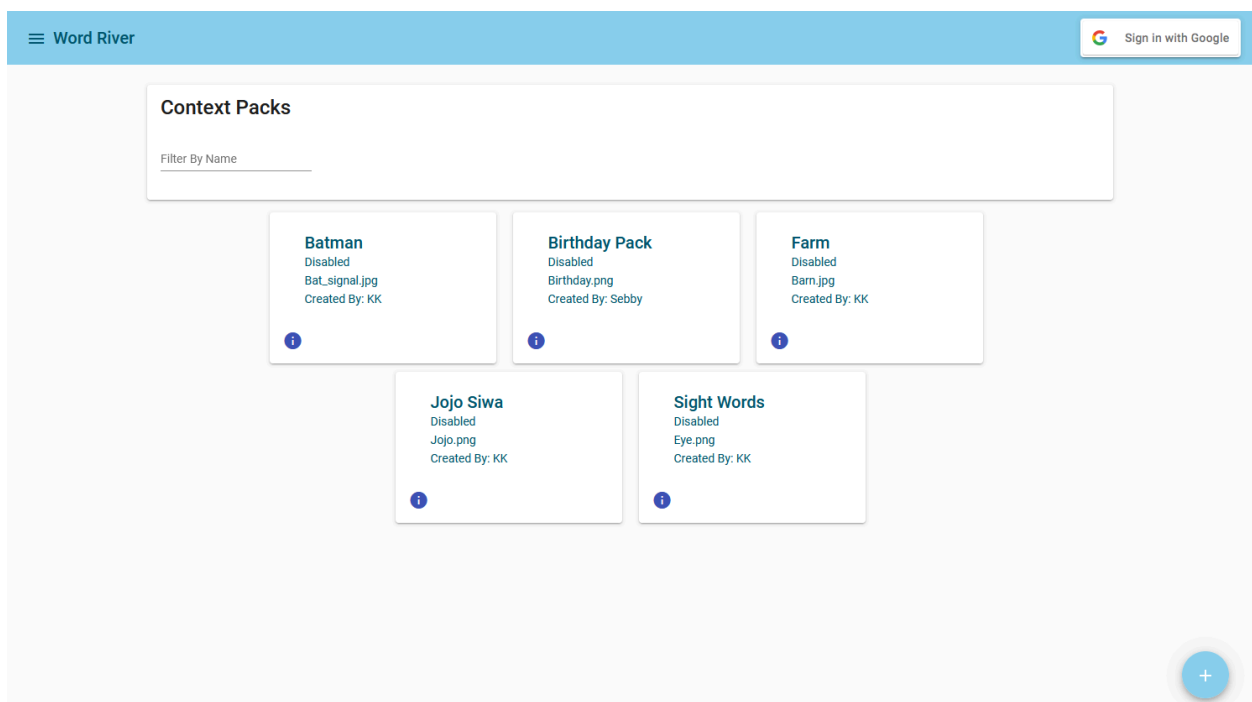




What is Word River?

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.

Sight Words

Disabled

Eye.png

Created By: KK

View:

All Word Lists

Pre-k

Enabled

Nouns:

here i it me we you

Verbs:

can come find go help is jump look make play run

said see

Adjectives:

big blue funny little my one red two three yellow

Misc:

a and away down for in not the to up where ?

K

Disabled

Nouns:

he she that there they this

Verbs:

am are ate be came did do eat get like must

ran ride saw say want was went will

Adjectives:

black brown four good have new our pretty white

Word River

Hello, Isabelle

LOG OUT

My Learners

Filter Learners:

Filter By Name

Add A New Learner:

Enter New Learner's Name *

+

Jessie

i

↓

John

i

↓

Ashley

i

↓

Thomas

i

↓



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.

Jessie

Assigned Packs Toggle Word Lists Toggle Context Packs

Farm → Farm Animals Farm Equipment

Birthday Pack → Birthday

Sight Words → Pre-k K

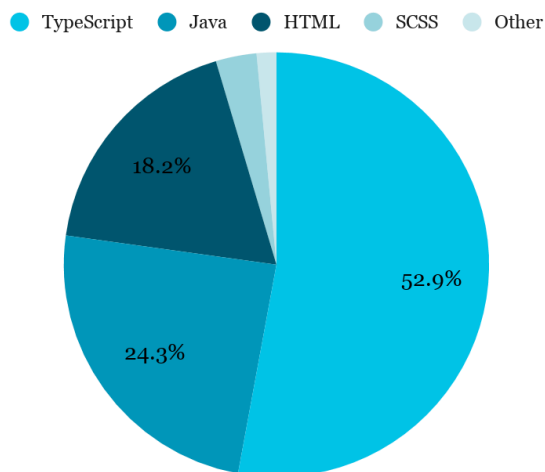
Assigned Words

?	a	all	am	and	are	at	ate	away	baa	balloon	bark	be	big
birthday	black	blow	blue	brown	but	cake	came	can	candle	candy			
card	cat	celebrate	chicken	cluck	colorful	come	confetti	cow	cupcake				
dazzling	decoration	delightful	did	do	dog	down	duck	eat	exciting	find			
food	food	for	four	friend	fun	funny	game	get	give	go	goat		
good	happy	hat	have	he	help	here	I	ice cream	important	impressive			
in	into	invitation	invite	is	it	joyful	jump	laugh	laughter	like	little		
llama	look	make	me	meow	moo	music	must	my	new	no	not		
now	oink	on	one	our	out	out	parent	party	pig	piñata	pizza		
play	play	please	present	pretty	ran	receive	red	ribbon	ride	run			
said	saw	say	see	she	sheep	so	soon	thank	thankful	that	the		
there	they	this	three	to	too	two	under	up	want	was	we	well	
went	what	where	white	who	will	wish	with	year	yellow	yes	you		

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.

Language Use



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.

The screenshot shows a web form titled "Create A New Context Pack". At the top, there are two tabs: "Add Form" (selected) and "JSON-Preview". The form contains the following elements:

- Name ***: A text input field containing "Bob the Builder".
- Icon**: A text input field with a checkbox to its right.
- CREATE WORDLIST**: A button.
- Wordlist**: A section header.
- Name ***: A text input field containing "Bob the Builder's Tools" with a checkbox to its right.
- Nouns**: A section header with an upward arrow icon.
- Word Cards**: Three cards for "Wrench", "Hammer", and "Tool". Each card has a "Word" field, a "Form" field, a red "X" button, and a trash icon. For "Wrench", the form is "Wrenches". For "Hammer", the form is "Hammers". For "Tool", the form is "Tools".
- Verbs**, **Adjectives**, and **Misc**: Three expandable sections with downward arrow icons.
- SUBMIT CONTEXT PACK**: A button at the bottom.

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.

