**System Test Scenarios**

Scenario 1: Create Account (Pass)

1. Start Emergent Playground
2. Navigate to “Login” on the navbar
3. Click “Don’t have an account? Register”
4. Register for account
   1. Username: “test”
   2. Email Address: “test[@gmail.com](mailto:ethan@email.com)”
   3. Password: “test”
   4. Confirm Password: “test”
5. Check navbar for username

Scenario 2: Login (Pass)

1. Start Emergent Playground
2. Check the navbar says “Login”
   1. If not, click “logout”
3. Click “Login” on navbar
4. Log into account
   1. Email Address: “[test@gmail.com](mailto:test@test.com)”
   2. Password: “test”
   3. Click “LOGIN” button
5. Check navbar for username and logout

Scenario 3: Draw (Pass)

1. Start Emergent Playground
2. Click on black background
3. Check that the place clicked now has a white pixel

Scenario 4: Square Draw (Pass)

1. Start Emergent Playground
2. Click brush icon of left toolbar
3. Select the square icon
4. Click on black background
5. Check that white square was drawn on the background

Scenario 5: Square Sizing (Pass)

1. Start Emergent Playground
2. Click brush icon of left toolbar
3. Select the square icon
4. Click arrow icon on right side of the screen
5. Navigate to “VIEWER OPTIONS”
6. Click on the value for Brush Size
7. Type in “2”
8. Click once on the background
9. Check that the background now has a 2 by 2 white square where clicked

Scenario 6: Continuous Play (Pass)

1. Start Emergent Playground
2. Click arrow icon on right side of the screen
3. Navigate to “VIEWER OPTIONS”
4. Click “Continuous Play” to enable it
5. Draw on the canvas
6. Check that the simulation did not pause

Scenario 7: Play (Pass)

1. Start Emergent Playground
2. Draw on the canvas
3. Click the play icon on the left toolbar
4. Check that the state of the canvas has changed

Scenario 8: Preset CA (Pass)

1. Start Emergent Playground
2. Draw on the canvas
3. Click the play button
4. Observe the behavior of the current rule
5. Click the arrow icon on the right side of the screen
6. Navigate to “PREMADE RULES”
7. Click on “Brian’s Brain”
8. Draw on the canvas
9. Click on the play button
10. Check that the canvas changes following a different rule

Scenario 9: Language (Pass)

1. Start Emergent Playground
2. Click the arrow icon on the right side of the screen
3. Navigate to “LANGUAGE”
4. Write working code
   1. Define states
      1. Open State Definition
      2. Input “ffffff” as color and name it live
      3. Input “000000” as color and name it dead
   2. Implement Game of Life in code window
   3. Press “COMPILE”
   4. Build well-known Game of Life constructs on canvas
      1. Use Wikipedia as a reference
      2. Build the same in playgameoflife.com
      3. Observe correct behavior in playgameoflife.com
   5. Press play button to the left
   6. Compare behavior between the two
5. Clear out the code window
6. Press “COMPILE”
   1. Check if popup error is displayed
7. Write erroneous code
   1. Write “uint x = 0;”
   2. Press “COMPILE”
   3. Check if the canvas breaks
   4. Check console to see errors
8. Write working code with strange colors
   1. Follow same steps as 4
   2. Change colors for live and dead to something random
   3. Do same comparison

Scenario 10: Save CA (Pass)

1. Log into account
2. Navigate back to Canvas
3. Draw shapes onto canvas
4. Click the title field in the middle of navbar
   1. Type in “TEST SAVE”
5. Click the “SAVE” button to the right of the title
6. Click the user’s username link in the navbar
7. Check that there exists a post titled “TEST SAVE”

Scenario 11: Load CA (Pass)

1. Log into account
2. Navigate back to Canvas
3. Draw shapes onto canvas
4. Click the title field in the middle of navbar
   1. Type in “TEST LOAD”
5. Click the “SAVE” button to the right of the title
6. Click the user’s username link in the navbar
7. Click the post titled “TEST LOAD”
8. Check that the shapes are the same as previously drawn

Scenario 12: Community Page (Pass)

1. Log into account
2. Click the Community link in the navbar
3. Check that posts by other users can be seen

Scenario 13: Community Post Load (Pass)

1. Log into account
2. Click the Community link in the navbar
3. Click on another user’s post
4. Check that the new state was loaded into canvas

**Sprint 1:**

1. Animated: Cellular Automata rules were tested to ensure that they animated properly
2. Premade: Premade rulesets were tested for accuracy, performance, and were tested to ensure that they could be easily swapped between
3. Fullscreen: Tested to ensure that the fullscreen button makes the browser fullscreen
4. Pen: Tested swapping between different pen types to ensure that the crosshair changed
5. Zoom: Tested to ensure that zooming the canvas could be done in real time without affecting the cellular automata animation

**Sprint 2:**

1. Pause: Tested to ensure that pausing the state of the cellular automata could be completed without affecting the state and could be resumed from the pause
2. Shader Lang: tested by using the shader language to implement the premade rules
3. Welcome/About: tested to be navigable to in the frontend and appearing properly in browser
4. Save: tested by swapping states and ensuring that they could be resumed easily without affecting the automata

**Sprint 3:**

1. Account: Tested to ensure that an account could be created, logged into, logged out of, and that the owner of the account was able to use it to access their own rulesets and not be able to access other account’s rulesets as well as being able to publish posts by using this account
2. Color Picker: Tested with a variety of rulesets to ensure that the color picker worked consistently and properly
3. Properties\_UI: Tested to ensure that all aspects of the properties tab allows for the user to access the associated functionality of the corresp
4. Community Post: Tested to ensure that posts could be displayed in a list format on the community page in addition to the links to the posts working as intended with users being able to sort posts properly according to their selected sorting method
5. Error Message: Ensured that error messages appear on log in errors

**Sprint 4:**

1. Post to Canvas: Tested to ensure that posts from the database are able to be accessed by the frontend in the community posts section of the site
2. Saved Posts: Tested by clicking on the populated links in the community posts section and ensuring that every link corresponded with the correct post in addition to being able to access the post in the same state that it was saved under.

**Unit Tests**

**Testing Report**

**Testing Methodology**

Testing was conducted manually across different browsers, operating systems, and hardware with a focus on targeting the systems and software that the majority of internet users are likely to use.

During development, pair programming, peer review, and continuous testing were used frequently whenever possible considering time constraints. Pair programming was conducted via in person coding sessions. During the pair programming sessions, team members would alternate between writing the code and reviewing the previously written code. By alternating the person reviewing the code in this manner we were able to find more bugs via differing perspectives. Modules were tested for functionality as they were being written and were rewritten as necessary to fix issues that came up. Additionally, interfaces were tested for functionality as soon as they were in a state where it was possible to do so.

When testing manually, code was tested with an expected functionality in mind and if it did not perform according to this expected functionality, this would be addressed instantly.

**Testing Limitations:**

* Time Constraints
  + We were unable to utilize Unit Testing as we decided to prioritize coding time to ensure that the product would be in a useable state
  + Instead, we opted to use less comprehensive testing that could potentially result in more bugs slipping through
* Testing pool constraints
  + Testing was only conducted by the immediate members of our team
  + Outside testers could potentially find issues that may have been overlooked due to the relatively small team size
* Useability testing constraints
  + Since testing was only done by the members of the team who are familiar with cellular automata and how to use our site, it is impossible to know how the average internet user would react to the UI and general useability of the site

**Conclusion:**

In spite of testing limitations and time constraints, the main features of the site that we wished to implement are all currently working to a level where a user could reasonably use the website for the purpose of creating and sharing their own cellular automata rulesets.