SAGE Final Report

# Learning Metrics

Ruimin Zhao rz2390

Neng Chen nc2734

**Abstract**

Most of our progress were completed based on close discussion with our mentor during weekly meeting. The related user stories lie in the Gameful Affinity Space Epic - Learning Metrics Feature - user stories related to progress bar, spider graph, and hairball analysis. Concretely implementation and outcomes mostly lie in mLab database entities relation clarification and modification, front-end code modification so as to aggregate the metrics features onto master account and without any conflicts to other features like adding classes, and some hairball package exploration as well as modification work for our game result assessment.

December 19, 2018

## 1 Introduction

Computational thinking has become a hot topic in teaching computer science. There are arguments saying that computer science is about the study of information, and computational thinking concept could might not necessarily result in programming language but could be a good way to see the possibilities there [1]. The SAGE project is trying to develop such a tool to utilize the gamefied way to teach students computational thinking and thus programming [2].

There are lots of work with machine learning and deep learning involved and many full-stack development. Here for our team, we are mainly focusing on the full-stack development port which is visualizing the students' game results score in a graphical way. We worked on a codebase in NodeJS using AngularJS, and its connected cloud-based MongoDB database. We also worked on the modification on package to make it usable for our customized cases during our second half of journey.

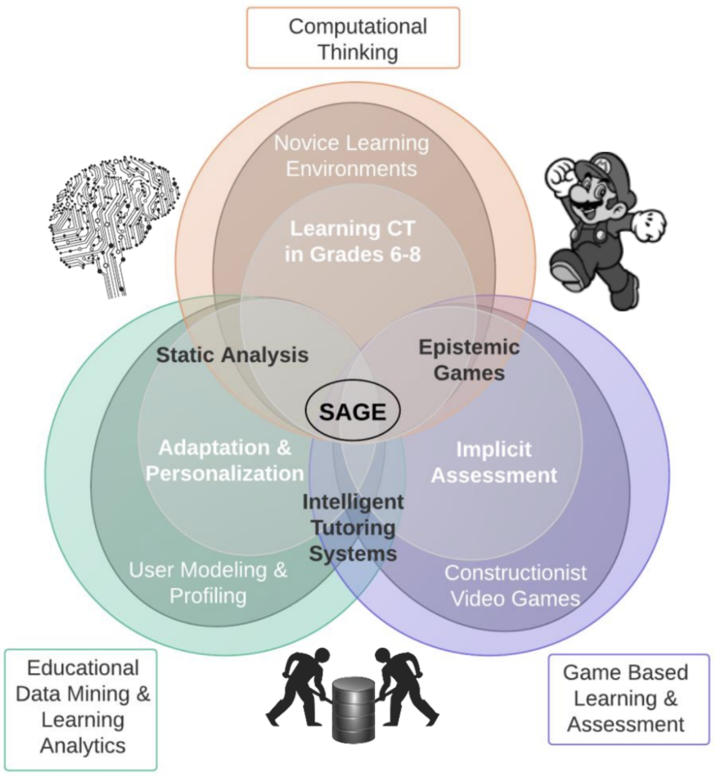
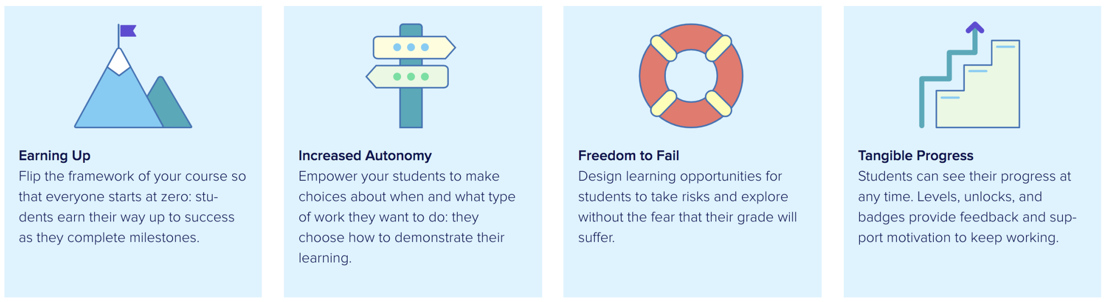

Computational Thinking
Novice Learning Environments
Learning CT in Grades 6-8
Static Analysis
Epistemic Games
SAGE
Adaptation & Personalization
Implicit Assessment
Intelligent Tutoring Systems
User Modeling & Profiling
Constructionist Video Games
Educational Data Mining & Learning Analytics
Game Based Learning & Assessment

Figure 1: SAGE

## 2 Related Work

In terms of gameful teaching or learning tool for students to learn computer sience or programing language only, there are already some reasonably mature tools out there. Two examples can be GradeCraft [3] and CodeAcademy [4]. They do share some common features with our SAGE project in terms of the portion of work we focused on - learning metrics. For example, the GradeCraft also provide progress tracking feature in its main functionalities, and CodeCademy provides structured courses which is also contained in our project when the learning metrics is displayed in a structured way consistent with our game structure - games within different missions.



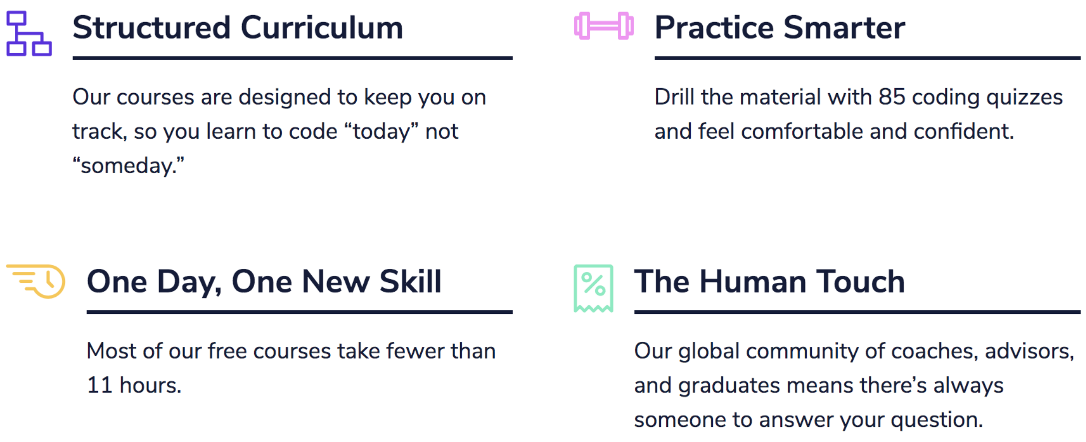
**Earning Up**
Flip the framework of your course so that everyone starts at zero: students earn their way up to success as they complete milestones.

**Increased Autonomy**
Empower your students to make choices about when and what type of work they want to do: they choose how to demonstrate their learning.

**Freedom to Fail**
Design learning opportunities for students to take risks and explore without the fear that their grade will suffer.

**Tangible Progress**
Students can see their progress at any time. Levels, unlocks, and badges provide feedback and support motivation to keep working.

Figure 2: Gradecraft



**Structured Curriculum**
Our courses are designed to keep you on track, so you learn to code “today” not “someday.”

**Practice Smarter**
Drill the material with 85 coding quizzes and feel comfortable and confident.

**One Day, One New Skill**
Most of our free courses take fewer than 11 hours.

**The Human Touch**
Our global community of coaches, advisors, and graduates means there’s always someone to answer your question.

Figure 3: Codecademy

# 3 Background

## 3.1 Architecture

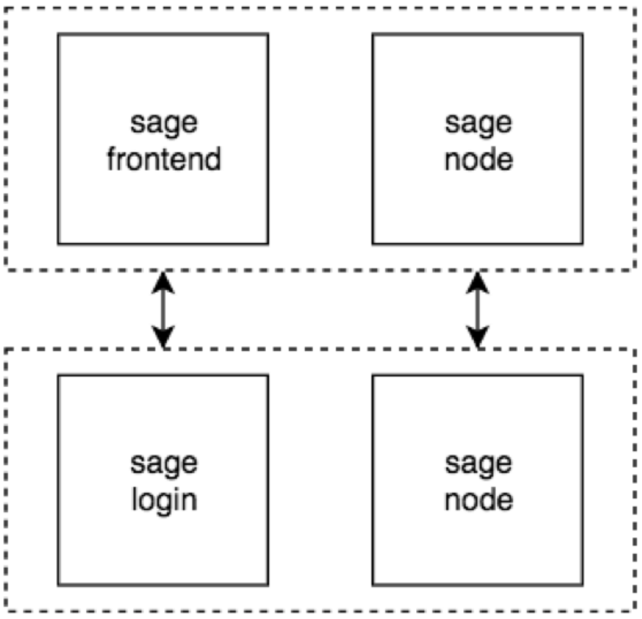


Figure 4: Learning Metrics Architecture

- User usage: We allow users to register either as student or teacher to use this web application.
  - Student: Students could log in as a student and then click on Metrics option from the navigation bar and see their own learning progress clearly.
  - Teacher: Instructors could log in as an instructor and then click on Classes option from the navigation bar, and then chose Metrics option and see the enrolled students' overall learning progress in an aggregated way.
- Codebase: For this project, the involved repositories and cloud-based MongoDB (mLab) database relationship can been in Figure above.
  - sage-frontend repo  sage-login mLab: The code for triggering the Learning Metrics graphs (progress bar, spider graph, and badges earned) lie

in here and it send requests to mLab to extract the metrics data from the corresponding collection called studentmetrics.

– sage-node repo  sage-node mLab: The related collections all lie in sage-node database within the mLab hosted mongoDB server. The collections related to games lie in here. Ideally in the future, we might want to move the learning metrics collection into here or make its information available from the game collection hosted here.

– Plugin: The hairball plugin should be used to assess the game result after a game is finished and the score obtained from the hairball should be saved into database so that the metrics graphs actually show real score that make sense rather than hardcoded as what is done for now.

## 3.2 Problems

There are concretely three problems to solve and the progress milestones follow this path as well: we tried to solve these three problems one by one in summary.

- First problem to solve: Integrate the learning metrics graph into master account without any conflicts with other features.
- Second problem to solve: Modify the hairball package so that it could generate game result on our sage sb2 file.
- Third problem to solve: Trigger the hairball assessment and store the real score for metrics graph to use (ongoing).

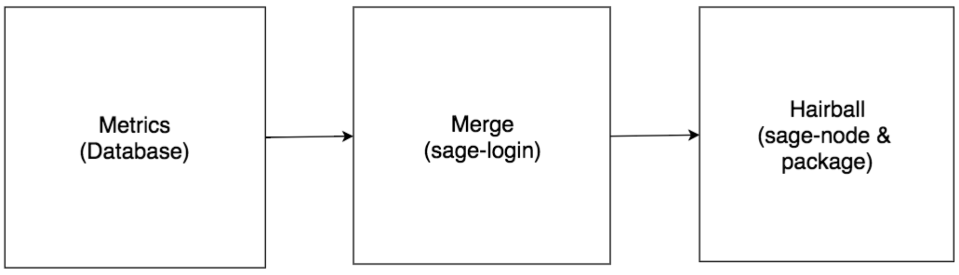
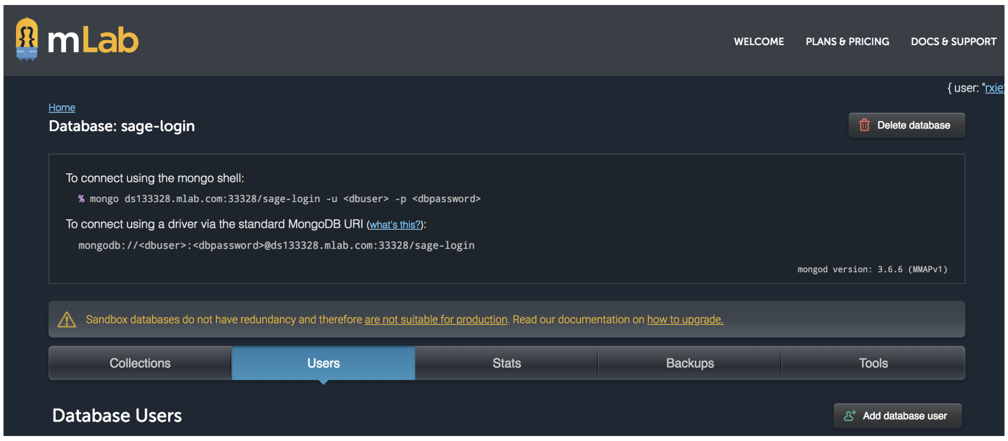


Figure 5: Milestone

# 4 Implementation

## 4.1 Learning Metrics

- Metrics Reproduce:
  - Outcome: As described in the link `https://gudangdaya.atlassian.net/wiki/spaces/SAG/pages/579436651/Learning+Metrics+Badges+Reproduce`, a step-by-step guideline for reproducing the learning metrics progress bar and spider graph in master accounts rather than customized accounts could be seen [5].
  - Process: The reason why we were doing this was that initially, the previous team completed the learning metrics code and related database set up, however, the graph generation could not be done in the shared master account in our whole project and potential conflicts with other existing ongoing development in master account have not been solved. As a result we started to solve this first problem by looking into the codebase for learning metrics and the related database.



Database: sage-login

To connect using the mongo shell:

```
% mongo ds133328.mlab.com:33328/sage-login -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard MongoDB URI (what's this?):

```
mongodb://<dbuser>:<dbpassword>@ds133328.mlab.com:33328/sage-login
```

```
mongod version: 3.6.6 (MMAPv1)
```

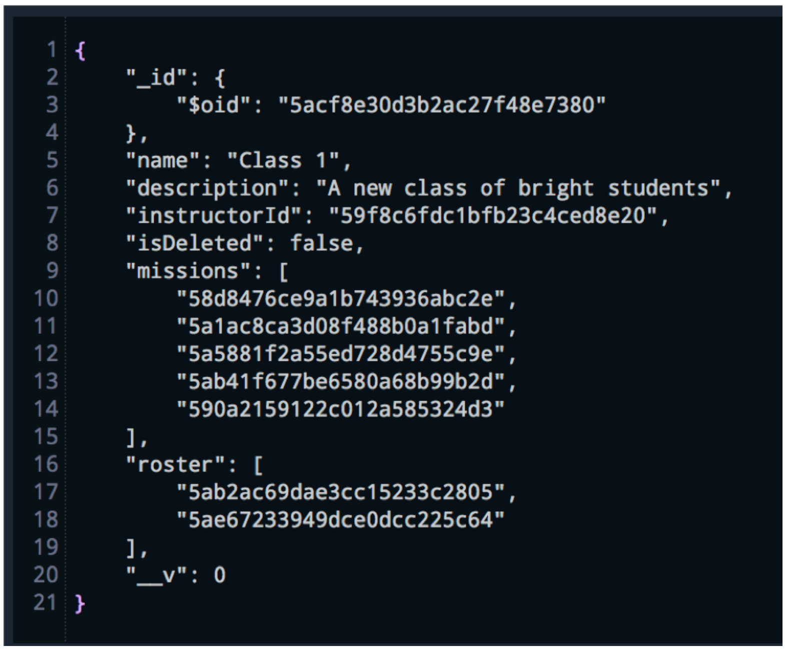
Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on how to upgrade.

Database Users

Figure 6: SAGE database

  - Findings: Due to the lack of the game result generation UI, the score were manually inserted into the database and there are more than 3 collections related to one graph, which took us some time to figure out the correct schema to follow so as to reproduce them on master account. However, these usage of NoSQL database has potential issue due to its lack of developer-defined primary key. And this was actually one of the biggest block when we tried to aggregate the metrics feature with the class feature.

- Merge Metrics into Classes:
  - Outcome: We made the metrics graphs able to be shown on master account at the first stage and then we cleared the database collections and fixed some errors (warning) in codebase to make this feature not conflicting with another feature developed by another team.
  - Process: We moved the Metrics feature into Classes option to avoid duplicate places holding metrics feature. We found out that the class collection was used by both metrics feature and the class enrollment feature and these two features have different expectation on the entity schema. As we mentioned above, the database we used is NoSQL but the way we coded seems to treat those collections as SQL data. For example, we expect for one instructor, there should be only one entity containing all the information related to class. However, in reality, the database class collection has more than one entities for a single instructor ID (one for class enrollment, and one for metrics generation). This was one of the biggest blocks we met with during the process of solving this second problem. We solved this issue by merging the two entities and keep all the attributes in one entity so as to meet the requirement for both two features.



```json
{
    "_id": {
        "$oid": "5acf8e30d3b2ac27f48e7380"
    },
    "name": "Class 1",
    "description": "A new class of bright students",
    "instructorId": "59f8c6fdc1bfb23c4ced8e20",
    "isDeleted": false,
    "missions": [
        "58d8476ce9a1b743936abc2e",
        "5a1ac8ca3d08f488b0a1fabd",
        "5a5881f2a55ed728d4755c9e",
        "5ab41f677be6580a68b99b2d",
        "590a2159122c012a585324d3"
    ],
    "roster": [
        "5ab2ac69dae3cc15233c2805",
        "5ae67233949dce0dcc225c64"
    ],
    "__v": 0
}
```

Figure 7: Class collections

## 4.2 Hairball

- Run Hairball on sage sb2 file: The Hairball package itself is an open-source package written in Python [6]. It was designed to get assessment results on sb2 file generated by D Scratch [7]. We took a closer look at the library itself to re-factor the hairball library to make it work for standard sb2 file provided by the scratch-analyzer repository first. And after that, we tried to understand the hairball package source code better and able to start to actually manipulate the code to make it able to assess our sb2 game result. This has took quite a process of trails and errors since the package source code itself is a new repository and we should also learn how plugin work etc. so as to understand the current pipeline designed and then re-factor it.

Welcome to scratch! :D
by LovelyLadybird100

Instructions

So press the green flag and space for the next slide!

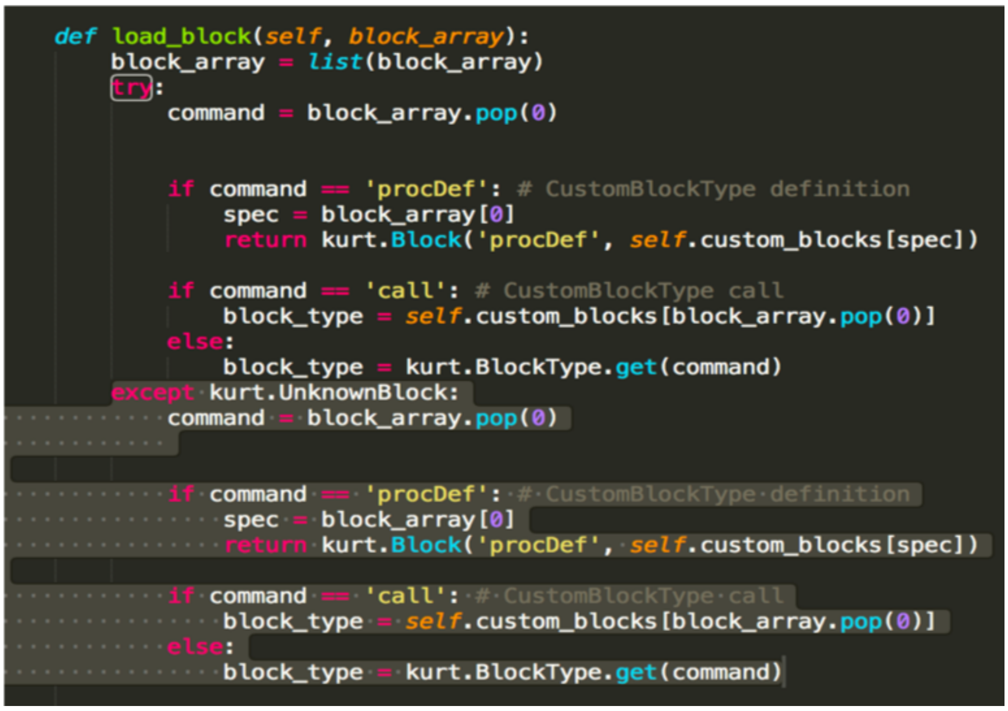
Notes and Credits

Scratch is an awesome place so let's keep it that way! :D
As I said it the project have fun and just be yourself!
Feel free to ask any questions! :)
Be creative
- @LovelyLadybird100

Shared: 5 Aug 2015 Modified: 5 Aug 2015

Figure 8: Scratch

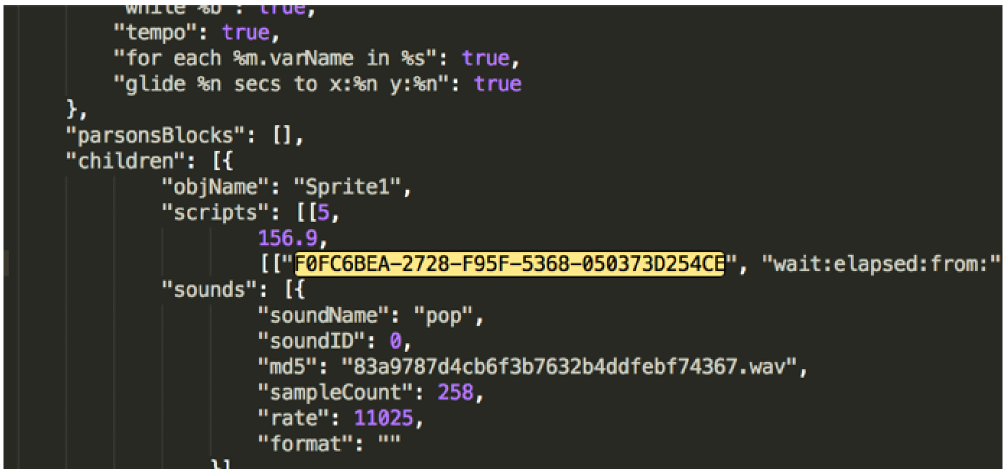
We modified the Kurt library after a series of trails and errors when trying to run the Hairball on sage sb2 file. We mainly added some exception handling to add the ability to solve our particular structure of json file (contain game results which is required by the Hairball) as can be seen in Figure 9.



```python
def load_block(self, block_array):
    block_array = list(block_array)
    try:
        command = block_array.pop(0)

        if command == 'procDef': # CustomBlockType definition
            spec = block_array[0]
            return kurt.Block('procDef', self.custom_blocks[spec])

        if command == 'call': # CustomBlockType call
            block_type = self.custom_blocks[block_array.pop(0)]
        else:
            block_type = kurt.BlockType.get(command)
    except kurt.UnknownBlock:
        command = block_array.pop(0)


        if command == 'procDef': # CustomBlockType definition
            spec = block_array[0]
            return kurt.Block('procDef', self.custom_blocks[spec])

        if command == 'call': # CustomBlockType call
            block_type = self.custom_blocks[block_array.pop(0)]
        else:
            block_type = kurt.BlockType.get(command)
```

Figure 9: Hairball-Kurt modification

The reason why we had to make this modification was that our sage sb2 added a unique ID for each game result json file for the purpose of fulfilling the needs from the machine learning and deep learning portions of work. However, the Hairball package initially cannot handle this special structure, and we had to add a portion of code to handle this structure for our customized json structure.



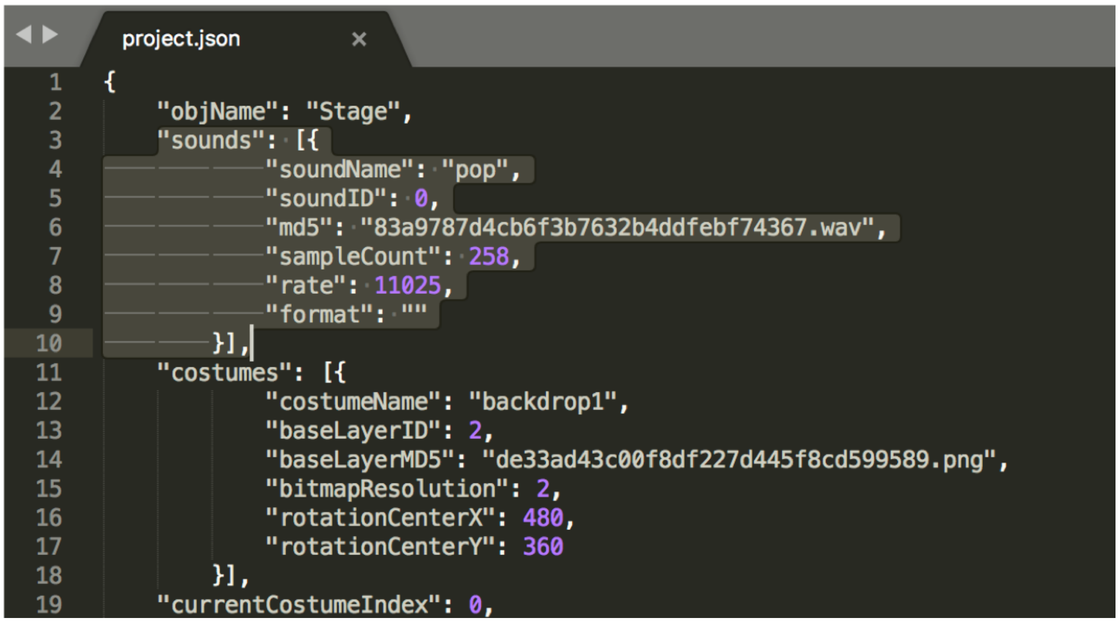
```json
        "write %b": true,
        "tempo": true,
        "for each %m.varName in %s": true,
        "glide %n secs to x:%n y:%n": true
    },
    "parsonsBlocks": [],
    "children": [{
            "objName": "Sprite1",
            "scripts": [[5,
                    156.9,
                    [["F0FC6BEA-2728-F95F-5368-050373D254CE", "wait:elapsed:from:"
            "sounds": [{
                    "soundName": "pop",
                    "soundID": 0,
                    "md5": "83a9787d4cb6f3b7632b4ddfebf74367.wav",
                    "sampleCount": 258,
                    "rate": 11025,
                    "format": ""
```

Figure 10: SAGE json unique ID

- Run Hairball on json file: After managed to make the Hairball run on sage sb2 file, we then tried to optimize this pipeline further - to avoid keeping all the redundant files such as images and sound files generated all together with that core json file. This is for the purpose of speeding up the frequent uploading and downloading process of that sb2 file.

  We tried several approaches to modify the Kurt library to make it able to run a json file rather than expecting for a zip sb2 file. However, this approach took us quite along time and seemed to be not very possible since the initial process of Kurt dealing with a income sb2 file was very complicated and it was virtually imporrsible to bypass the upzip step. Therefore, we chose another approach - to keep a sb2 file as the input but deleted all the files except that json file. In this way, we could make less change on the library.

  We focused on the json parsing process in Kurt library. And we found that the json file has some related sounds and costumes attributes which was the core reason why the Kurt library is expecting for those sounds and images files even after we deleted them from the sb2 file. As a result, we deleted these two attributes from the json file and we do not get errors anymore when we run Hairball on that shrinked sb2 file. And the score we got remaines the same as the sb2 file without any files deleted. In the future, we should make this manually deleting file process as program and keep that portion of code together with the sb2 generation process.


project.json

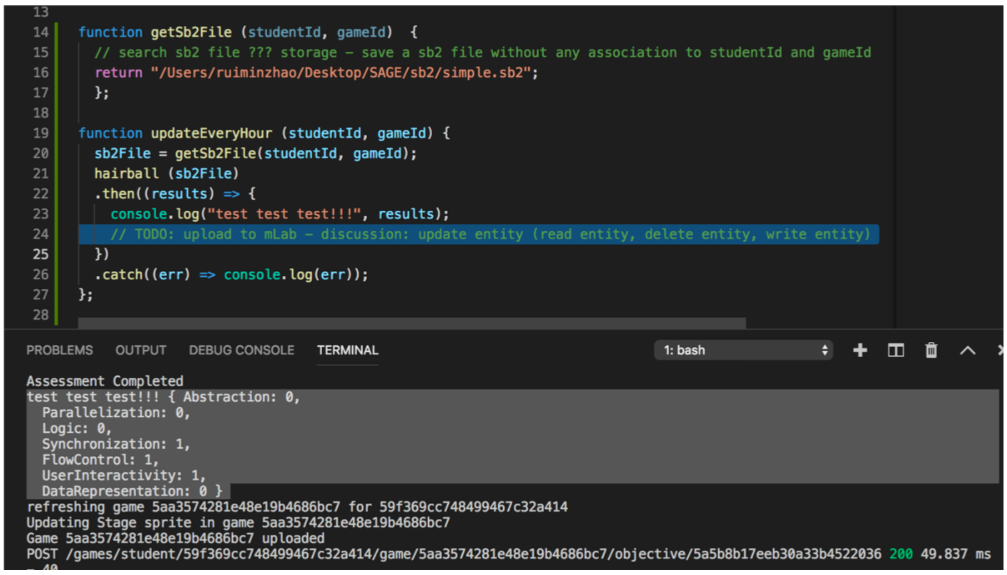
```
{
    "objName": "Stage",
    "sounds": [{
            "soundName": "pop",
            "soundID": 0,
            "md5": "83a9787d4cb6f3b7632b4ddfebf74367.wav",
            "sampleCount": 258,
            "rate": 11025,
            "format": ""
        }],
    "costumes": [{
            "costumeName": "backdrop1",
            "baseLayerID": 2,
            "baseLayerMD5": "de33ad43c00f8df227d445f8cd599589.png",
            "bitmapResolution": 2,
            "rotationCenterX": 480,
            "rotationCenterY": 360
        }],
    "currentCostumeIndex": 0,
```

Figure 11: SAGE json sounds and images

## 4.3 Assessment Trigger

- Interfaces: After we modified the Hairball related libraries and make it able to get real score, we then started to find the correct way to trigger the assessment process so as to get that real score in program and then save them into metric collection so that our learning metrics graphs could be generated based on real score from real assessment rather than being hardcoded. We added three interfaces for this purpose with still some hardcode path or file name in it due to our reliance on the sb2 submission feature. This could be crucial future work.

- Trigger Assessment: But for now, basically, we trigger the assessment every second when the student is in a game. The seven score for computational thinking results can be obtained successfully as can be seen in Figure 12 in code terminal. In the future, we want to save these score into metrics collection once we finalize the place to keep metrics collection and how to merge the two process - game submission  sb2 storage and our game result assessment on that uploaded sb2 result.



```
function getSb2File (studentId, gameId)  {
  // search sb2 file ??? storage - save a sb2 file without any association to studentId and gameId
  return "/Users/ruiminzhao/Desktop/SAGE/sb2/simple.sb2";
  };

function updateEveryHour (studentId, gameId) {
  sb2File = getSb2File(studentId, gameId);
  hairball (sb2File)
  .then((results) => {
    console.log("test test test!!!", results);
    // TODO: upload to mLab - discussion: update entity (read entity, delete entity, write entity)
  })
  .catch((err) => console.log(err));
};
```

```
Assessment Completed
test test test!!! { Abstraction: 0,
  Parallelization: 0,
  Logic: 0,
  Synchronization: 1,
  FlowControl: 1,
  UserInteractivity: 1,
  DataRepresentation: 0 }
refreshing game 5aa3574281e48e19b4686bc7 for 59f369cc748499467c32a414
Updating Stage sprite in game 5aa3574281e48e19b4686bc7
Game 5aa3574281e48e19b4686bc7 uploaded
POST /games/student/59f369cc748499467c32a414/game/5aa3574281e48e19b4686bc7/objective/5a5b8b17eeb30a33b4522036 200 49.837 ms
```

Figure 12: Assessment trigger interfaces

## 5 Conclusion and Future Work

A brief conclusion about this project would be that our work was pretty much related to making some back-end findings and modification for database collections relationship and libraries change so that the future team could have the available tools (such as a correct collection structure for master account and a working Hairball-Kurt library on sage sb2 without redundant files as well as a start point for game assessment triggering process connecting the game submission).

However, there are still some concrete future work to document down.

- sb2 file trigger: We should make the sb2 file uploading process and the sb2 file assessment process consistent with each other in the future.
- learning metrics database: We might want to merge the metrics required data into the game collections hosted in sage-node database on mLab. In this way, the game assessment trigger code and metrics generation code could stay in the same place, which make more sense in terms of structure.

## 6 Reference

1 Computational think concept in computer science https://computationalthinkingcourse.withgoogle.com/unit

2 SAGE project https://github.com/cu-sage/About/blob/master/sage_primer.pdf

3 GradeCraft https://www.gradecraft.com/

4 CodeCademyhttps://www.codecademy.com/

5 Learning Metrics Reproduce Documentation https://gudangdaya.atlassian.net/wiki/spaces/SAG/pages/579436651/Learning+Metrics+Badges+Reproduce

6 Hariball Package https://github.com/jemole/hairball/tree/master/hairball/plugins?from=singlemessage&isappinstalled=0

7 D Scratch https://scratch.mit.edu/projects/72329908/