# Sage Final Report

## Teacher Interface Wireframes & Game Routes

Work by Gavi Rawson

Spring 2018

# Abstract

This paper describes two major improvements to Sage. The first is a redesign of the teacher interface. The new design introduces simplicity, intuitive interaction, and a consistent design, which ultimately increases user understanding of and engagement with the interface. The second major accomplishment is the introduction of Game Routes into Sage Scratch. Game Routes allow teachers to add structure to games to ensure students learn specific programming concepts.

# Introduction

The teacher interface wireframes and Game Routes are two very different concepts and require different attention. Therefore, half of this paper will cover the wireframes, and the other half will cover Game Routes. Each half contains sections on related work, implementation, and future work.

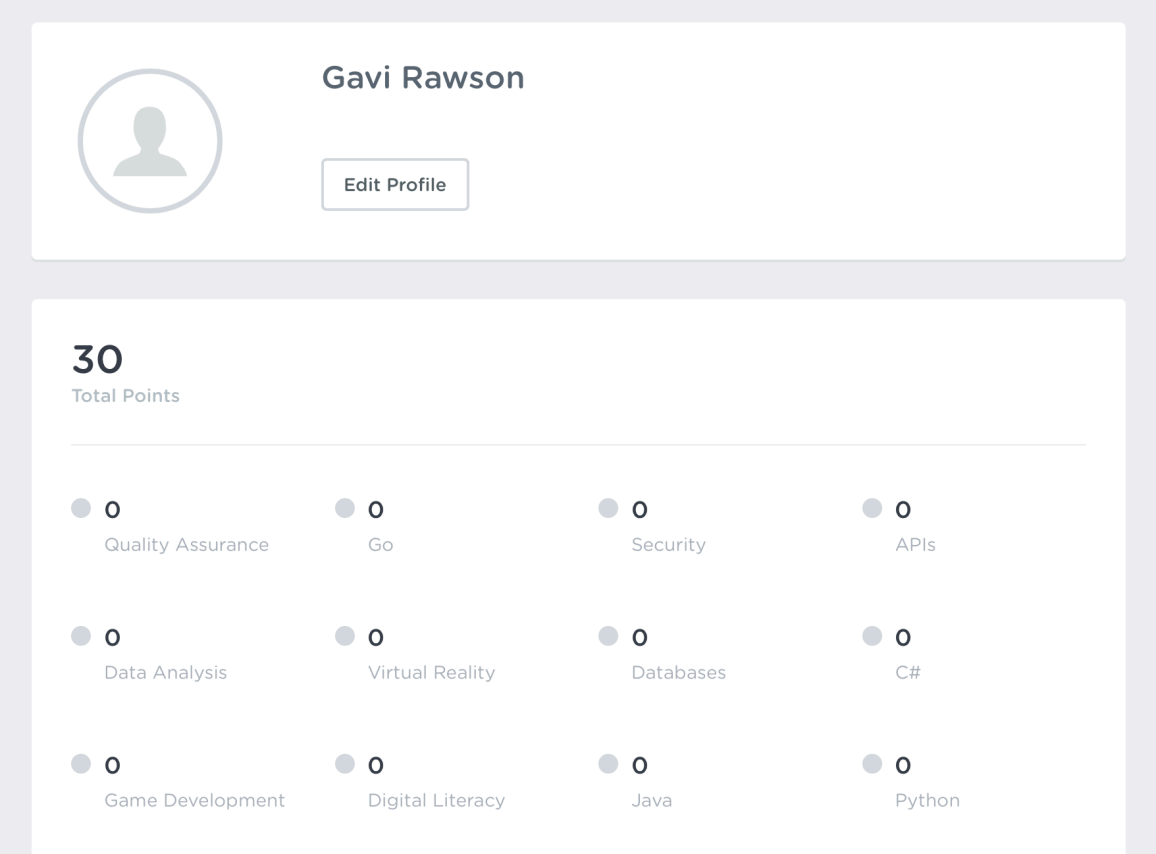
# Related Work: Teacher Interface

Sage is an incredibly useful tool that allows teachers to communicate programming concepts to their students. Teachers interact closely with a desktop application to assign tasks to students and view statistics on learning progression. Sage has an incredible number of features, but it is critical that the interface allows access to these features. Teachers must be able to easily understand and navigate the interface to maximize their teaching potential.

Planning a redesign of the teacher interface involved studying other solutions in production. By visualizing existing successful implementations and methods, we can understand what

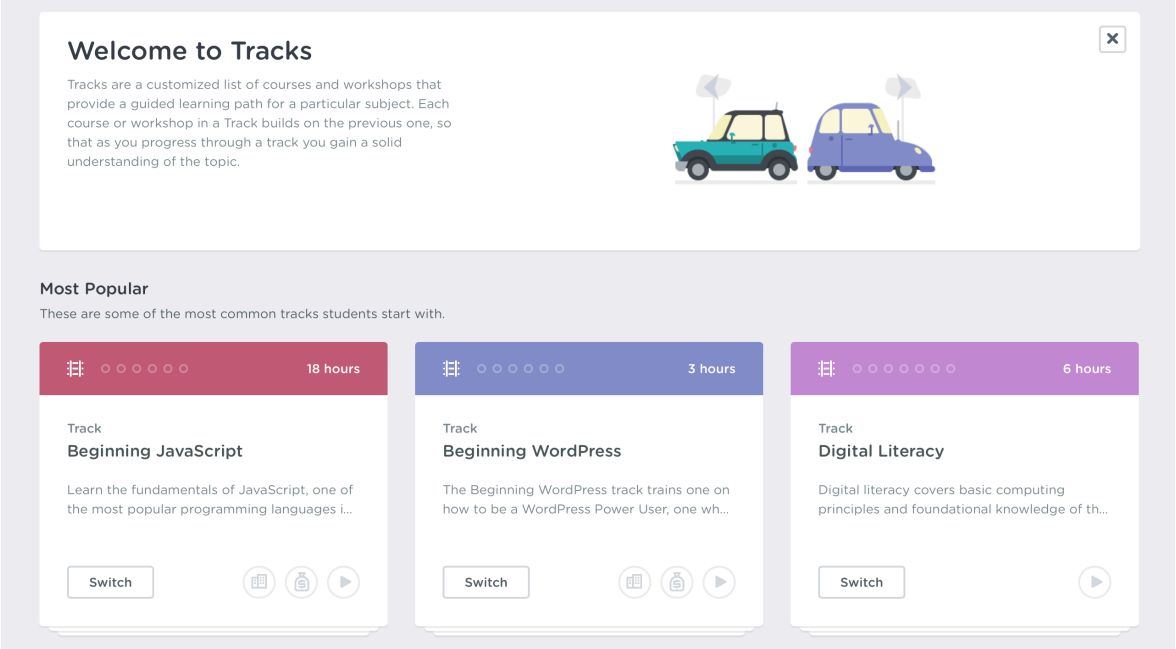
should be added into our implementation to enhance the teaching experience. We can also learn what is unnecessary for our use case and might bloat the application.

Team Treehouse was a major inspiration for the teacher interface redesign. Team Treehouse is a web application that teaches programming to users of all skill levels. The website is beautifully designed and incredibly interactive. UI elements are organized well and are informative enough without cluttering the system. Users can track their progress though courses in an engaging way and are rewarded when they accomplish goals.




Team Treehouse shows overall user progress.

One major strength of the application is its ease of use. When a newly discovered page is opened, the application shows a card with helpful tips for navigating the page. This guides users through the application and minimizes the time necessary to learn the system, to better focus on utilizing the system.



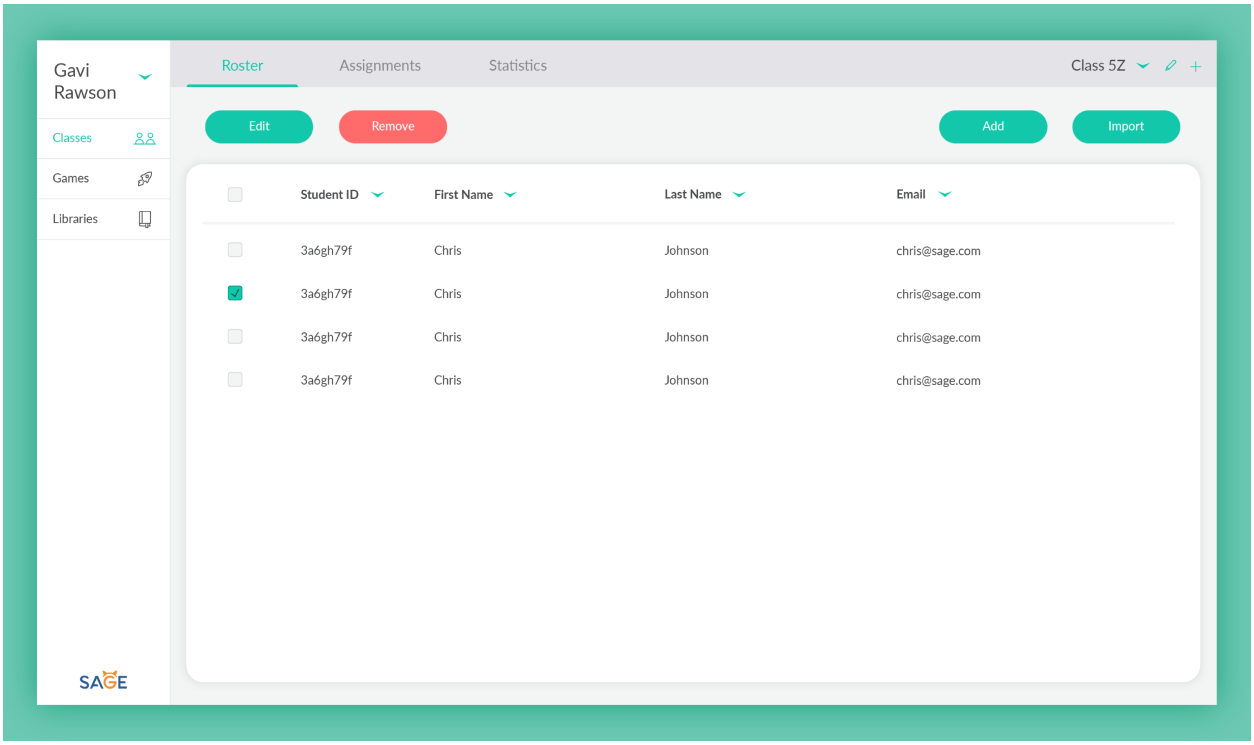

Team Treehouse shows helpful tips on the top of newly discovered pages.

# Implementation: Teacher Interface

A few major overhauls have been done to the teacher interface to encourage a more intuitive and instructive experience. Navigation bars and page structures are rethought to simplify overall movement through the pages. Distracting colors and icons are stripped away and buttons with similar functionality are clustered together. Further, consistent designs are used across modules ensuring learning one page is enough to learn the entire application. Lastly many new features are introduced and instruction is added into pages that contain unintuitive concepts or interactions.

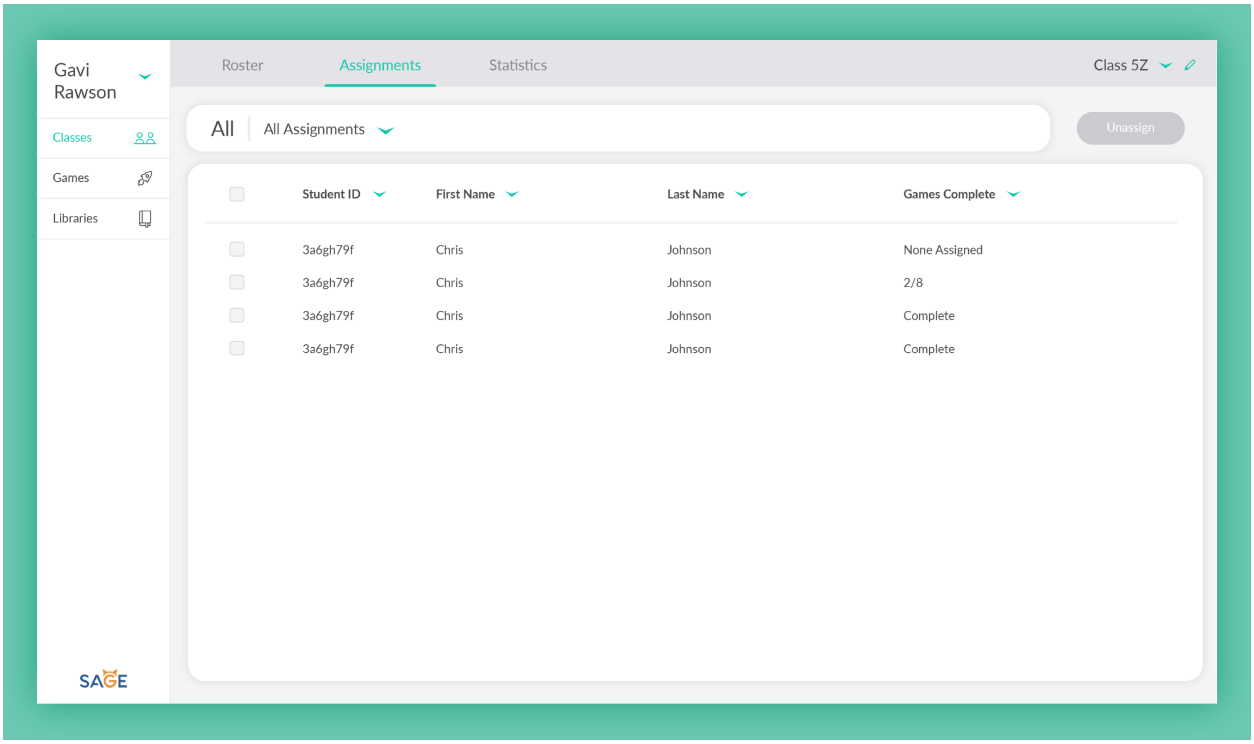
# Classes

The first major addition to the teacher interface is the Classes module. The module introduces an organized way for teachers to view all students in one or many classes they teach. Classes can be created manually or with a csv file to easier add a large class. Managing a roster is as easy as selecting one or many students and using the control buttons on the top of the page.




Roster page in the classes module showing an overview of all the students in a class.

The Classes module also provides an easy way to assign tasks to individual students. A list of all students in the class is always shown on the page. A teacher can select an assignment on the top of the page, and the interface will indicate which students have been assigned to the selected assignment. A teacher can select one or more students and assign or unassign tasks.

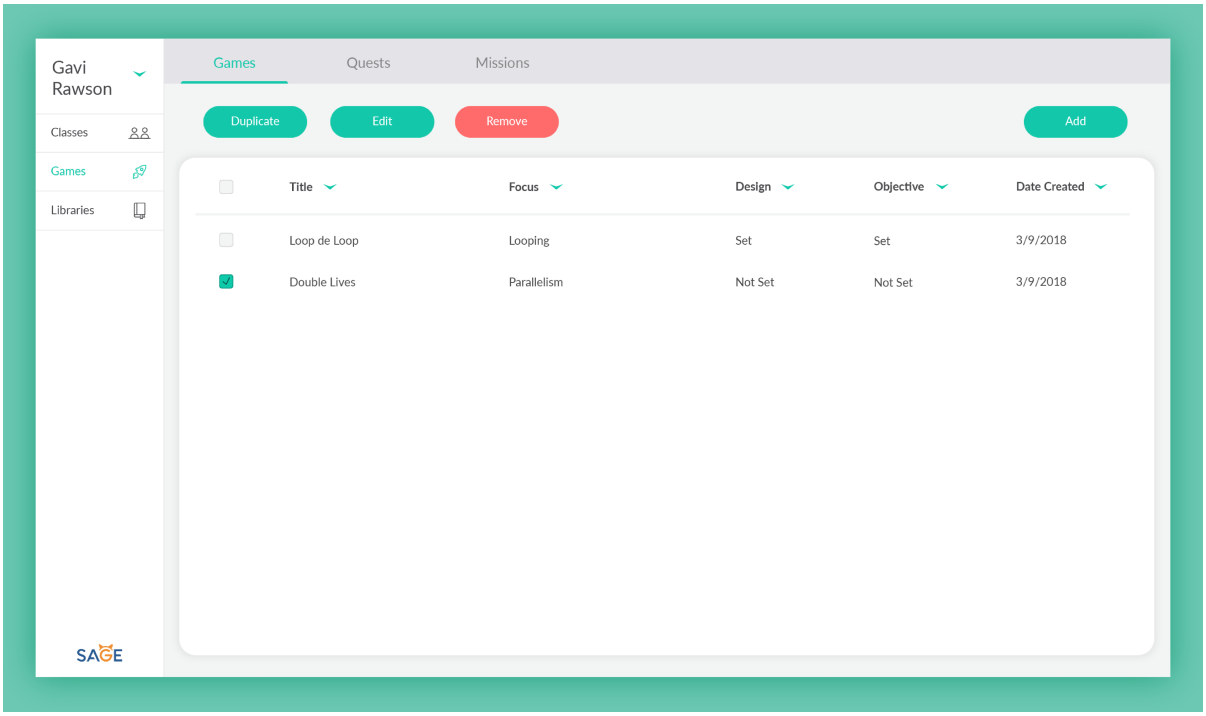



Assignments page in the classes module showing an overview of all students assigned to an assignment.

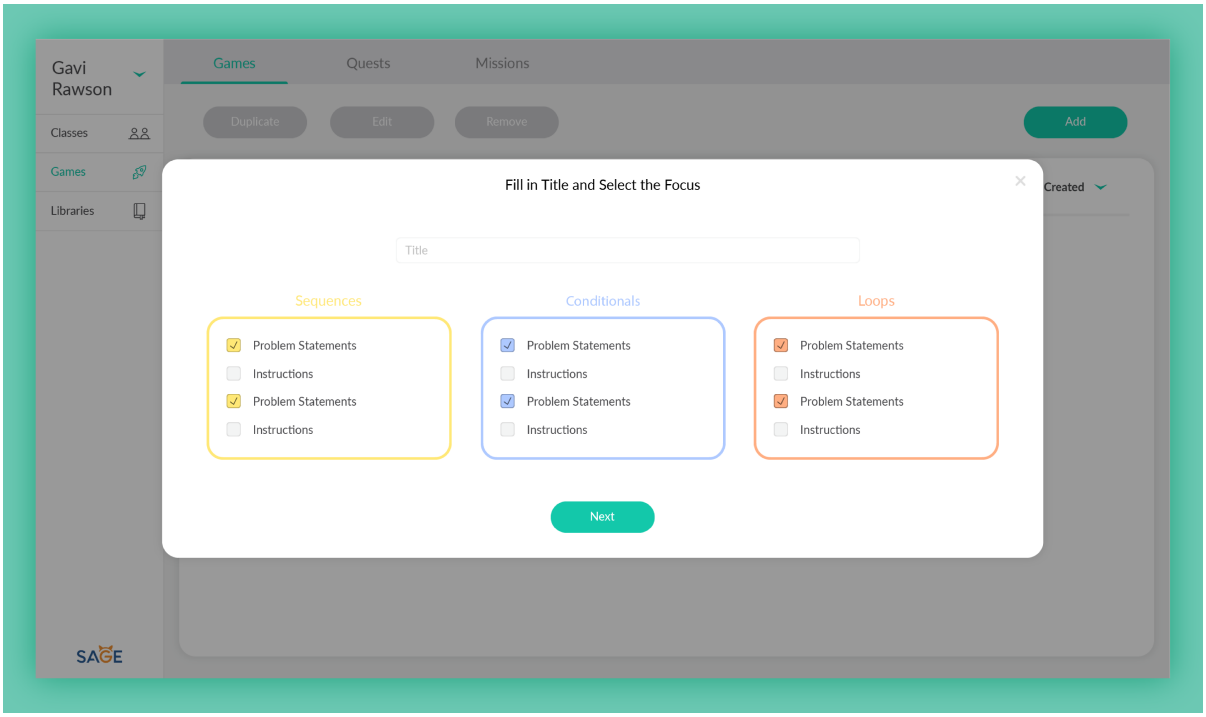
The Statistics page is the last in the Classes module. The page allows a teacher to view in depth statistics on students to better understand learning progression on an individual level. Teachers can easily spot excelling or struggling students to encourage and assist when necessary. The page has been designed by the Learning Metrics team and goes by the name Metrics.

## Games

The second major update is to the process of creating games. In an interface similar to the classes module, teachers can create Scratch games and specify instructions, objectives, and focal areas. Games can also be modified and duplicated to create a robust set of tools for teachers.

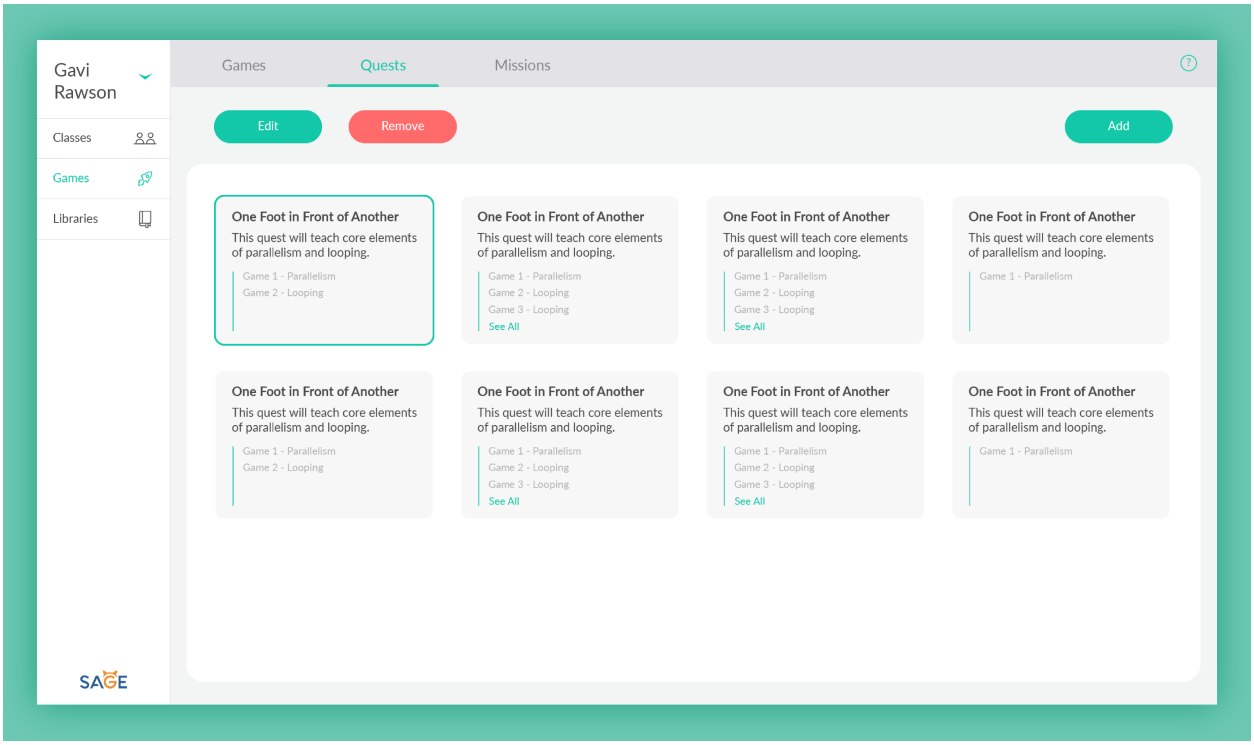



Games page shows the games a teacher has created.



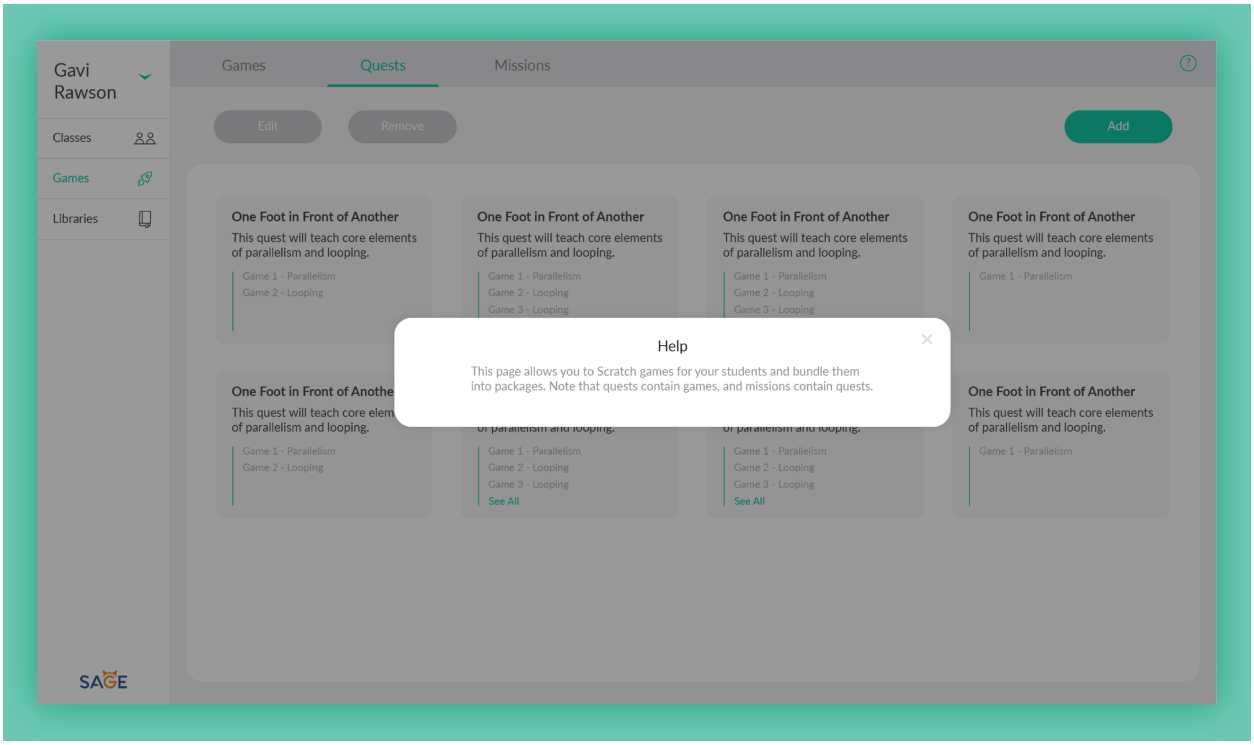

A step in the process of creating a new game. Here the teacher chooses the focus area of the game.

The Quests and Missions pages are also redesigned in the Games module. The new design offers a clean way to associate games with quests and missions to ultimately assign to students. During the creation process, teachers can specify the order of games in quests and missions. Once quests and missions are created, teachers have a bird's eye view of their contents.




Bird's eye view of the quests a teacher has created.

The page also introduces an instructional element. The meaning of and relation between games, quests, and missions is unintuitive for a first-time user. Therefore, there is a question mark on the top of the page that when clicked offers an explanation of the concepts. This instructional element can be extended to other pages as needed to create a more intuitive application.




A popup after clicking the help question mark in the top right corner. The popup explains the concepts on the page.

# Future Work: Teacher Interface

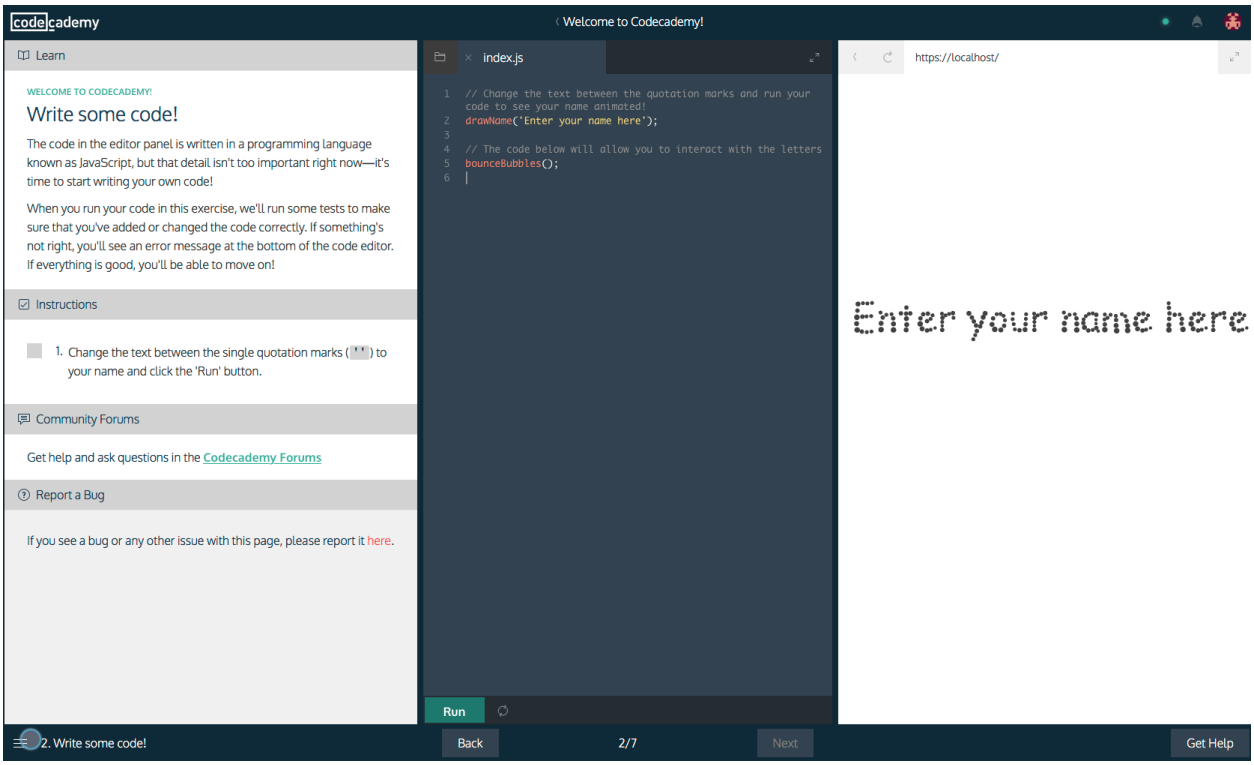
We successfully redesigning the teacher interface, but there is always room for more work. The interface will benefit from more control over a personal profile. Teachers should be able to edit their profile information and set specific settings, such as a reminder system for assigning or grading assignments. Further, the libraries module still needs to be fleshed out, as it is an important way for teachers to discover pre-built Scratch games. But ultimately, now that the teacher interface

is revamped, it is time to apply the same efforts and concepts to the student interface.

## Related Work: Game Routes

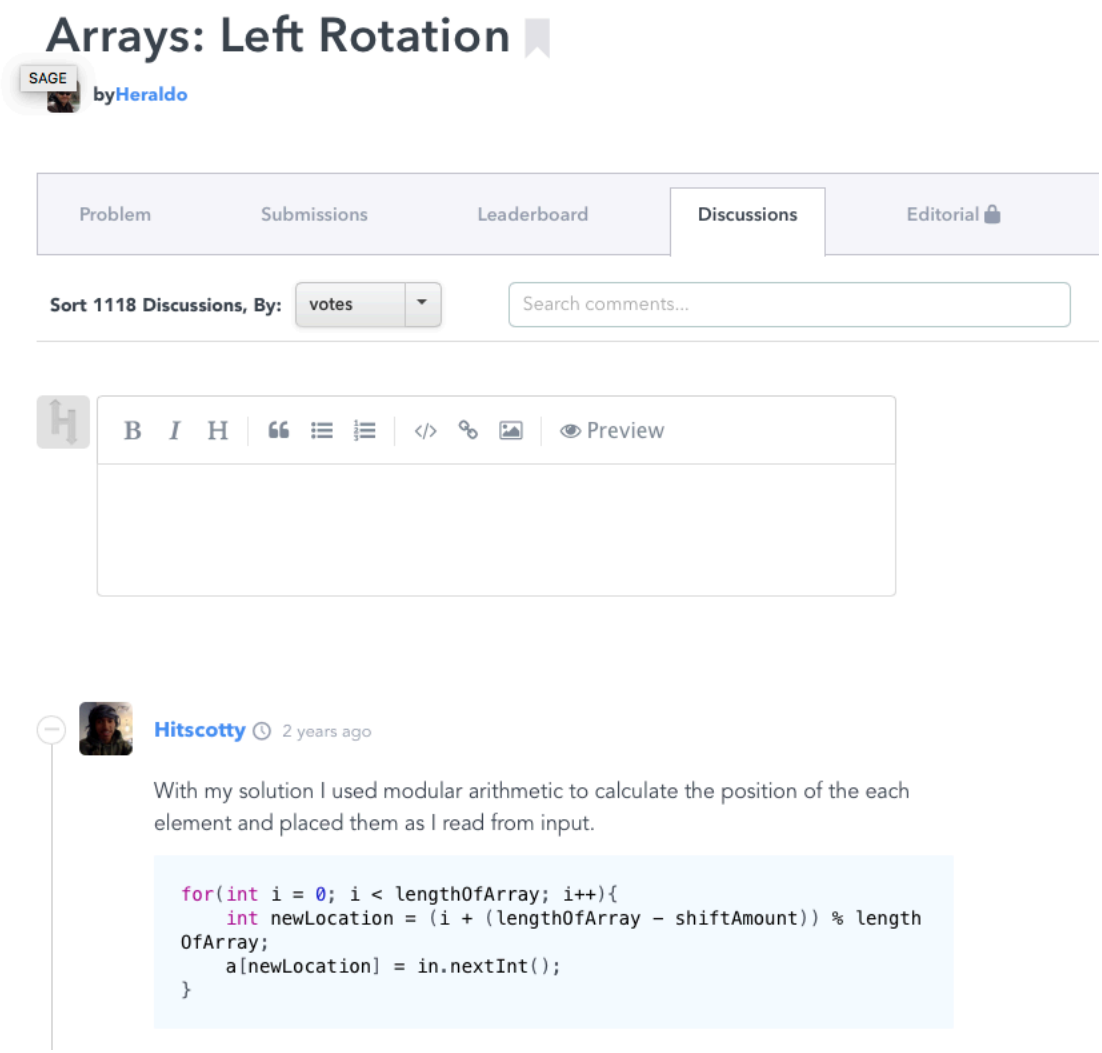
Game based learning is an incredibly engaging method of teaching programming. Many game based learning approaches have games with different constructionist elements. The more construction a game has, the more directed the user will be to solving the puzzle at hand and ultimately learning the targeted concept.

Code Academy is a successful web application that teaches users programming concepts. Code Academy uses a step by step approach in their lessons. Lessons are split into parts with separate instructions. Users complete a part by writing code as described in the instructions and running the program to see their result. This approach guides the users slowly through the programming process. Tasks are not overwhelming because they are introduced in pieces. Further, users have the satisfaction of immediately seeing the result of their work by running their program.




Code Academy offers a step by step instruction process. Instructions are on the left and the run button allows a user to see the result of the code. Steps can be cycled through with the back and next buttons.

HackerRank is another environment that teaches programming. This platform is more for those with prior programming skills that wish to practice targeted concepts. HackerRank provides a programming a question the user must solve in any manner they please. The goal is to write an efficient program in the context of the problem. Since the solution process is very open ended, they provide a discussion section where users can discuss their particular solution. They also provide an editorial section that highlights the editor's particular solution. These features give insight into different possible solutions for the task.




HackerRank provides an editorial and discussion section to provide users insight into potential solutions for the task at hand.

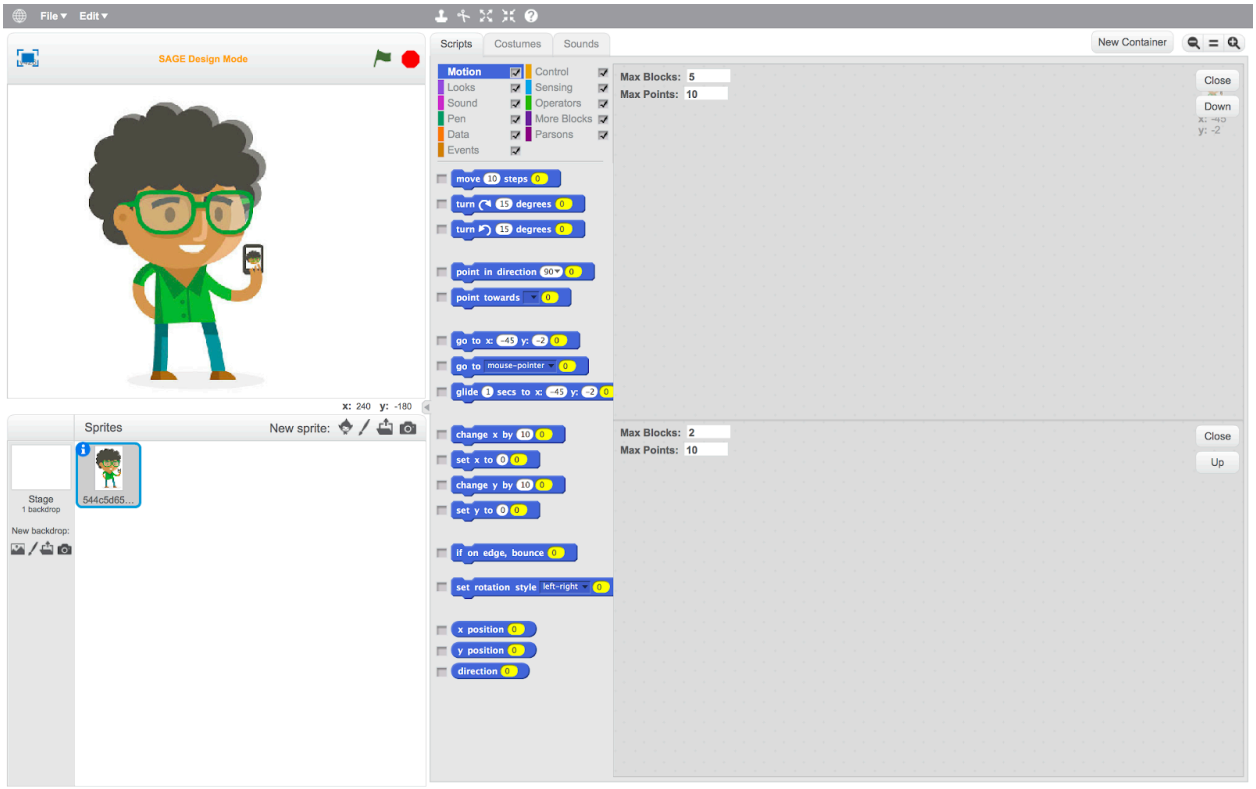
# Implementation: Game Routes

Game routes have been introduced into Sage Scratch in order to add more construction to games. Scratch games do not necessarily have one solution and users are not forced to approach the game in any particular manner. Users can utilize a vast number of blocks to find

a solution to the game. There is one workspace to place these blocks, and each workspace is associated with a selected sprite.

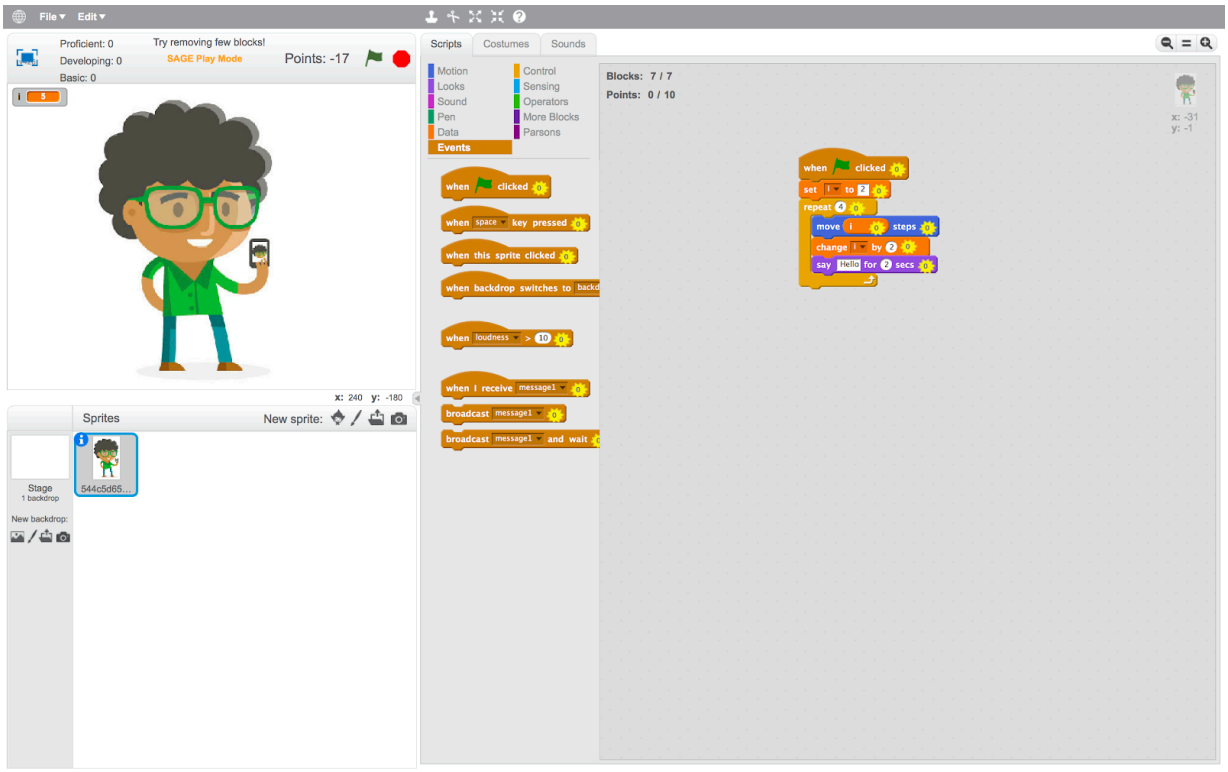
Teachers often want to teach specific concepts, and therefore need the ability to add more construction and direction to games. Game routes provide this constructionism. The concept of a Game Route is to direct students toward a specific route to solving a problem. A Game Route contains one or many workspaces in which students can place blocks. Each workspace can have specific constraints, such as the maximum number of points or blocks that it may contain.

Teachers can begin to construct Game Routes from Sage Design Mode. Once activated a control button appears to create a new workspace. When at least two workspaces have been created, they can each be closed and reordered. Constraints on each workspace are defined by entering values into a workspace's text fields corresponding to max points and max blocks. Each sprite in the game has its own associated Game Route, and thereby its own workspaces and constraints.




Sage Design Mode. Three workspaces are shown on the right. Control buttons and constraints help teachers tailor the workspaces.

Students do not have the same control once Design Mode is turned off. Workspaces and constraints are locked into place. As a student drags blocks into the workspaces, the point and block counts update accordingly. Blocks cannot be dragged into a workspace if either of the constraints fail.




Seven blocks have been added to the workspace. The system will not allow a student to add further blocks without removing one first.

With this new Game Route concept, teachers can direct students towards specific solutions. Say a teacher would like to convey the concept of looping. Previously, students had no constraints on the blocks they may place. If the task was to complete an action four times, they could have approached the problem in a naïve manner and placed many redundant blocks. However, if a teacher places a block constraint on a workspace, similar to the image above, students are barred from using a naïve approach that requires too many blocks. They are pushed towards a specific route to the solution.

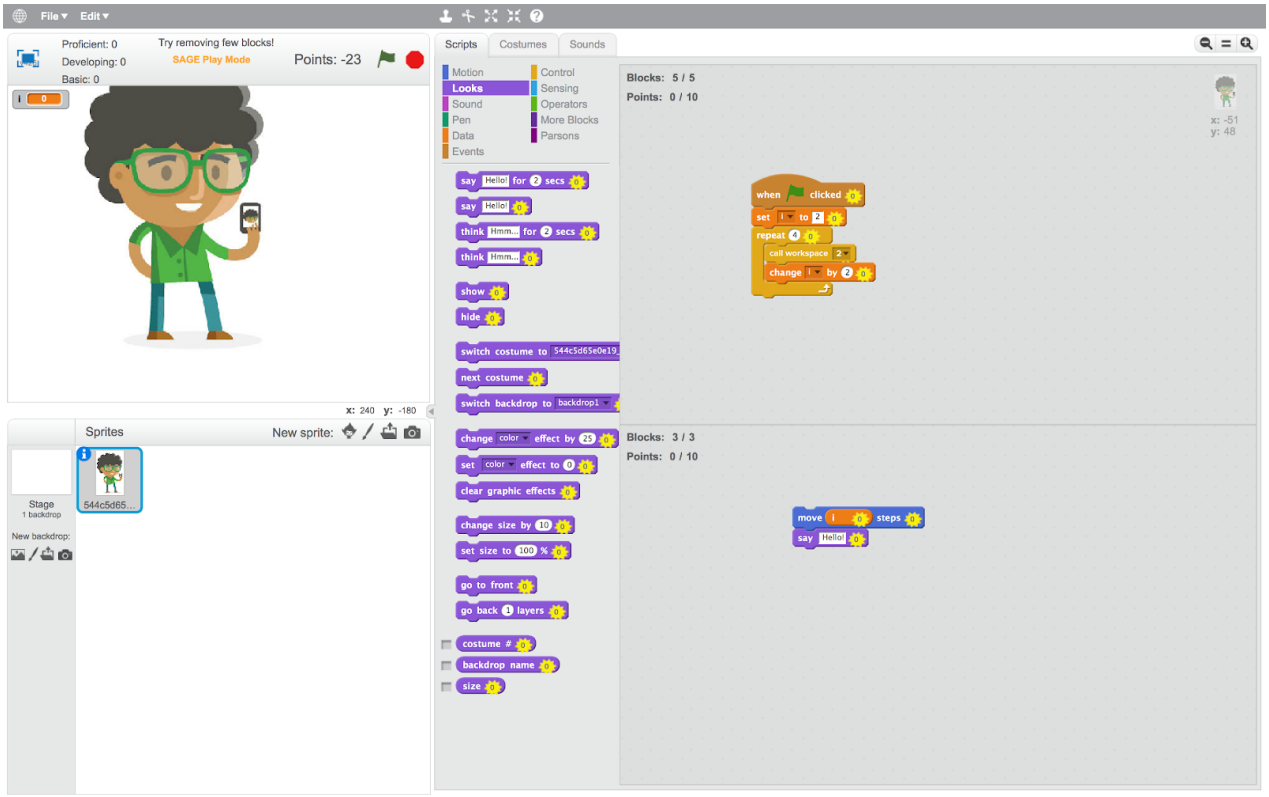
# Future Work: Game Routes

We successfully introduced Game Routes into the Sage Scratch framework. Scratch normally only allows placement of blocks within one workspace for a given sprite. It took much reworking to extend Scratch to allow placement of blocks in many workspaces, and to ensure execution of the blocks worked as expected. Now that the groundwork for Game Routes has been placed, we can start to add more features around it to improve its usefulness.

The first feature is to allow teachers to specify which blocks can be added to a workspace. By restricting blocks, teachers can create targeted games that teach specific programming concepts. This can also help avoid the situation in which a student begins a game and gets overwhelmed by the vast number of Scratch blocks available.

The second possible feature is to add Scratch blocks that allow users to control the interaction between workspaces. Suppose there are two workspaces in the game and each contain a block stack:

- The block stack in the second workspace might have a block on the top of its stack that gets triggered when all blocks in workspace one have finished executing.
- The block stack in workspace one might contain a block that executes all blocks in workspace two. Once the blocks in workspace two finish executing, the block stack in workspace one continues execution. This is similar to a function call.




A new block in workspace one reads “Call workspace 2.” When reached, this block can execute all blocks in workspace two, then continue execution in workspace one. This is a great way to teach the concept of function calls.

The last feature is saving a Game Route. Currently the system only supports saving one workspace. Thus, the number of workspaces and constraints they contain is not saved. We can extend the saving functionality by adding new tags in the output JSON files.

# Conclusion

Both the teacher wireframes and Game Routes are successful building blocks and allow future teams to continue building new features. The teacher wireframes are blueprints for teams that work on the user interface. One team has already begun implementing class administration

into the system using these wireframes. Additionally, now that Game Routes is integrated into the system, future teams can work to build features on top of it to allow teachers to create more targeted games.

# Works Cited

*Team Treehouse*. [Online] https://teamtreehouse.com.

*Hacker Rank*. [Online] https://www.hackerrank.com.

*Code Academy*. [Online] https://www.codecademy.com.