**Team 16: Dragon Drop**

**Sprint 1 Retrospective**

Ancil Trent, Brandon Xu, Charlie Hyun, Lucy Cheng, Mark Lim, Sarah Mi

**What Went Well**

User Story #1

* Original attempt to create a user class didn’t go well
  + React works with components primarily
  + Components have states that can be difficult to mutate between different files
  + Resolution: Switch from a class to a collection of functions that other components can call

User Story #2

* Accessing user data / checking credentials
  + MERN stack makes the process simple
  + Some difficulty learning how to handle data returned from a database call
  + Once a full understanding of the process was gained, both these tasks were smoothly implemented

User Story #3

* Checking password / account existance
  + Thanks to knowledge gained in user story 2 this was easily done
  + Confirming account exists is as simple as checking if database request returns null
  + password check is not done in a “secure” fashion, correct password is given to the client to check against entered password
  + In future sprints more security will be implemented

User Story #4

* UI for Dragon Drop home page
  + We were able to implement basically everything that we had imagined with the initial homepage
  + Looks very similar to UI mockup we had made

User Story #5

* Creating components
  + We now have a framework for creating new components (component, componentTools, componentView, etc.)
  + Components can be dragged + dropped, can click component to view options, clicking component shows a panel
  + Still need to work on resizing, overlapping, etc. but the core functions are mostly there
* Button component
  + Button component works, with both functionality and editable text.
  + Thanks to existing draggable components, implementation was quickly learned.
  + Used the built in libraries for button styling.

User Story #6

* Download
  + Basic image downloading functionality was easily implemented.
  + Menu options were all available and viewable to user.

User Story #7

* Saving functionality
  + Able to capture the current status of the page when a user clicks save
  + Status of page connects to database (however not fully implemented, discussed under what did not go well section)
    - Learned to send and retrieve the image as a base64 string

User Story #8

* Maximum page count
  + Check how many pages are returned when requesting all pages under given user
  + If equal to current page count, don’t allow additional to be created
  + Easy to implement
* Page class
  + As with user class, ended up going with functions instead
  + Very similar to user functions, database calls going to page collection instead of user collection

User Story #9

* Page Deletion
  + This entire story went well, with all criteria met and tests working
  + Project’s visual disappears from page through React state updates
  + Database calls for deletion were implemented
  + Deletion popup worked well using toastify library

User Story #11

* Public/Private setting
  + Setting public/private and writing to database
  + Learning how to write to the database with this task helped with other cases where I also had to write to the database.
* Searching
  + Searching for users was implemented
  + Decided to filter data after querying the database. I ran into issues trying to filter during querying

User Story #12

* Dragging new textboxes on the edit page was implemented
  + Was able to successfully integrate components from a different library into our own project
* Rich text editor menu
  + Tool menu was successfully implemented on right clicking a text box component, the menu included bold, italicize, and underline buttons
  + Learned how to bind buttons to specific functions and how to listen to a right click
  + Learned how to create a new class component inside a different component

User Story #13

* User Page
  + UI pulls pages dynamically from the database depending on user logged in
  + Create new page button has functionality
  + Project Previews can be interacted with and have a variety of options
  + Homepage button is easily seen in top navbar
  + Overall it has a lot of good functionality and design.

User Story #14

* Edit Page Image component
  + Can add images to the edit page
  + Can upload images using a web hosted image link and pressing a button
  + Can change the image once an image is uploaded.
  + Figured out how to send information from the component panel to the actual component

User Story #15

* Shapes
  + Conditional rendering upon choosing a different shape went well, which is the main difficult part of the shape.

User Story #18

* Undo/Redo
  + Library for edit page “BuildUI”
  + BuildUI creates a node tree of all components and their values
  + This makes implementing undo/redo significantly easier
  + Call to undo goes through tree and delete the latest change
  + Call to redo goes through tree and restores the change
  + Node tree creates a built in undo/redo history
  + Also handles only allowing undo/redo when their are changes

**What Did Not Go Well**

User Story #5

* Overarching component class was not created since that is not how React and build-ui’s setup works.
* Button’s functionality is minimal.

User Story #6

* The rest of download functionality was not able to be covered in this story.
  + HTML / CSS and scripting functionality
  + Being able to run the download as a website
  + Renaming and Duplication were also not implemented.

User Story #7

* Sending saved data to database
  + We were not able to retrieve the HTML/CSS code to save to and retrieve from the database
  + Sending and retrieving project preview snapshots also not fully implemented yet
    - We were unable to get the unique project IDs to save information to
    - Currently hardcoded a project ID that shows the saved image

User Story #10

* Color tool/selector
  + Created the tool with color wheel, RGB, Hex, and dropper functionality
  + However, was not able to connect the actual inserted components to the color
  + Was unsure how to send data from child component (selected color) to parent (selected editing component)

User Story #11

* Disabling editing and toggling privacy settings for other user’s projects was not disabled.
  + Ran out of time connected the UI render with an if-statement

User Story #12

* Modifying text
  + Created tools for bolding, italicizing, and underlining text but was not able to connect to the user input from textbox
  + Rich text editor is a separate class component from the Text component, and we were not able to figure out how to send the input text data from one component to the other

User Story #13

* User Page
  + Visually not the best looking
  + Code architecture behind page preview is confusing and not uniform
  + Can be buggy when accessed the wrong way (without logging in)
  + Progress was made between three different people at much different times
  + Much of the UI is not polished

User Story #15

* Shapes
  + Difficult to drag around
  + Awkward resizing
  + No functions on click
  + Only three current shapes

User Story #16

* Dragging components
  + BuildUI makes this difficult
    - components are not React components but instead constant objects
    - This makes mutating values of the components such as size and position difficult
  + Resolution: Used a separate movement library and wrapped it around BuildUI
  + Allows component movement
  + However, BuildUI is unaware of the movement of components
* Keyboard arrow presses
  + Possible with an arrow key listener and the movement library
  + Simply didn’t have enough time to implement
* Manual position data edit
  + Toolbar is a property of BuildUI
  + BuildUI can’t see component movement
  + Result: This feature could not be implemented with current setup
* Movement locking
  + Same issue as manual position editing

User Story #17

* Resizing components
  + Same issue as movement and same resolution with library above BuildUI
  + Resizing is very buggy
    - Edges of components are not visible for dragging
    - Only works with some components (doesn’t work with shapes)
    - “Edges” of components do not line up with the contents of the component
* Manual size editing
  + Same issue as movement
  + BuildUI toolbar doesn’t know the size of components
* Maintain aspect ratio
  + Same issue as manual size editing

User Story #19

* Overlapping components
  + Movement was initially locked, so components were not able to overlap
  + Could not start on any of the tasks until movement was fixed

**How to Improve**

* User stories
  + As we continued to work, we realized that our user stories did not cover all required tasks. In order to complete task A, we realized we had to set up task B, C, D to even start A. This took up a lot of time that did not go towards our listed user stories.
  + During the presentation with our grader, it was hard to show a fluid walkthrough of the entire application since our user stories were in random order. We found that we were jumping around a lot. For the next sprint, we will try to order our user stories so that they flow sequentially.
  + We found that we had accidentally created a lot of user stories/tasks/acceptance criteria that overlapped with each other and it caused a lot of confusion within the group of who was doing what task.
  + Towards the end of the sprint, we began to deviate from our assigned user stories. In the next sprint, we will work to try to assign user stories that make sense together to each member so we can stick to user story assignments.
* Github
  + Merge with main more frequently
  + Too many branches: team members had multiple branches and weren’t making frequent pulls into main
    - Some members’ code broke and it was too difficult to revert or fix.
  + Our code ended up very fragmented until the last week
* Productivity
  + Meet deadlines
  + Set meeting lengths in order to be more productive during our meetings
  + A lot of sprint work was done at a single time right before the week’s deadlines, causing issues of not being able to communicate issues with team members until the last second
* Testing
  + All tests were manual for Sprint 1
  + Including simple unit tests will net us additional points for little work cost