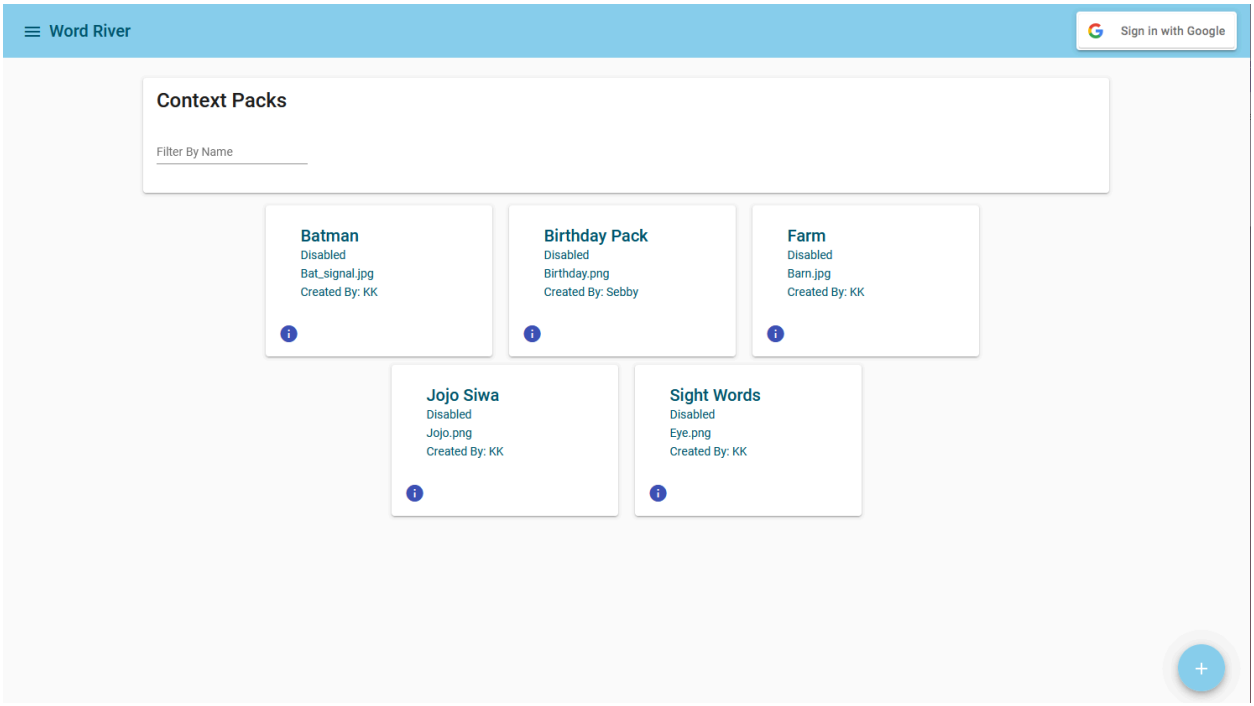
Isabelle Hjelden
University of Minnesota Morris 2023

Word River

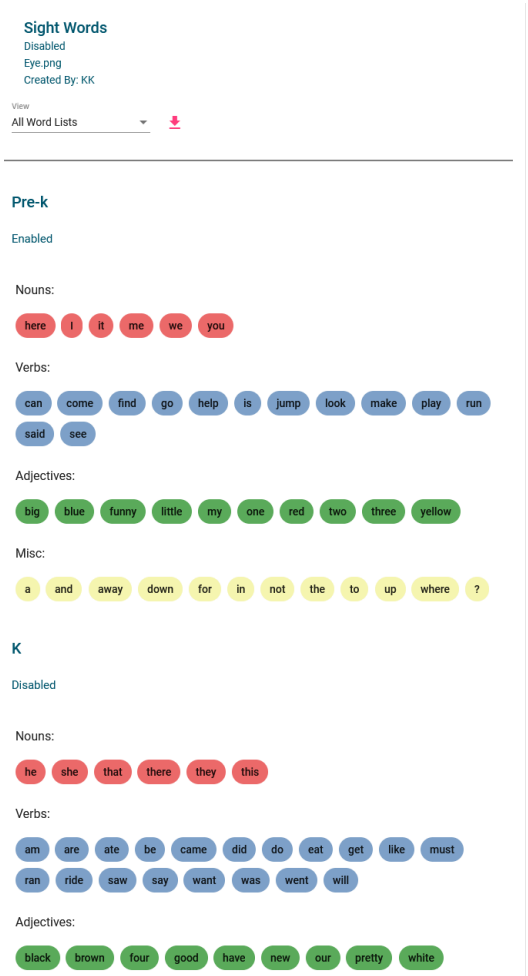
What is Word River?

Word River is an add-on application that is intended to be used alongside Story Builder. The purpose that Word River serves alongside Story builder is to assist teachers, researchers, or parents in organizing their context packs and assigning them to their learners.



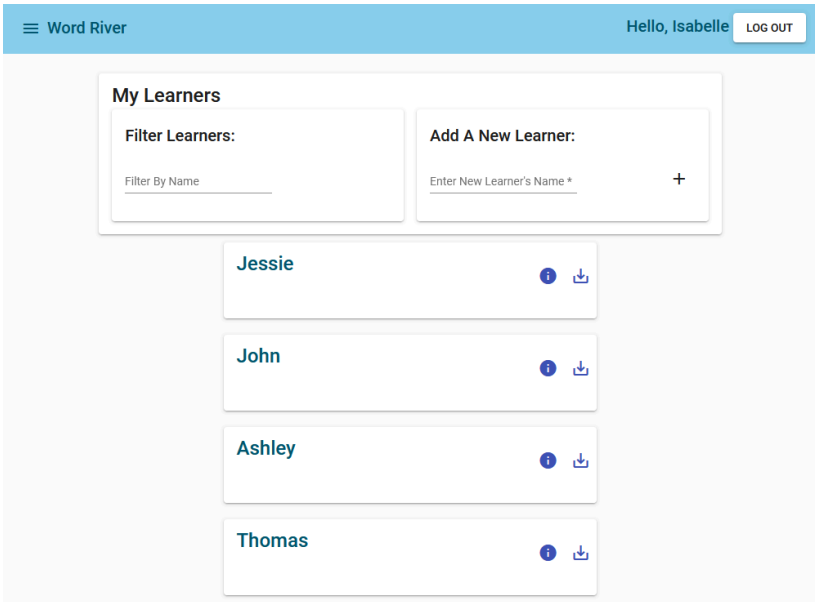
What is a context pack?

Context packs are a pack or bundle of wordlists that all share a common theme or idea. For example, there is a sight word context pack, this pack contains two-word lists, one for pre-K and one for K. A user can assign the whole context pack to a learner, or they can assign certain wordlists within the pack. The purpose of the context pack is to make it easier for the users to know what themes they are selecting for their learners.



What is a learner?

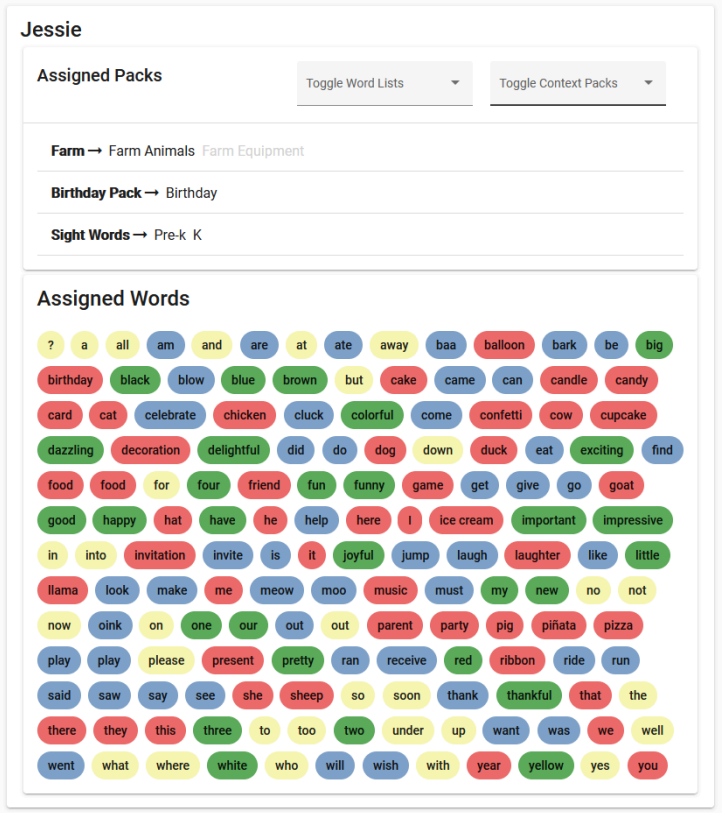
A learner is a student or child that the user is assigning a context pack to. Word River's intended use is for users to organize their context packs and learners. While learners use their assigned context pack through the other application, Story Builder, meaning that learners will not be using Word River directly.



What can a user do in Word River?

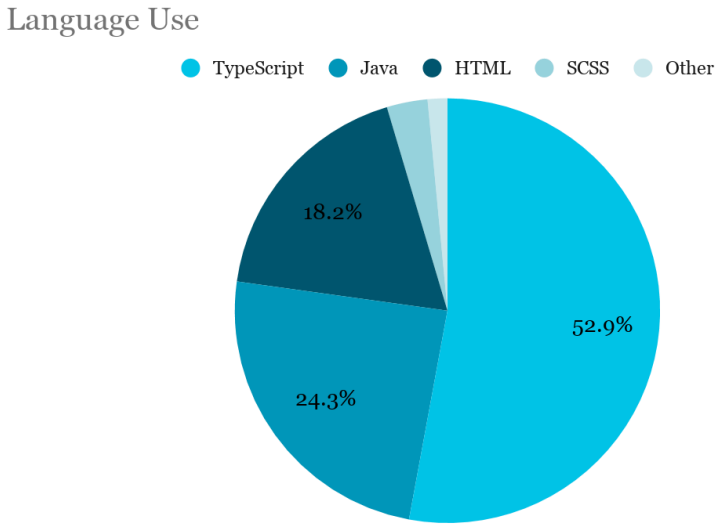
In Word River, a user can add, edit or delete context packs. Similarly, within the packs, the user can do the same function to the wordlists. When creating new word lists in a context pack, the user will type out the words and their different forms under their specific word type (noun, adjective, verb, or miscellaneous)

For the learners' aspect, a user can create a learner and assign context packs to them. When a learner has assigned context packs, the user can download a single JSON file that contains all the learners' context packs.



The tools and platforms used to create Word River.

For Word River, our team used several different languages and tools. The most predominant languages are TypeScript, Java, HTML, and SCSS. The front-end of Word River is built up with Angular 11 as well as Angular Material. The back-end of Word River is built with Java as the server and MongoDB as the database. Digital Ocean was used to host the website. Our team used Karma, Cypress, and JUnit for unit testing. Our team used Zenhub to manage the development of Word River.



How we made Agile work.

Throughout this project, we used the agile programming method. In the beginning, we had a workshop, called an inception deck, to determine the direction of the project. The inception deck included a full rundown of what we were there to do, creating a “feature and not list”, and creating epics and stories based on the information both given to us and we created.

Our team was fortunate to have our customer participate in the inception deck because we were able to receive immediate feedback on what our customer liked and disliked. After creating the epics and stories, we were able to make a smaller list of epics that we thought would be doable in each iteration.

We broke this project up into three iterations spanning over nine weeks, each iteration was used effectively to complete the epics that we sold to our customer at the beginning of the iterations. This is part of the agile programming method because we were continuously providing working software to our customer at the end of each iteration.

Our team was also performing continuous integration. This means that we were testing software immediately and moving it into our main branches as soon as possible.

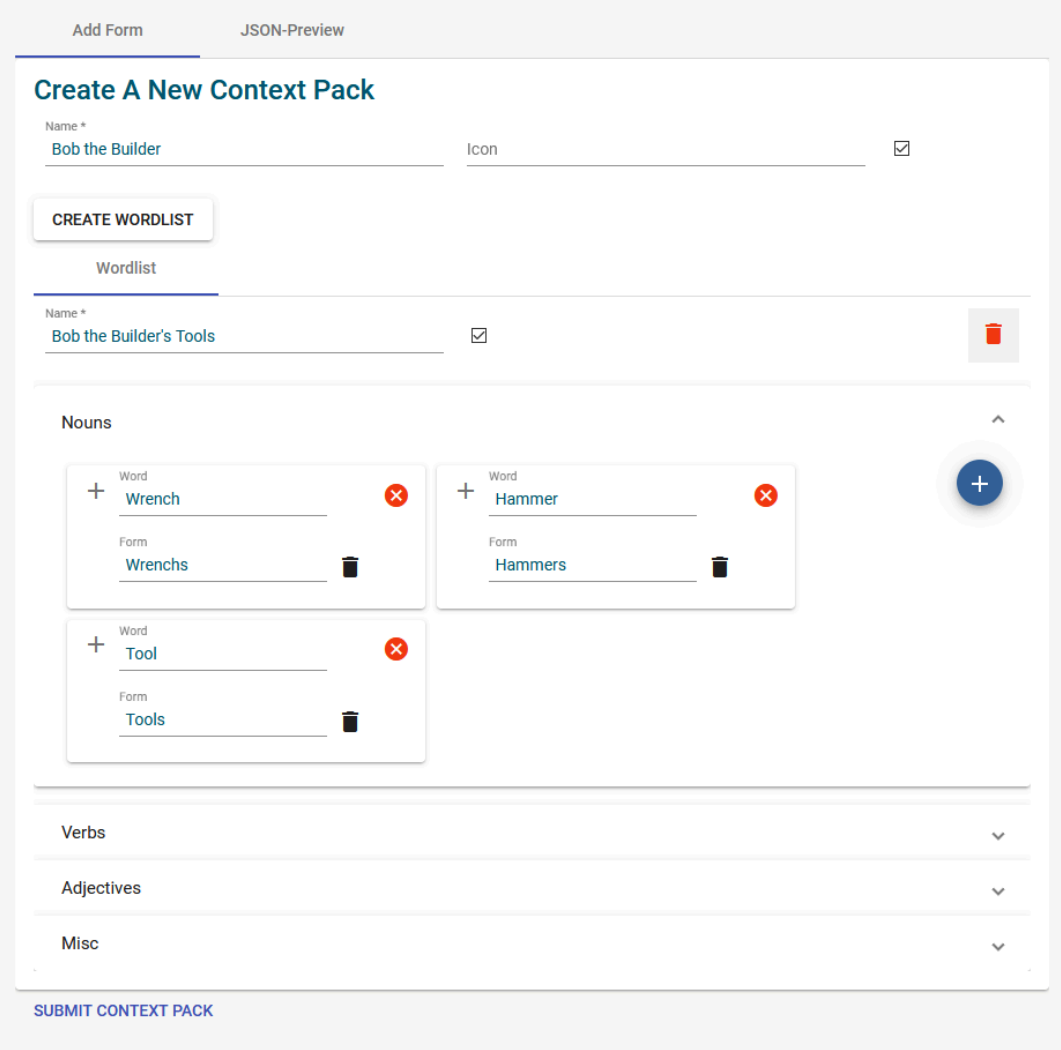
ZenHub played a major role in our Agile development. We were able to plan out all the epics and their necessary steps to completing them through ZenHub and keep track of their progress.

My contributions to Word River.

Looking over all three iterations, I primarily worked on the front-end of Word River and assisted in several areas with the back-end.

For the first iteration, I began working on the server-side and helped with the creation of it. Eventually, I moved on to work on the client-side to start implementing the view context page cards and info pages, but I had to move back to help complete the server because there were a lot of technical issues that needed to be resolved. Those issues went unresolved for a decent amount of time until it was brought to my attention. I was able to fix the issue along with one of my teammates and we continue to work through the rest of our project.

In the second iteration, I was added to a different team and we decided to use their code from the previous iteration. I helped with the research of how to implement editing context packs, however, I worked entirely on the client-side to redesign the adding context pack page. I spent a lot of time learning about Angular Materials and what worked and didn't work for our codebase.



For the final iteration, I assisted in creating a part on the server-side logic for adding a new wordlist to already existing context packs, then I moved on to creating a side-navigation for in between the context pack and learners, back buttons for certain pages that our team thought was necessary, and adding in the ability for a user to export a single JSON file that contains all of the learners' assigned context packs.