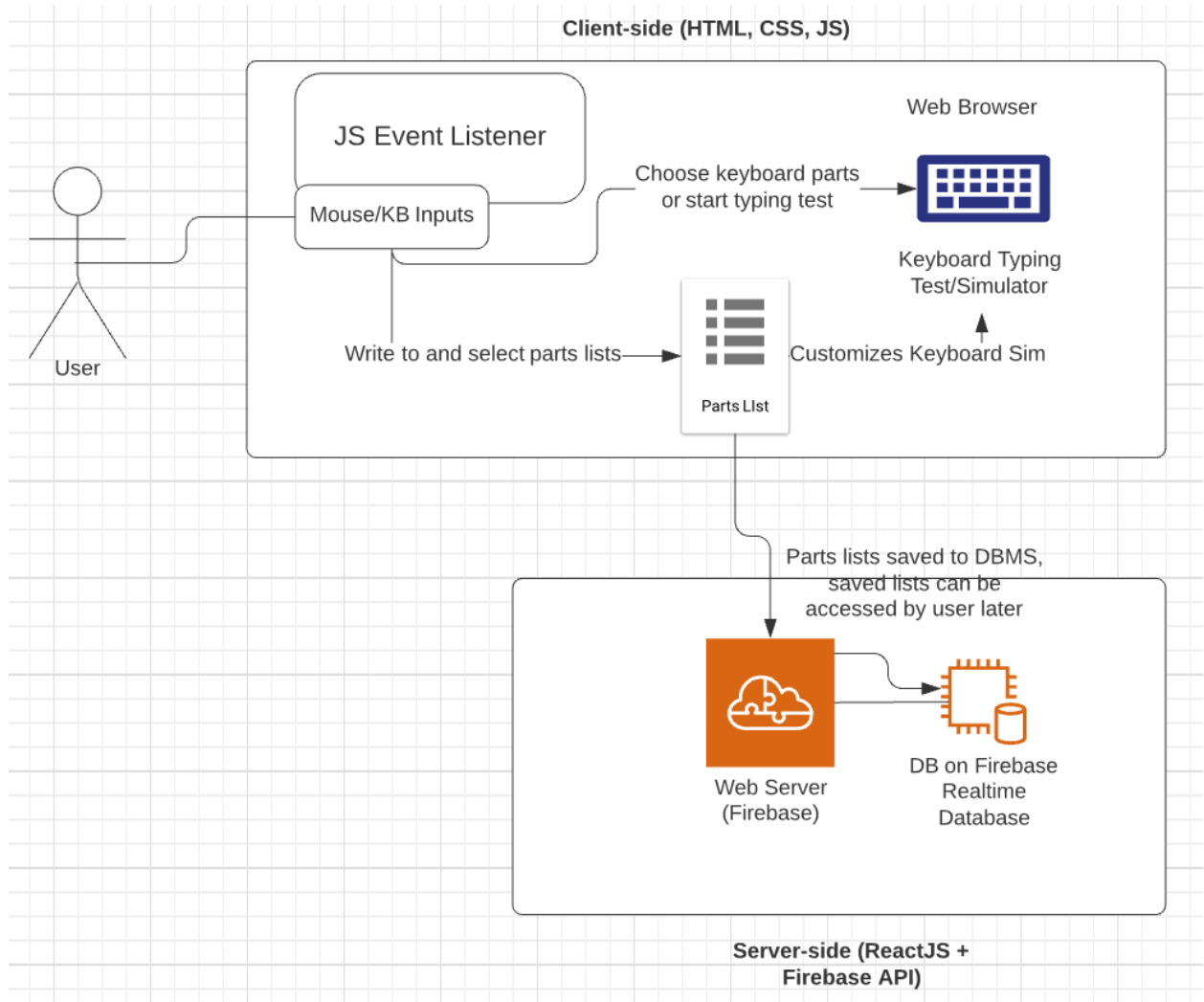
Website URL: <https://kb-part-picker.web.app>

Last revised: May 3, 2022

Table of Contents

Table of Contents	1
System Architecture	2
Software Modules	2
Hardware Components	3
User Interface	3
Interfaces to External Systems	3
Hardware, Software, and System Requirements	3
Hardware	3
Software	3
System	3
External Interfaces	4
Firebase	4
Software Design	4
Class Diagram and Class Specifications	4
Design Considerations	4
Interaction Diagrams	5
User Interface Design	6
Homepage	6
Parts List Creation Page	7
Account Creation Page	8
Glossary of Terms	9
References	10

System Architecture



Software Modules

On the client side, the user will interact with the web application with the help of JavaScript `EventListener()` functions to get keyboard inputs to help with keyboard typing simulations. *KBPartPicker* will be displayed with HTML5 and CSS through whatever web browser the user chooses, though complete compatibility will be ensured with Mozilla Firefox first.

On the server side, the Firebase API will be used within ReactJS to store and read Realtime Database data. The Firebase website will create an application instance to host *KBPartPicker*.

Hardware Components

The web application will run on the user's computer through a web browser of their choice, and the web server and database will be maintained by Firebase. As *KBPartPicker* will be compatible with Mozilla Firefox first, users should make sure their computers meet the minimum requirements outlined here:

<https://www.mozilla.org/en-US/firefox/96.0.3/system-requirements/>

User Interface

The user interface will be created with modern HTML5, CSS, and ReactJS (React JavaScript) practices as of 2022. Users will interact with the web application mainly through JavaScript `EventListener()` functions and will be able to select keyboard parts through dropdown menus.

Interfaces to External Systems

KBPartPicker will interact with the web server through Firebase API calls, and any requests to the database will be handled by the web server.

Hardware, Software, and System Requirements

Hardware

The minimum requirements correspond with any Windows computer that can run Mozilla Firefox:

- Pentium 4 or newer processor that supports SSE2
- 512MB of RAM / 2GB of RAM for the 64-bit version
- 200MB of hard drive space

Users will also need an internet connection. Whether they connect through wi-fi or ethernet is irrelevant as *KBPartPicker* will not require intensive loads such as software downloads.

Software

As *KBPartPicker* is a web application, it will initially be developed to display properly in the Mozilla Firefox browser but should time allow, Chromium-based browsers should also be supported. There are no plans, however, to support any mobile devices.

System

KBPartPicker will be developed with the most recent version of Mozilla Firefox available as of the writing of this document, Firefox 96.0.3, which requires a Windows 7 or higher operating system.

External Interfaces

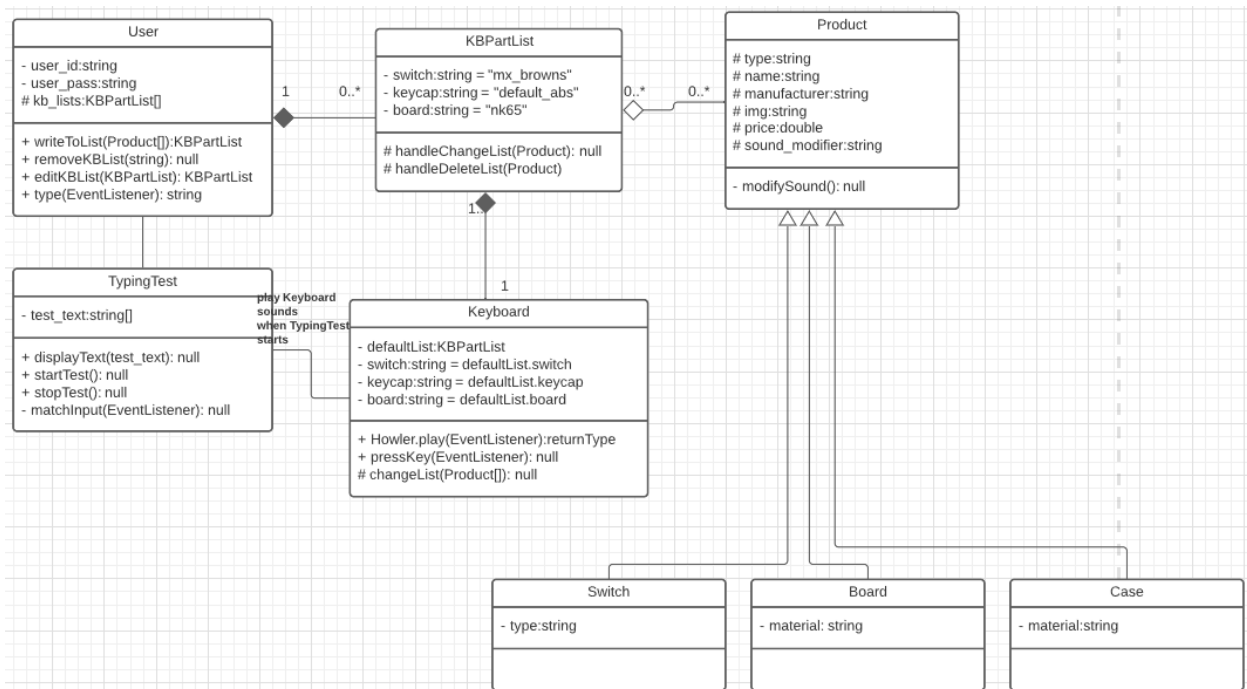
Firestore

This project will be hosting a web server and database management server on Firestore. Documentation can be found here:

<https://firebase.google.com/docs>

Software Design

Class Diagram and Class Specifications



Design Considerations

User was designed for flexibility, as there may be additional implementation in the future to allow manufacturers to upload their own sounds. For now, however, to

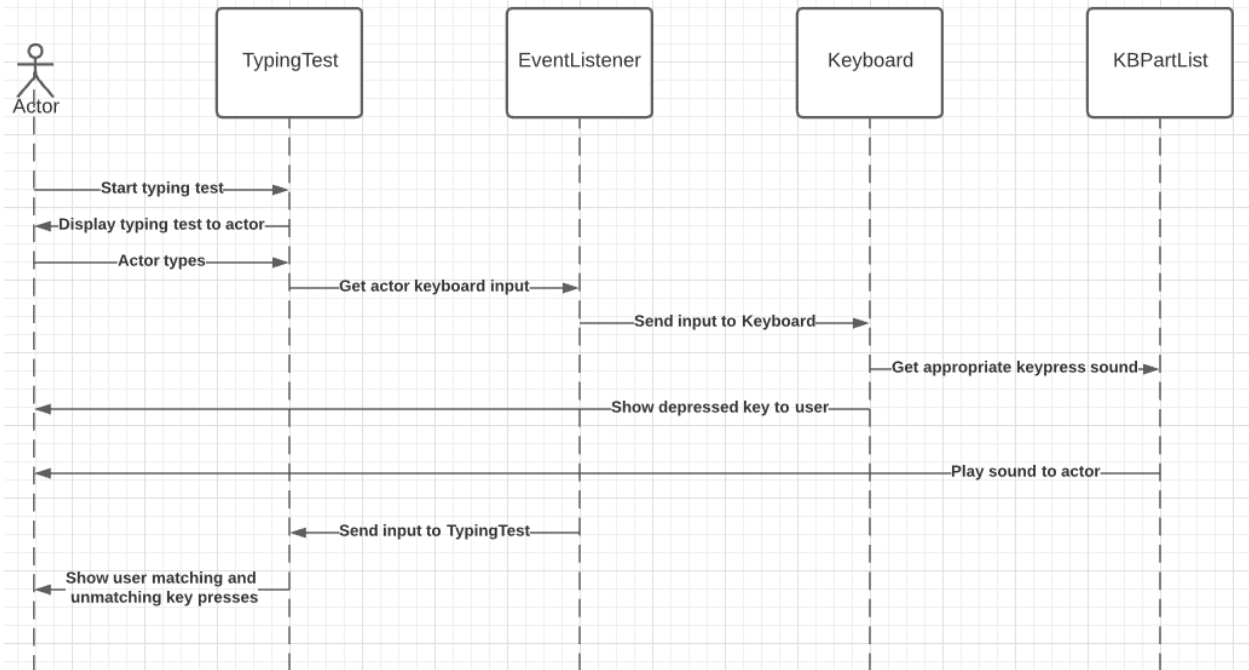
reduce complexity this will stay an additional feature and **Users** will only be able to create and modify lists and interact with the **Keyboard** through **TypingTest**.

Product was created to increase abstraction, as **Switch**, **Board**, and **Case** share multiple of the same properties.

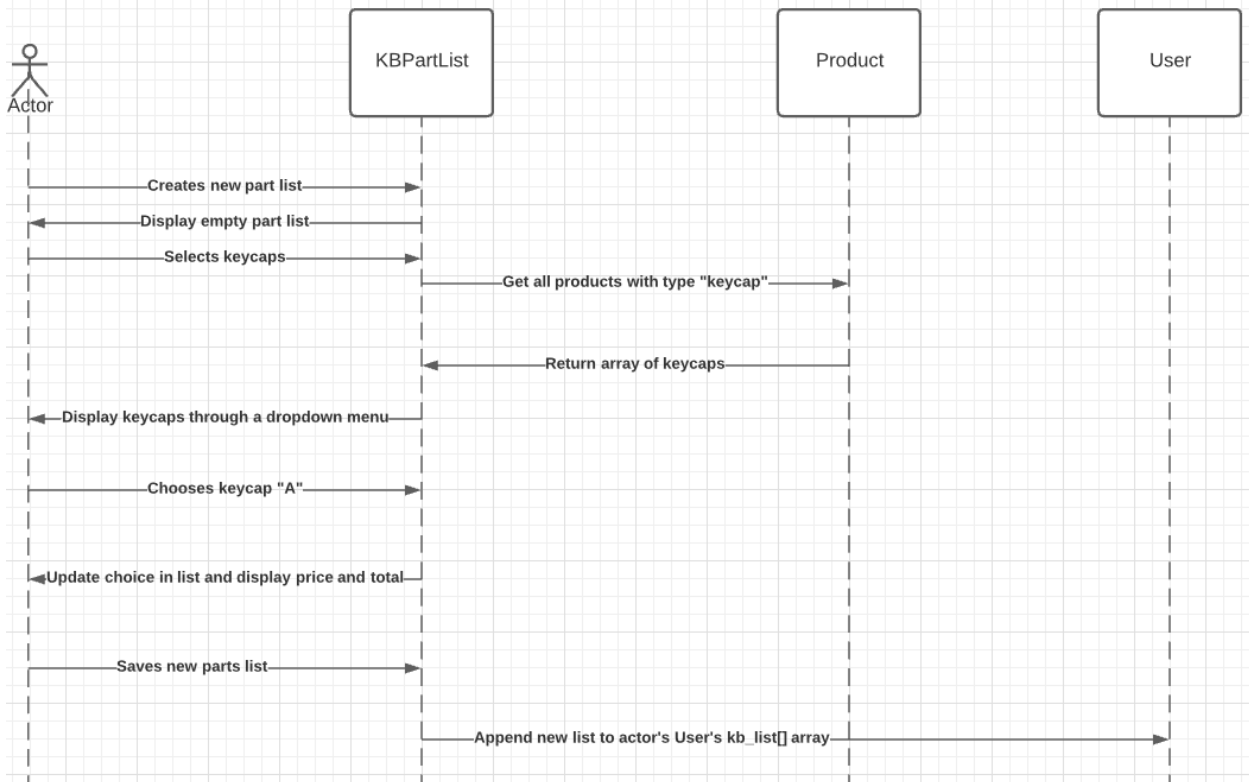
TypingTest and **Keyboard** were designed with loose coupling in mind. **TypingTest** will gather user input which will be translated to **Keyboard**, while **Keyboard** will play appropriate keypress sounds and take on the look of an actor's chosen **KBPartList**.

Interaction Diagrams

Use Case: Typing Test on Homepage

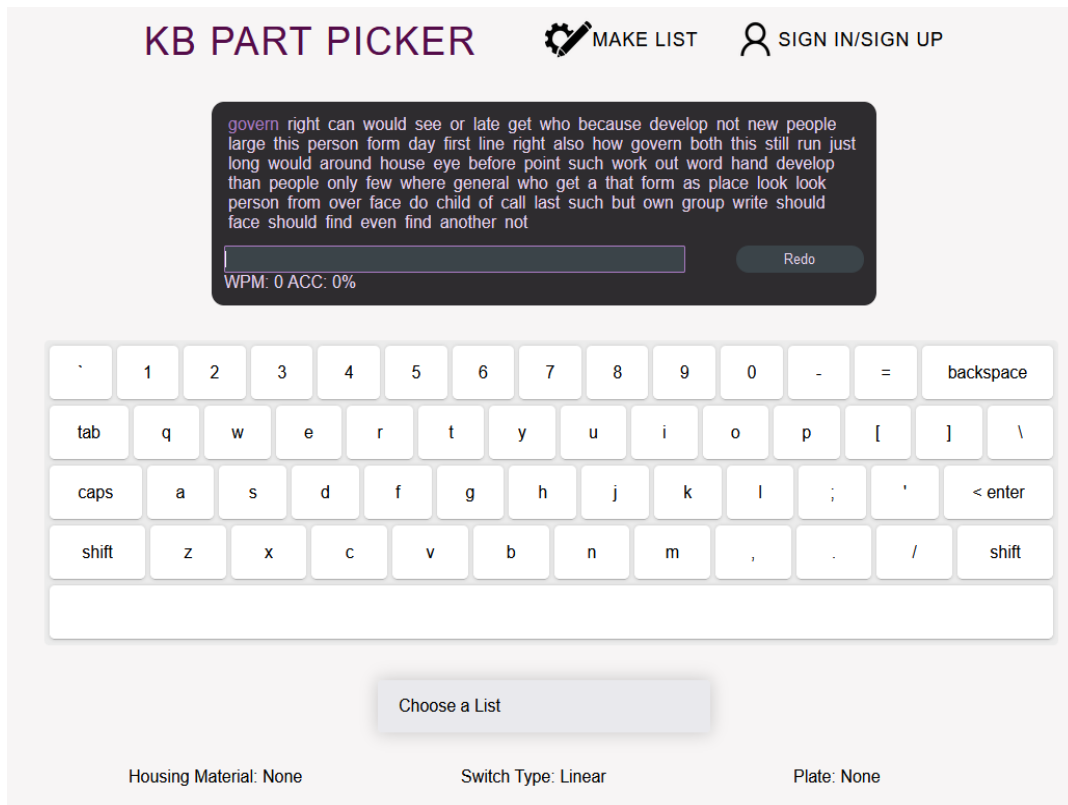


Use Case: Creating KBPartList (Adding Keycap)



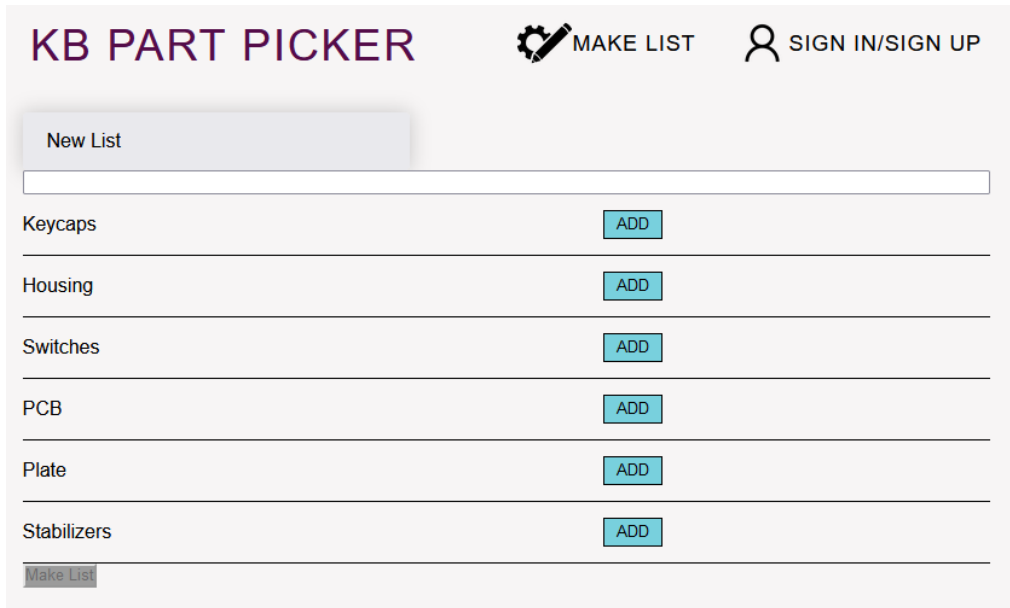
User Interface Design

Homepage



When first entering the site, users will be brought to a page with a pre-built keyboard list, the option to customize the keyboard switch, keycap, and board, and a button to bring them to a page to create a new keyboard parts list. They will also be able to conduct a typing test to hear the custom keyboard in action.

Parts List Creation Page



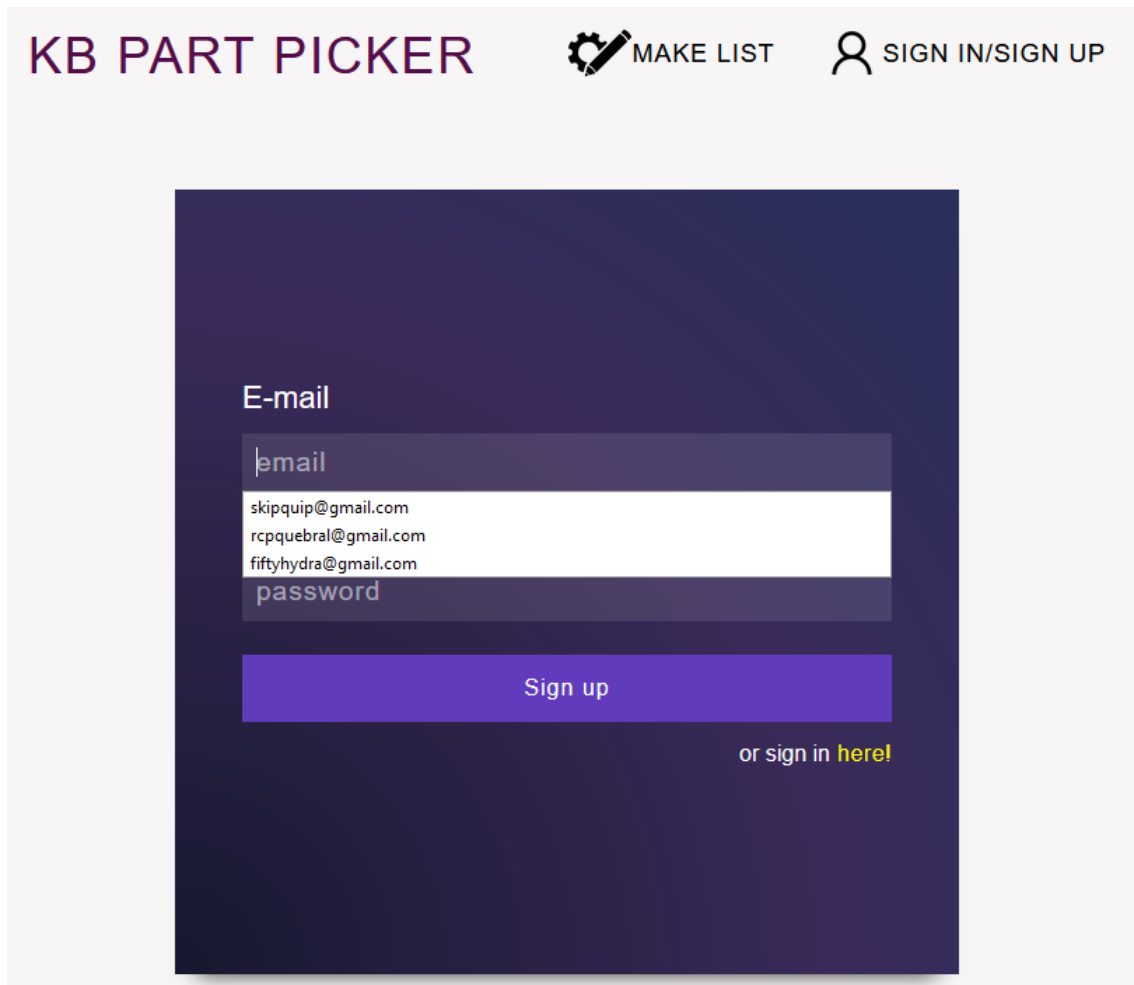
The screenshot displays the 'KB PART PICKER' interface. At the top, there is a header with the title 'KB PART PICKER' in purple, a 'MAKE LIST' button with a gear icon, and a 'SIGN IN/SIGN UP' button with a user icon. Below the header, there is a 'New List' button. A search bar is positioned below the 'New List' button. The main content area features a list of keyboard components, each with an 'ADD' button:

Component	Action
Keycaps	ADD
Housing	ADD
Switches	ADD
PCB	ADD
Plate	ADD
Stabilizers	ADD

At the bottom of the list, there is a 'Make List' button.

If the user clicks the "MAKE LIST" button, they will be brought to a page similar to PCPartPicker, where they will be able to create and save a list of keyboard parts, pick an already existing parts list, and see the price of their current list's items and the total price. Should any compatibility issues arise, the site will alert the user with red text at the bottom left. If the user wants to return to the homepage, they can click on "KB PART PICKER" at the top of this page

Account Creation Page



The screenshot shows the 'KB PART PICKER' header in purple. To the right are two links: 'MAKE LIST' with a gear icon and 'SIGN IN/SIGN UP' with a person icon. Below the header is a dark blue modal form. The form has a title 'E-mail' in white. It contains two input fields: the first is labeled 'email' and has a list of suggestions (skipquip@gmail.com, rcpquebral@gmail.com, fiftyhydra@gmail.com); the second is labeled 'password'. Below these fields is a large purple 'Sign up' button. At the bottom right of the form, there is a link that says 'or sign in [here!](#)'.

If the user clicks "SIGN IN/SIGN UP" in the top right corner, they will be brought to a form where they can enter their e-mail address and a password which must be of 6 characters of length or more. The user can then click the "Sign up" button or click "or sign in here!" to be given a "Sign in" button instead. In either case, the user must enter something in both fields, or the site will give an error message. If the user is signing in, the e-mail address and password must be in the Firebase Authentication database, or else the form will inform the user that the e-mail does not exist.

Glossary of Terms

Terms	Definitions
<i>Abstraction</i>	Removing aspects from a class to reduce it to its most essential characteristics.
<i>Client Side</i>	In a client-server relationship, this refers to anything that is displayed to the client or is performed on the client's system
<i>Database Management Server (DBMS)</i>	Software which stores, retrieves, and runs queries on data.
<i>Loose Coupling</i>	Designing dependent classes to rely on each others' data as least as possible.
<i>Server Side</i>	In a client-server relationship, this refers to operations performed by a server
<i>Web Application</i>	An application stored on a remote server that is transmitted through the internet by a web browser.
<i>Web Browser</i>	Software used to access and display websites and web applications.
<i>Web Server</i>	A computer that stores website components and software to accept and transmit data for the website.

References

Firebase Documentation (n.d.). Retrieved February 28, 2022, from
<https://firebase.google.com/docs>

Firefox 96.0.3 system requirements. Mozilla. (n.d.). Retrieved February 6, 2022, from
<https://www.mozilla.org/en-US/firefox/96.0.3/system-requirements/>