SAGE Dashboard: Final Report

Ruicong Xie and Tianci Zhong
COMS E6901, Section 14 and COMS 3998 W Section 14
Fall 2016

Table of Contents

1.0 Introduction

Animation and Gameful programming has been a key movement for educating young people with elementary programming skills and introducing computational thinking skills. There are many animation programming environment, AgentSheets (Repenning, 1993), Alice (Cooper, Dann & Pausch, 2000), and Hands (Pane, 2002) before Scratch (Resnick et al., 2009) came out and become popular. The Scratch environment is a popular tool and community for primary and middle school students learning basic programming skills and improving computational thinking. The Scratch started as a tool for teaching after school students elementary programming technique under the guidance mentors, which gained really positive feed. The students experienced great improvement on both the width and breadth of the function they written. The research done by Christopher Scaffidi and Christopher Chambers (Scaffidi & Chambers, 2012) shows a mixed results of Scratch's learning environment as it moves toward a boarder online users without assistance and guidance of mentors. The drop out rate among Scratch users is extremely high with a median duration of 3 months involved in the Scratch environment. Among the ones who do not drop off, there is an improvement of their social interaction skills, but the learning curve is relatively flat. This result reflects the currently Scratch environment and community are unable to keep their users engage for a long period time and do not have effective guidance to help improve users computational thinking skills and break their limits. This problem motivates us to improve the current environment of Scratch to become more engaging with contests and more helpful with instructors' guidance.

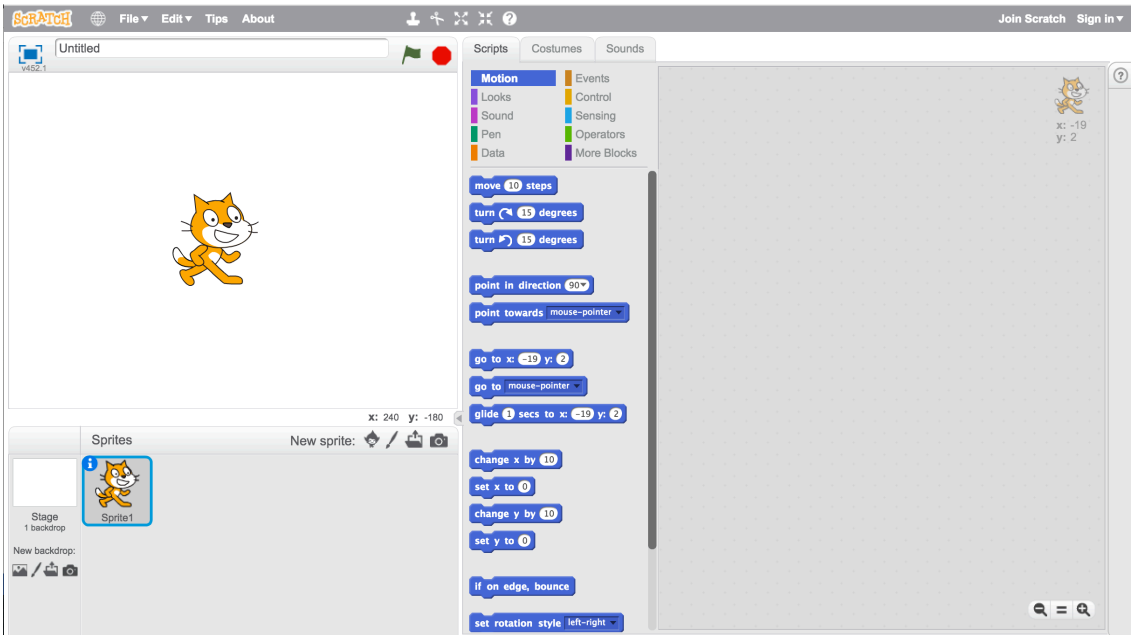

Figure 1. Scratch Interface

Scratch is the underplaying platform for SAGE (Bender, 2015), which aims to provide information for instructors to monitor students' progression for each assignment and more constructively adjust the assignment they created. It also aims to provide a game-based learning system for primary and middle students to improve their computational thinking skills. This project is focused on the dashboard element for the SAGE system.

Computational thinking is an essential concept driven the development of SAGE. Computational thinking is not just an essential skill for computer scientist. It also has been considered as one of the fundamental skill that is critical to every child's analytical ability (Wing, 2006). Computational thinking involves solving abstract problems, decomposing large complex problem, approximating problems' difficulty, forecasting potential risks and issues, considering the worst case scenarios, controlling the uncertainty and thinking recursively (Wing, 2006). Many instructors are motivated to incorporate computational thinking into their teaching methods, but they are lack of reliable ways to assess this kind of skill.

This project is motivated by the difficulty of quantitatively present the learning progression of students and quantify their degree of computational thinking. It is focused on the visualization of students' study progression and their improvement of computational thinking through the dashboard on SAGE platform to assist both students and instructors. With the dashboard, students can compare their learning progression with other students who are working on the same project or solving the same problem on the gameful learning platform SAGE (Bender, 2015). It also provides useful information of student programming activities that assist instructors by providing real-time help to the students. Instructors can conveniently track each student's progression and keep the student inside the Zone of Proximal Flow. The Zones of Proximal Flow (Basawapatna, 2013) is a concept combining Vygostsky's Zone of Proximal Development and Csikszentmihalyi's state of Flow. The concept of flow is focused on how one can have optimal experience by performing an activity at one's limit. The Zone of Proximal Development is concentrated on what one can achieve with external help. The Zones of Proximal Flow aims at the point to keep the learner continually have optimal experience while pushing one's limit with accessible external help (Murphy, C., Kaiser, G., Loveland, K., & Hasan, S. 2009). In this way, students can have the optimal experience with deep concentration, feeling of enjoyment, and the incentive to explore more.

In order to help students reaching such preferable state of flow, instructors need a more automatic and direct visual presentation that reflect the changes in students' learning process. It should be able to detect when a student is stuck at a problem for a relative long period of time and inform the instructor to give real-time help to the student. The dashboard should also assist the instructor to adjust the problems' degrees of difficulty as student progress.

Through a project-first approach, students are more motivated compare to the traditional theory first approach and are able to gain knowledge inside Zones of Proximal Flow. It creates a more nature way for each student push their own limit. This project-first approach also increases the difficulty for instructors to evaluate each student's improvement. To control the assignment's difficult and individualize each assignment, instructors are given the role of game designers who create games with Scratch for students to play while improving their computational thinking skills. Scratch is a programming language with an online platform that enable everyone to create and share their own projects. Scratch opens the opportunity for instructors with no previous programming experience to create games. SAGE extends Scratch by incorporating tooling that collects and analyses data to visualize students learning progression. With these additional tools, instructors gain more control over the whole project and at the same time, students still have the freedom to collaborate and compete through the open online platform of Scratch.

2.0 Related Works

Animation and game-based programming has been a key movement for educating young people with elementary programming skills and introducing computational thinking skills. Although there exist many related animation programming platforms, there are still a lot of space in the area, abstracting quantitative information from student actions while their doing their assessment and quantifying students' progression, for improvement. There is not a definite way to measure a student's computational thinking skills, but there are some researches focused on different perspective of the computational skills and some ways to quantify and visualize them. The quality of the instructors' guidance also hugely influences their students' learning experience. Having some kind of metrics to indicate if a student is in trouble, below the preferable zones of proximal flow, can help instructor give special attention to the student and give timely guidance. The section below discussed some related research conducted in the area of improving student learning experience by all kinds of assisting tools, quantifying students' action for further analyze, and improving instructors' teaching methods.

2.1 Scratch

Scratch (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010) is a visual programming environment that is initially designed for children from age 8 to 16 at an after school computer science program to motivate them to learn programming. Scratch is used to create project with image and sound, media and script. These projects are story telling games, sensor driven art, music project and so on that successful attracts young people's attention. The figure below shows some of these projects.

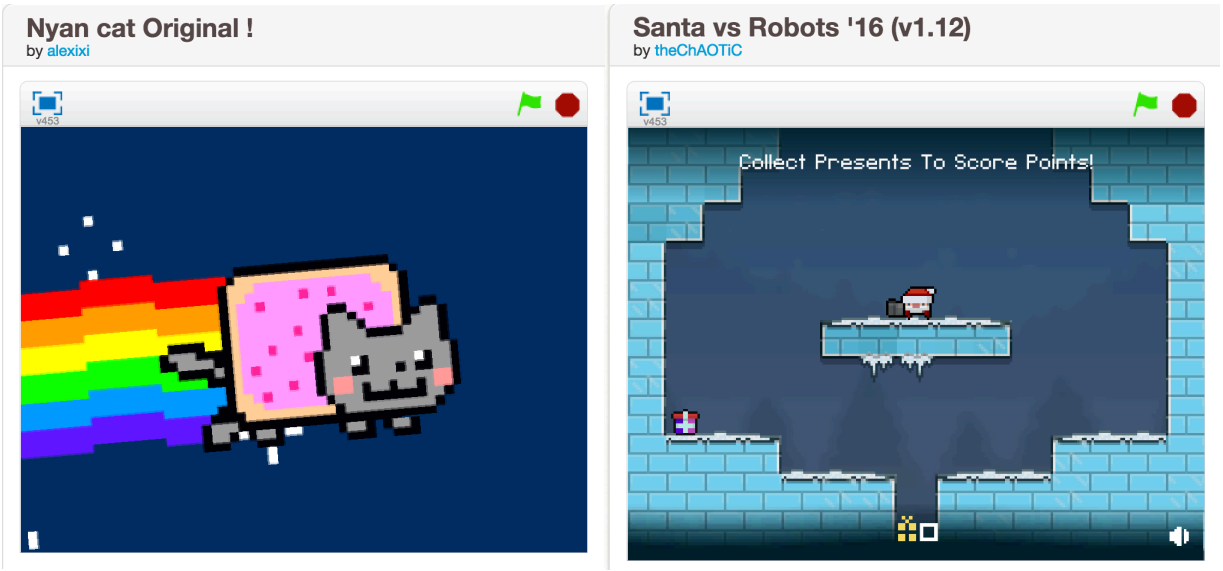

Figure 2. Example Scratch Projects

Scratch added media-manipulation activities that are popular in young people's culture and courage young people to learn hands on experience and improve by pair sharing. It is really different from the traditional text based programming language and one guided direction learning setting in classroom. Scratch editor has been chosen as the core language that will be used by the students in SAGE system.

2.2 Retina

Retina (Murphy, Kaiser, Loveland, & Hasan, 2009) is a tool that collects students' programming activities and processes the collected data to provide helpful information for both students and instructors. It tries to solve the problems such as how much time students spend on each assignment, what errors students made during the assignment, how each students stand in the class compare to other students. Retina collects students' compilation time, compilation error, runtime errors, stack trace and so on. On the instructor side, Retina displays a list of information about each students and aggregate data such as total number of compilation error, most common compilation error, average times spent on the assignment, etc. Figure 3 gives an instructor example view of Retina.

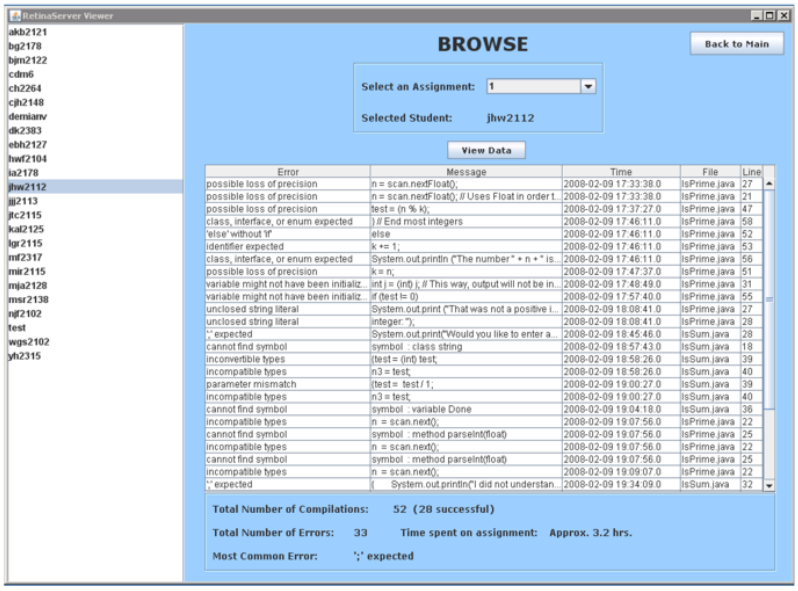

Figure 3. Retina Instructor View in Browse Mode

On the student side, Retina shows the number of compilation time and type of errors. It also gives suggestion on expected time consumption on upcoming assignment and give suggestion

about the potential errors the students might make. Figure 4 gives a student example view of Retina.

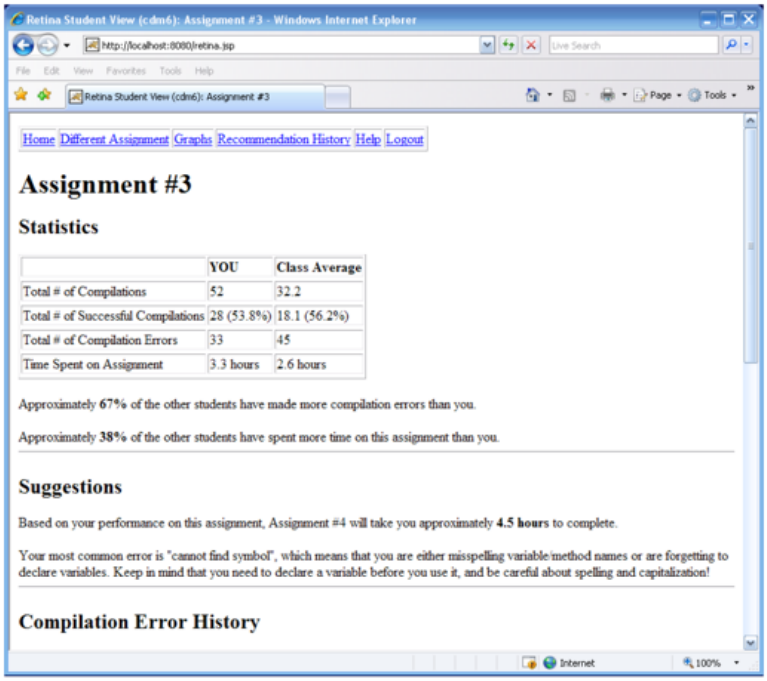

Figure 4. Retina Student View

Students are able to learn from their previous errors with those aggregate data and suggestion given for the upcoming assignment. Retina also displays how many students are currently working on the assignment and creates a committee feels to encourage students that they are not working alone. It is essential to keep students motivated and curious as they learn. We are adopting the similar concept that shows the attendance of the student. If the students logged into the system and tried to complete one of his or her assignment 7 days in a roll, he or she will get a badge to reward his or her effort. To motivate them and indicate that they are making progress, the dashboard will display a progression bar that shows how many subtasks are completed out of the total number of subtasks. By improving the interaction between instructors and students, Retina helps improve instructor's lecture and students' experience with the course. To improve the interaction between instructors and students, the dashboard in SAGE provides real-time information as students working on the assignment and instructors can respond accordingly as students get stuck on the assignment.

2.3 GradeCraft

GradeCraft (Holman, C., Fishman, B., Aguilar, S. 2013) is a custom learning management system that supports game-inspired courses and foreground the affordance of gameful learning environment. The developers of GradeCraft took an list of heavily researched techniques when designing their gameful learning platform. By deploying many simply gameful techniques, the developers established a solid foundation from which to build more nuanced gameful functionality in the future.

The GradeCraft platform includes two dashboard views, the Student Dashboard and Instructor Dashboard. The student dashboard is a single-page displaying of student's comprehensive course progress. Figure 5 shows an example of Student Dashboard.

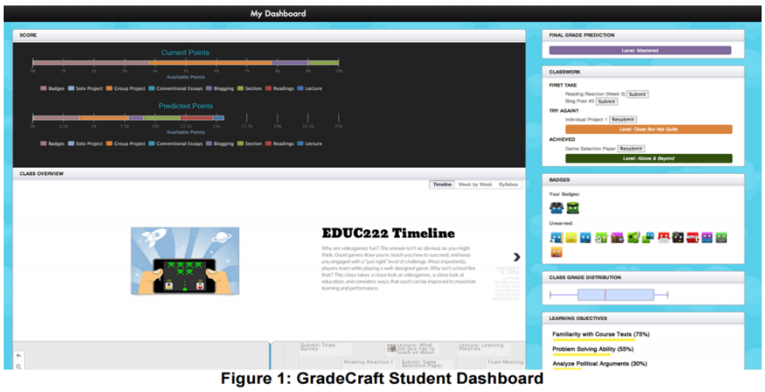

Figure 1: GradeCraft Student Dashboard

Figure 5 GradeCraft Student Dashboard Overview

The top progress bar serves an informational purpose and also have a motivational effect that help boosts user motivation to complete tasks. The top portion includes visual chart of the point and badges they have earned paired with ones that are still available to work on. Figure 6 shows an example of Student Dashboard badges.

Figure 6 GradeCraft Student Dashboard Badges

It also shows a To-Do list that highlights upcoming assignments, assignments that could be redone to show improvement or skill mastery, and, if possible, feedback on a recent successful assignment. It provides a convenient view for student to keep track of their upcoming assignments and course progress. Figure 7 shows an example of Student Dashboard To-Do list.

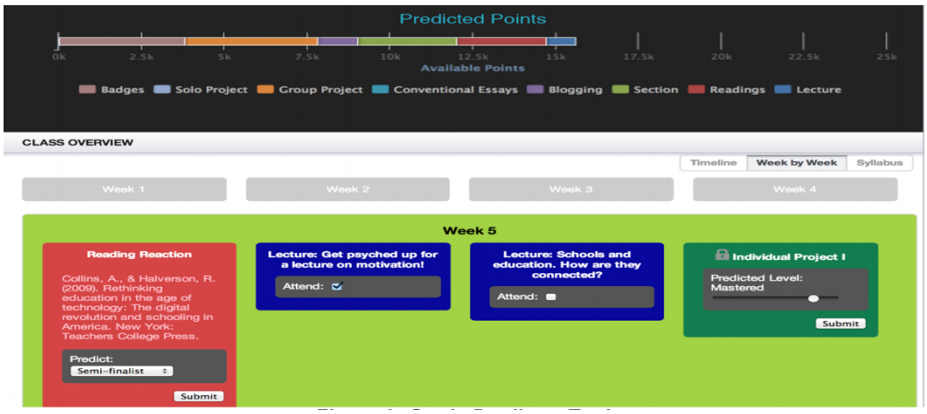

Figure 7 GradeCraft Student Dashboard Badges TO DO List

Tabs shows their progress towards achieving the course learning objectives. Class grade distribution is charted to show the students their overall performance against the class. Figure 8 shows an example of Student Dashboard Learning Objectives Tab and Class Grade Distribution Tab.

Figure 8 GradeCraft Student Dashboard Objectives Tab and Class Grade Distribution Tab.

Beneath the tabs is a display of the semester plan that students can manipulate, selecting between a calendar view, a list view, a timeline, and a tech-tree display of the semester dates and assignments. These displays also operate as the portal through which students submit their work, identify self-selected groups, record participation, predict their scores, and receive feedback. Figure 9 shows an example of Student Dashboard Calendar.

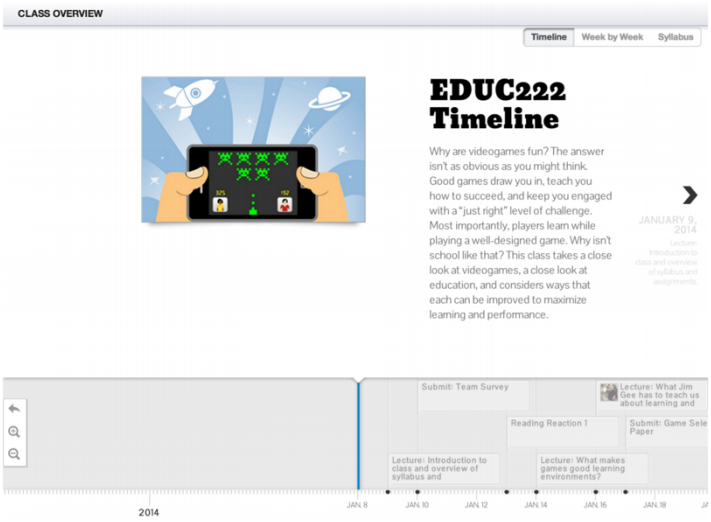

Figure 9 GradeCraft Student Dashboard Calendar

The instructor dashboard shows teachers how their class is performing in a single view. The ten lowest and highest performing students' grades are each visualized with stacked bar charts, each color segment reflecting achievements within an assignment. Instructors can rapidly isolate which students may be in need of more help. A box and whisker plot is used to capture the overall class performance, displaying the range of achievement as well as situating how the majority of students are doing. Figure 10 shows an example of GradeCraft Instructor Dashboard.

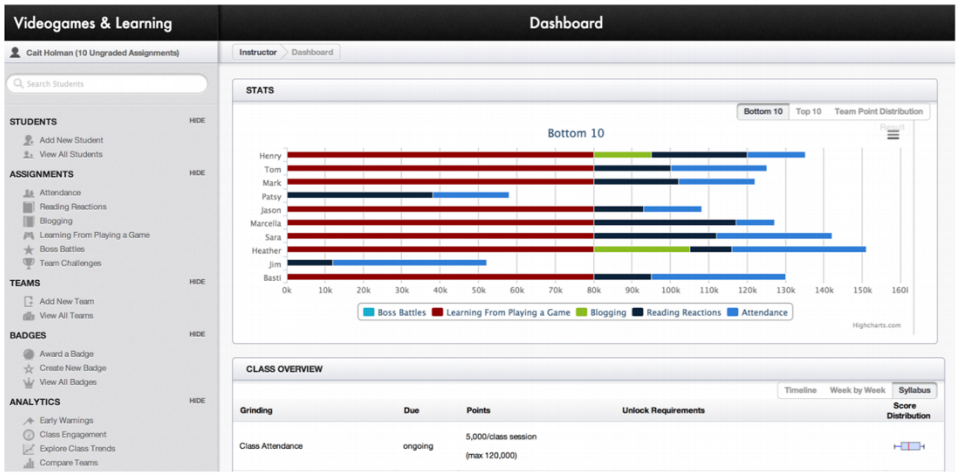

Figure 10 GradeCraft Instructor Dashboard

There is a list of techniques that can be immigrant from GradeCraft to the proposed project includes: using points and incremental levels instead of grades; awarding badges to recognize achievements and skill-acquisition; allowing students to redo assignments as many times as necessary to succeed; allowing students to determine how much assignments would count

towards their final grade, selective view of top and bottom performancer in class, and displaying generalized information regarding classmates' performance.

2.4 REACT

REACT (Koh, K., Basawapatna, A., Nickerson H., & Repenning A. 2014) is a real time graphic assessment tool that quickly gives teachers insight into students' mastery of computational thinking construct as they creating games and simulations. It's an embedded assessment and web-based system that does not focus on end-user's programming tools providing real-time system alter for teacher on the student's mistakes. An example of this feedback is illustrated in Figure 11.

Figure 11. REAL Assessment Feedback

Furthermore, it explicitly displays computational thinking pattern based assessment instead of lower-level programming construct. An example of this feedback is illustrated in Figure 12. It allows teacher to see which high-level computational thinking concepts student mastered and which ones they are struggling with on class and individual level, eventually allow them to make more effective instructional decisions.

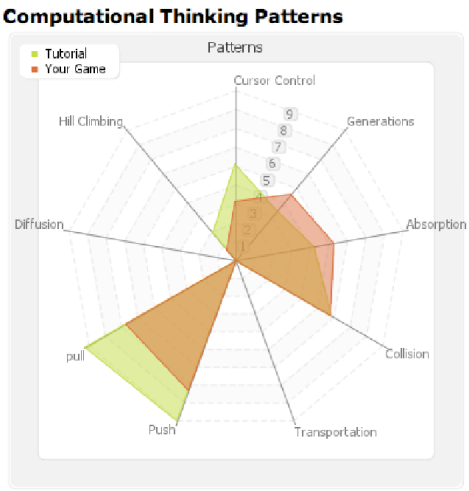

Figure 12 REAL Assessment Computational Thinking Pattern Feedback

In the proposed project, we can adopt a feedback system similar to the one developed by the REAL assessment team. For example, REAL generates personalized individual student's feedback based on computational thinking pattern. The proposed project can also leverage its web-based natural and generate automated evaluation results in real time for teachers to view.

2.5 The Dashboard of Open Learning Analytics

The dashboard is implemented to assist individuals on making decisions about teaching and learning (Siemens, G., Gasevic, D., Haythornthwaite, C., Dawson, S., Shum, S.B., Ferguson, R., Duval, E., Verbert, K., Baker, R.S.J.d. (2011)). It contains progression bars that indicates instructors' or students' self defined learning goals of the assignment. The decomposition of the assignment are lists for students control their own pace of learning. There are also categories for students to evaluate of the course and assignment and also serves as feedbacks for instructor to improve the course or assignment. Figure 13 provides a demo of the learner dashboard.

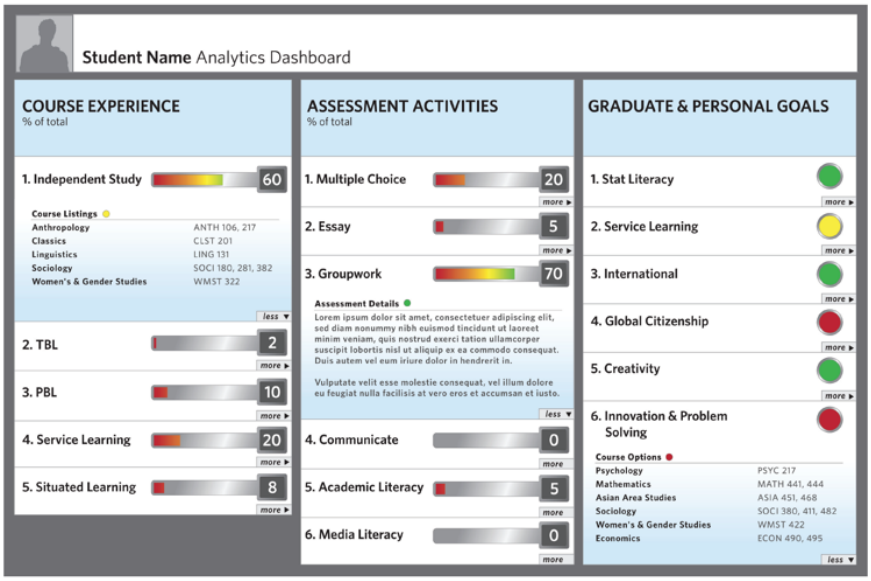

Figure 13 Learner DashBoard of Open LA

The instructor dashboard is similar to the learner dashboard that displays the course experience of each students and the goals that instructors aimed for each assignment. Based on the established researches, students' completion and satisfaction of a course can be impacted by students' attendances, connection of the rest learning committees, early assistance from instruction and etc. Figure 14 shows the view of instructor dashboard.

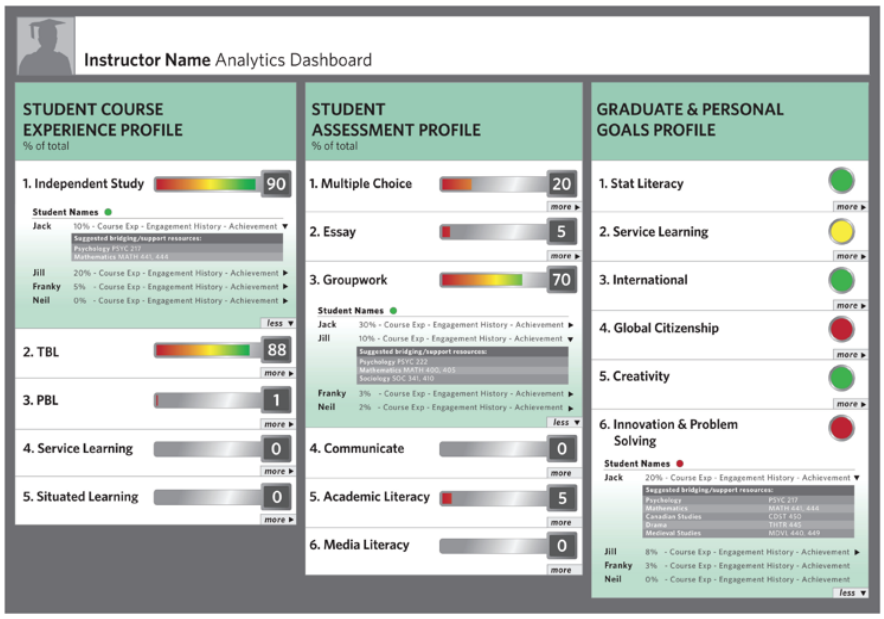

Figure 14 Instructor Dashboard of Open LA

With the dashboard, instructors' can identify which student need help at an early state. The students will get warning when attendance pattern changes or attendance drop substantially. These are some basic and important indicators that can also be adopted in SAGE's dashboard for instructors to provide assistance in time and for students to control their own learning pace.

2.6 Hairball

As more and more animation or game-like projects and languages introduced for teaching younger students how to program, the problem of how to evaluate this project surfaced. Unlike traditional text-based language, Scratch program does not have a text-based output file and can be easily test by conversional unit tests. Hairball (Boe, Hill, Len, Dreschler, Conrad, & Franklin, 2013) is an automated static analysis tool that can be used both by the students and the graders or instructors. It provides a "lint-like" analysis tool for Scratch that can be used as an evaluation for Scratch projects. Hairball can identify whether a Scratch program has certain construct or not, whether the implementation of a Scratch program is robust or not.

Hairball provided a platform easier to implement plugins for evaluating and analyzing Scratch programs. The plugin is required to ben written in Python. Even though Hairball is able to automate some part of the process of evaluating the Scratch project, the cohesion of a visual effect and the overall aesthetic effect of the program can only be evaluated manually.

2.7 Mastery

Mastery (Moreno-León & Robles, 2015) is a plug-in develop for the Hairball environment that analyzes a Scratch project by the seven categories: abstraction and problem decomposition, parallelism, logical thinking, synchronization, algorithmic notions of flow control, user interactivity and data representation. The basic set of rules used by Mastery to evaluate the Scratch project by computational thinking concepts are shown in Figure 14.

CT Concept	Basic	Developing	Proficiency
Abstraction and problem decomposition	More than one script and more than one sprite	Definition of blocks	Use of clones
Parallelism	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is > %s, two scripts on when backdrop change to
Logical thinking	If	If else	Logic operations
Synchronization	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Flow control	Sequence of blocks	Repeat, forever	Repeat until
User Interactivity	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is >%s, video, audio
Data representation	Modifiers of sprites properties	Operations on variables	Operations on lists

Figure 15. Evaluation criteria of Mastery

We also adopt and used this seven component concept in the dashboard. The Mastery plugin is used to generate a computational score for the submitted student assignment.

3.0 Features

The features of SAGE Dashboard can be divided into three parts, the login and register, the student dashboard, and the the instructor dashboard. The register page allows user to register for a student account or an instructor account. For login, the user needs to provide the correct email and password combinatory. After successful login, user will be redirected to the student dashboard or the instructor dashboard. The student dashboard displays the information about what courses the student is taking, his / her assignments' scores, completeness of the current homework he / she is working, and etc. The instructor dashboard displays information about what courses their teachings and the general performance of the student in the class.

3.1 Log in and Register

The only exist login functionally ask for the student username and their assignment id to redirect them to the assignment they want to work on. We implemented the login and register functionality for both Dashboard since there are no actual account login functionality in any other part of SAGE. This will be main gate way for user to get into the SAGE system and use any of the functional SAGE provides.

The register page allows the user to enter their email, password and their username. There is also a slide bar that allows the user to specify what kind of account they want to register, student or instructor.

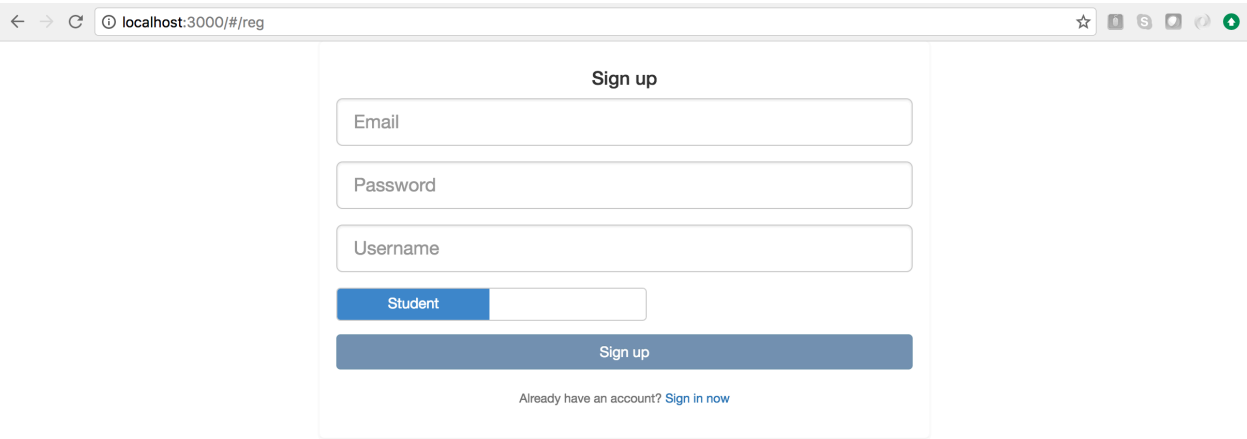

Figure 16. Register Page

The login page allows the user to enter their email and password and redirect them to the corresponding student or instructor dashboard if they entered the correct combination.

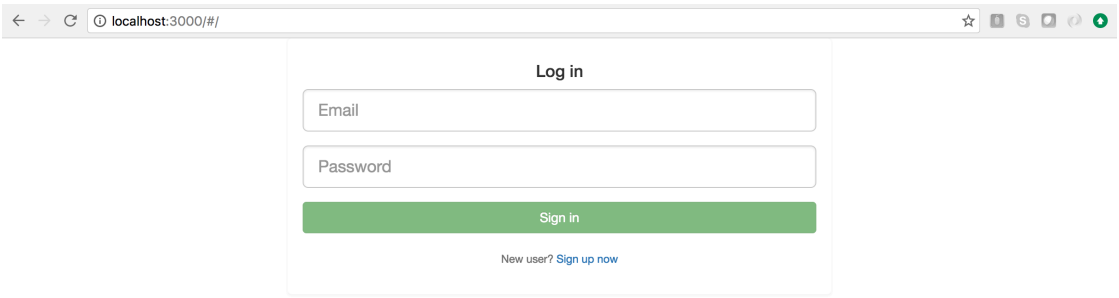

Figure17. Login Page

3.2 Student Dashboard

Student dashboard displays students' information about what courses they have registered, what homework do they have for each course, what are their score for each homework, what is their evaluation corresponding to seven categories and so on. The student's avatar and student's name are shown on the left side. There are also four tabs on the left side, Account, Overview, Tasks, and Hide/Show.

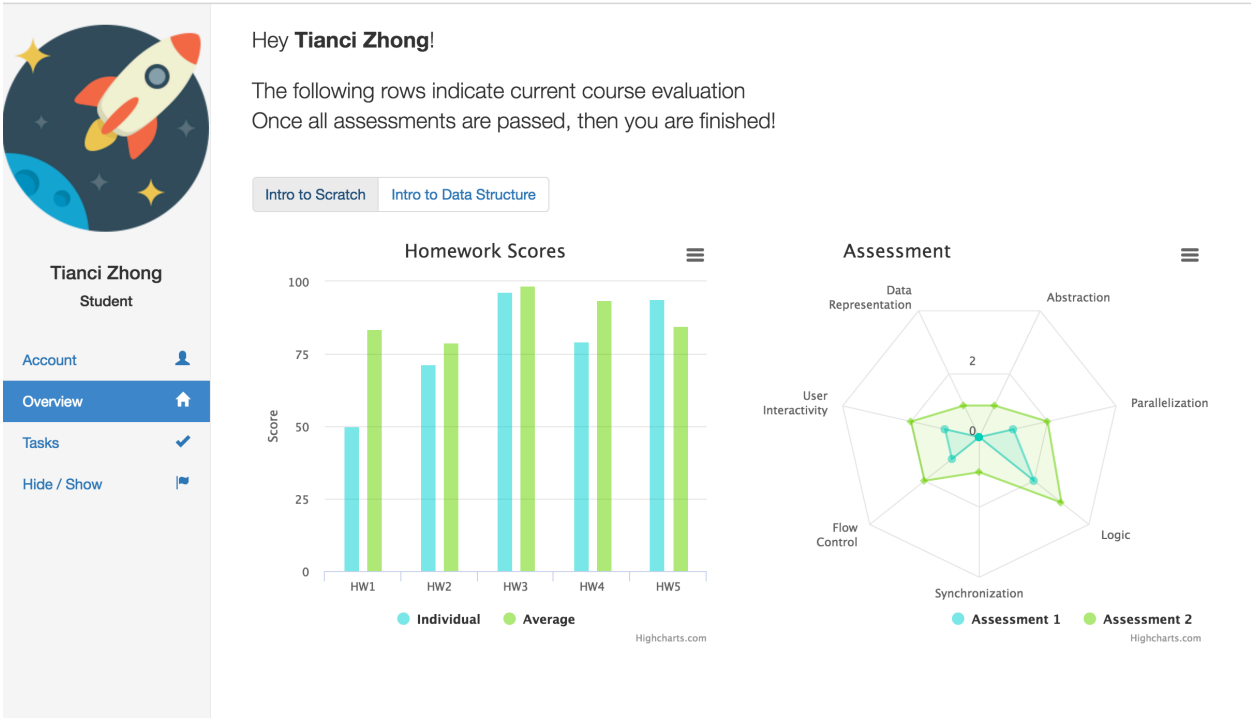

Figure 18. Student Dashboard Overview Page

The figure above shows the contains when the student clicks the Overview tab. The tabs near the top of the page shows the current courses this student has enrolled. The graph on the left shows the homework grades of this student for the course heighted in grey. It displays the students score in blue and the average of the students enrolled in the course in green, so the student can easily get a sense of their standing in the class. The graph on the right displays the assessment for each submission of the homework. Each submit is evaluated by these seven categories, Data Representation, User interactivity, Flow Control, Synchronization, Logic, Parallelization, and Abstraction. The heptagon provides a really visual comparison of each submit of the homework and how the student's progress in different area in details.

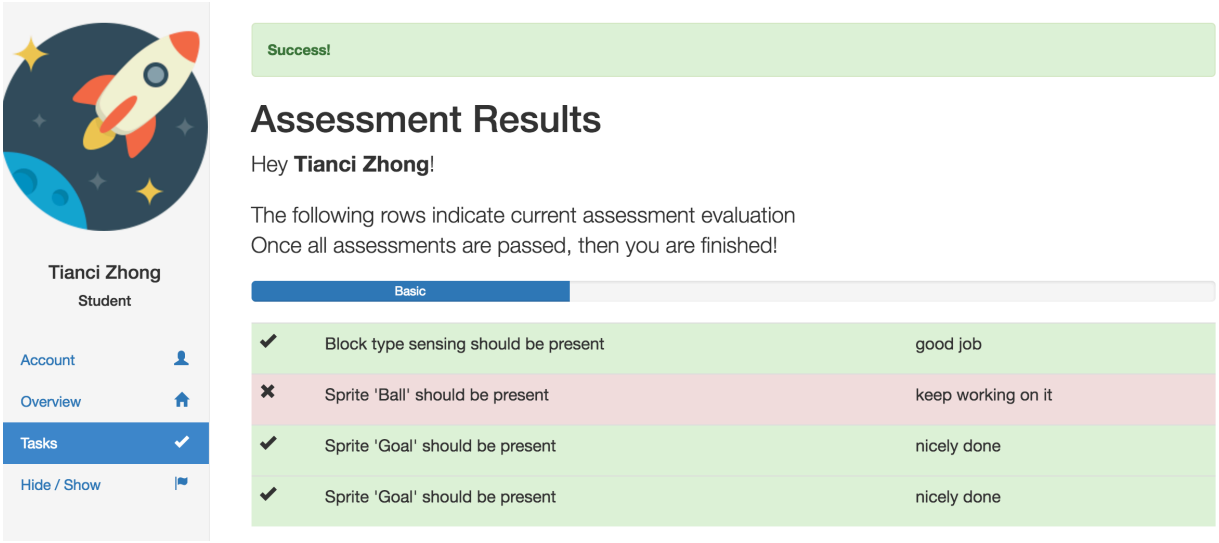

Figure 19. Student Dashboard Tasks Page

The figure above is the view inside the tasks tab. It shows the students the progression of the homework he / she is currently working on. There is a list of tasks that will be marked as complete or still in progress. It helps the student to keep track of what they have done and what he / she still need to implement for the homework. This helps the student breakdown each homework inside smaller piece instead of a huge block that makes them clueless.

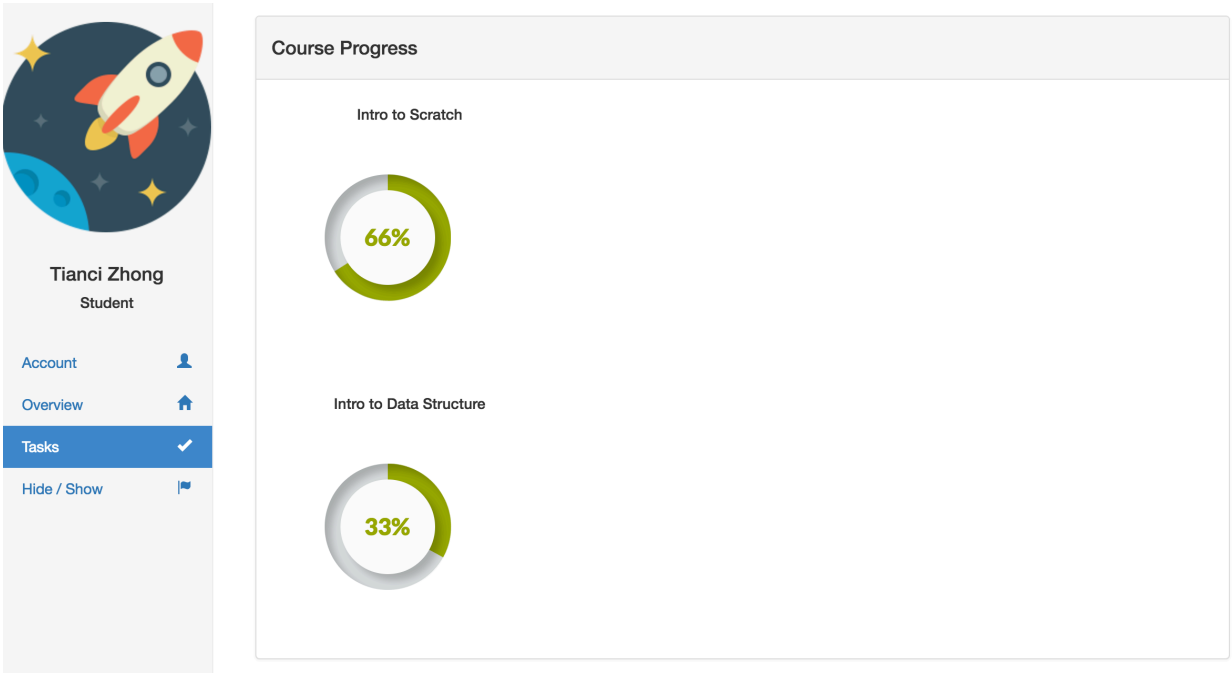

Figure 20. Student Progression bars

The figure above is also a view inside the tasks. It displays the overall completion of the course according to how many homework or tasks they have completed in total for each courses. It gives the student a sense of accomplishment as the percentage goes higher.

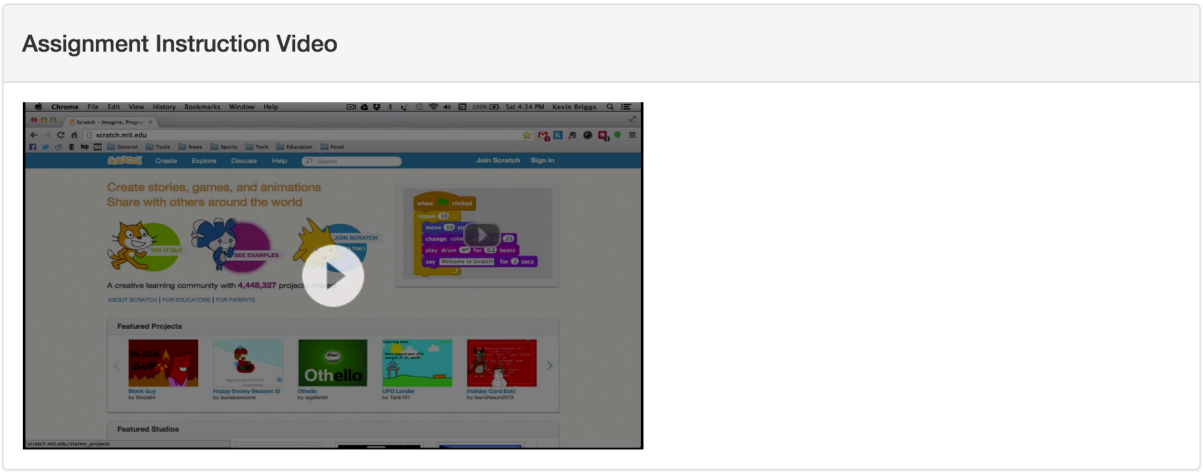

Figure 21. Video Guidance

Here is another view inside the task tab. For each homework, the instructor will upload assisted video for student, so students can view the video at this page by selecting the corresponding course and homework.

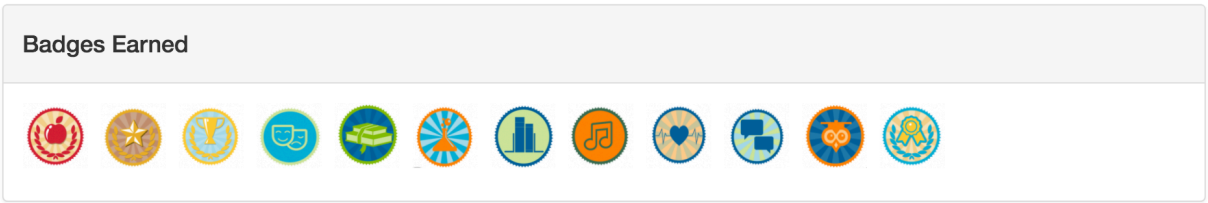

Figure 22. Badges System

The figure above is the display of badge system at the bottom of view inside the tasks tab. The badge system designed similar reward system inside many popular games. It motivates the students to get more badges by submitting cleaner code, better performance and more efficient code, and so on. It can also encourage students to login to the system more often by account the number of days they have login the system and worked on their assignments. It creates a positive competitive environment inside SAGE learning system. In the badge system, students are not really competing with other students. They are actually competing with themselves. Since each students' starting point is different and each might have different learning curves, students at the bottom of the class will not be discourage by other students.

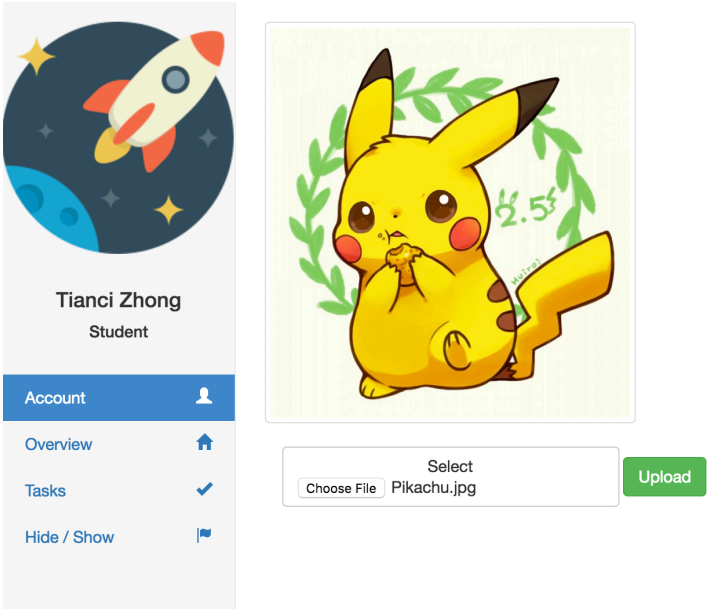

Figure 23. Avatar Upload

The figure above shows the Account setting page. Students can update their own avatar to personalize their account. This functionality helps students get a sense of ownership. The sense of ownership is essential to keep students coming back to the system.

The hide / show tab will hide or show the display on the right hand side. It is design to show the Scratch editor view after the Scratch editor embedded into this dashboard view.

Overall, the student dashboard is designed to be colorful and play since the main user will be primary or middle students. The bright and vibrate colors attract young people's attention and tricks them to think SAGE as an interesting game instead of a boring learning system for coding and improving computational thinking.

3.3 Instructor Dashboard

The instructor dashboard includes the Overview and Upload Video views. It is designed to assist instructor to get a general idea of how the students are doing in the class. It guides the instructor to adjust their teaching speed and the homework assignments' difficulty.

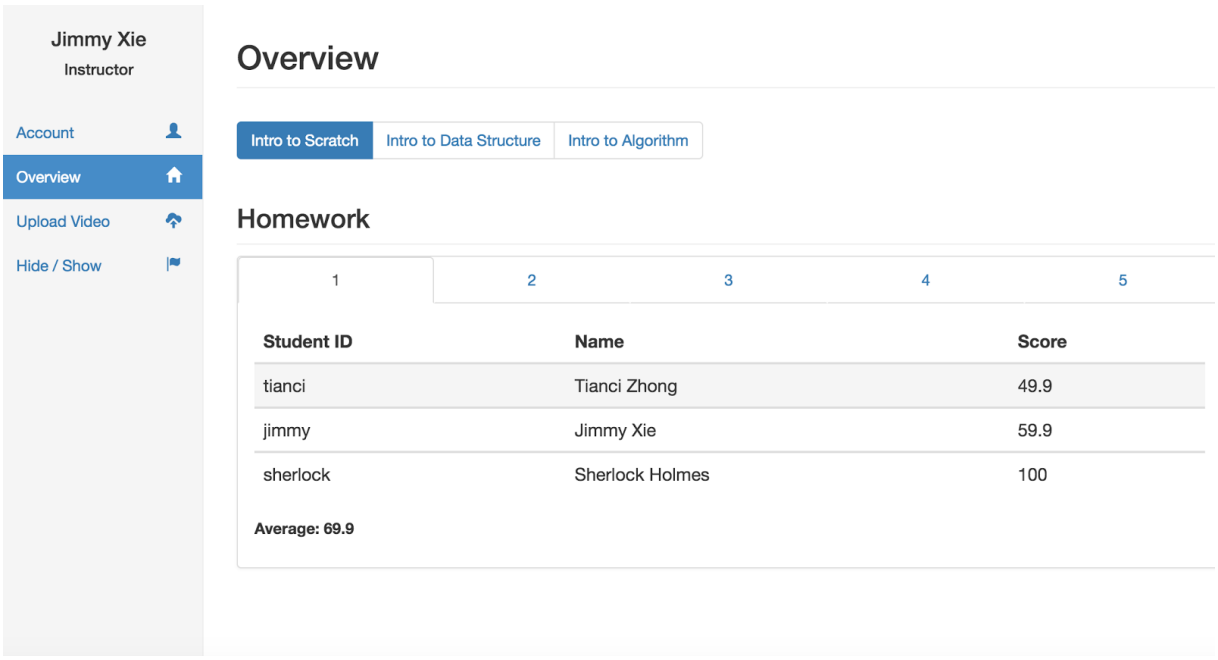

Figure 24. Instructor Dashboard Overview Page

The figure shows Overview view when logged in as instructor Jimmy Xie. It displays the course that this instructor is currently teaching as tabs at the top. User can click on the tab and the tab will be high lightened in blue. Underneath the course tabs, it displays the students' homework submission and scores. This informs the instructors how well the student does on each of their assignment.

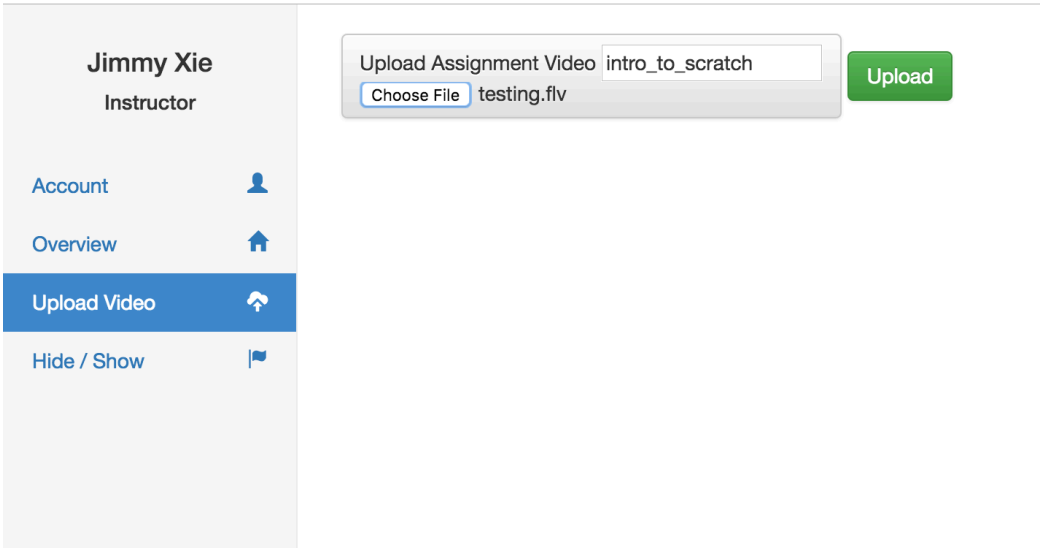

Figure 25. Instructor Video Upload Page

This figure above shows the functionality for instructor to upload video to cloud, so the students can access these video through their dashboards.

4.0 Architecture

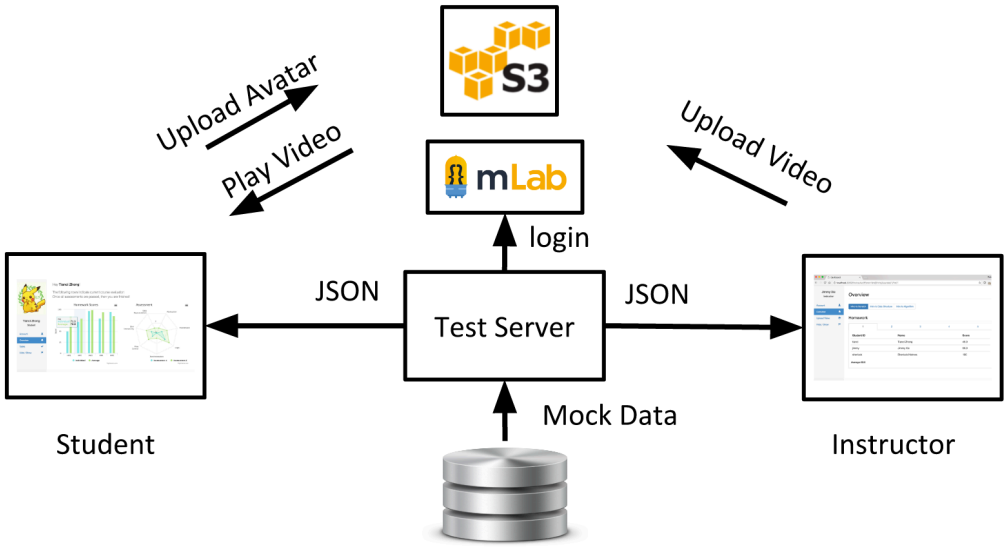

Figure 26. System Architecture

The figure above shows the architecture of SAGE dashboard. We wrote the test server in node.js. We used a local file directory to store all the mock up data. The mock data is stored in JSON format. The mLab provides online service to host MongoDB database on cloud, so it's easier to manage and set up when you run the server on local machines. We used the Amazon S3 storage service to store all the avatars and videos uploaded by the students and instructors. The test server will serve the mock data on local directory to the student and instructor dashboards as request through Restful APIs.

5.0 Implementation

5.1 Test server

The test server is implemented with node.js and express. Node.js is an open-source JavaScript runtime environment that is interprets by Google's V8 JavaScript engine. Node.js has asynchronous I/O which can optimize web application with many input / output operations. Node.js also has one of the largest open source library contributors. Most of these open source libraries are hosted on NPM which can be easily install with the ` npm install` commend. We choose Node.js as our test server because it has been widely used in many industries for front-end server.

The HTTP endpoints that servers the mock data are documented below:

Description	Get the student information with the given student id
Route	Get /stats/students/{id}
Response Code	200, on Success 403, on Unauthorized
Response Body on Success	{ "name": … , "id": … , "number_of_courses": …, "courses": [{ "name": … , "id": … , "number_of_hw": … , "individual": […], "average": […], "assessments": [{ "number_of_assessments": … , "data": [{ "name": "Assessment … ", "data": […] } …] } …] } …], "progress": 0.95, "badges": […], "tutorialID": …

Response Body
On others
	{
	status: "failed",
	message: "Not authorized."
	}

Description	Get the instructor information by the given instructor id
Route	Get /stats/instructors/{id}
Response Code	200, on Success 403, on Unauthorized

Response on Success	{
	"name": … ,
	"id": … ,
	"courses": [
	{
	"name": …,
	"description": …,
	"id": … ,
	"hw": […],
	"hw_averages": […],
	"students": […]
	}, …
	]
	}

Response on Others	{
	status: "failed",
	message: "Not authorized."
	}

Description	Get the instructor information by the given instructor id
Route	Get /stats/instructors/{id}/courses/{cid}/hw/{hid}
Response Code	200, on Success 403, on Unauthorized

Response on Success	{
	"data": [
	{
	"name": … ,
	"id": … ,
	"score": …
	}, …
	],
	"average": …
	}

Response on Others	{
	status: "failed",
	message: "Not authorized."
	}

5.2 Log in and Register

The login and register is implemented with the library “bcryptjs”. It will create a token for the user and when user try to assess data we will check if this token is authorized to access the information. The user information from register is stored at mLab with a MongoDB database on the cloud. The current account is our personal account. The mLab account is free. We believe a new account should be created and managed specifically for the SAGE development group. After the new database and collect to storage the user information created, the link in the code base that direct to the current database should be changed, which is inside the server.js file. The line 17 below should be modified and direct to the new database created.

```
13  var mongoose = require('mongoose');
14  // var useragent = require('express-useragent');
15
16  mongoose.Promise = global.Promise;
17  mongoose.connect('mongodb://user:user@ds133328.mlab.com:33328/sage-login');
```

The HTTP endpoints for login and register are documented below:

Description	Check the user’s email and password
Route	POST /auth/login

Request Body	{
	"email": " … ",
	"password": "… "
	}

Response Code	200, on Success 400, on Bad Request

Response on Success	{
	"token": "… ",
	"user": {
	"_id": "…",
	"email": "…",
	"fullname": "…",
	"role": "…",
	"__v": 0
	}
	}

Response on 400	{
	"message": {
	"password": "The password is not correct."
	}
	}

Response on Others	{
	"message": {
	"email": "This user does not exist."
	}
	}

Description	Register the user's email, password, username and role as student or instructor
Route	POST /auth/reg

Request Body	{
	"email": " … ",
	"password": " … ",
	"fullname": " … ",
	"role": " … "
	}

Response Code	200 on Success 409 when email has already registered

Response on Success	{
	"token": " … "
	}

Response on 409	{
	"message": {
	"email": "Email is already taken."
	}
	}

Response on Others	{
	"message": {
	"email": "This user does not exist."
	}
	}

The login page can be accessed by the path:
http://localhost:3000/#/

The login page will prompt the user is the email address they entered does not exit, it will prompt the user that this user account does not exit. It will prompt the user invalid password, if it sends the information to the database and the email and password combination does not match the record stored in the database. The password is encoded by using the library 'jwt'.

The register page can be accessed by the path:
http://localhost:3000/#/reg

The register page will prompt the user if the email they entered already exist in the database. The user can click on the slide to choose which kind of account they want to create. The figure below shows the database and collection we created in mLab.

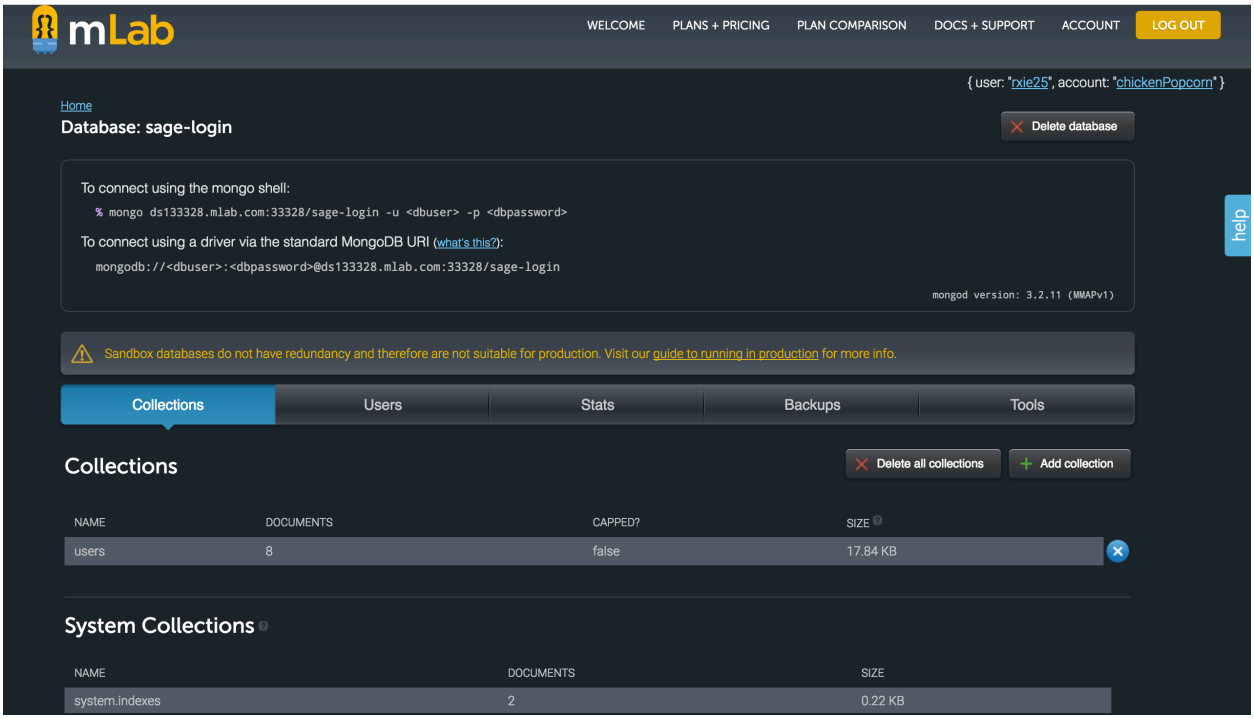

Figure 27. Database and Collections

5.3 Upload Avatar and Videos

The videos and avatars are upload to the Amazon S3 storage which allows easy access to the data on the cloud and it automatically handles database failures. It provides fast access, high availability and high secure for the data we store. For using the AWS services, an AWS service

account is needed. The accessKeyId and secretAccessKey is under the config folder inside the file named ‘aws-config.json’, which is not publicly distributed on our github repo. The AWS S3 service is free when the stored data is not large and the request mount is relative low. The accessKeyId and secretAccessKey currently used is our personally account. We believed a specific AWS account create for the group that develops the dashboard or other part of SAGE will be help and easier to maintain and run the code. The AWS account is free to create and contains a year of free services.

The HTTP endpoints for upload avatar and video to the S3 bucket are documented below:

Description	Upload student avatar to the S3 bucket on AWS
Route	POST /upload
Response Code	200, on Success

Response on Success	{
	"error_code": 0,
	"err_desc": null
	}

Response on Others	{
	"error_code": 1,
	"err_desc": " … "
	}

Description	Upload video to the S3 bucket on AWS
Route	POST /uploadVideo
Response Code	200, on Success

Response on Success	{
	"error_code": 0,
	"err_desc": null
	}

Response on Others	{
	"error_code": 1,
	"err_desc": " … "
	}

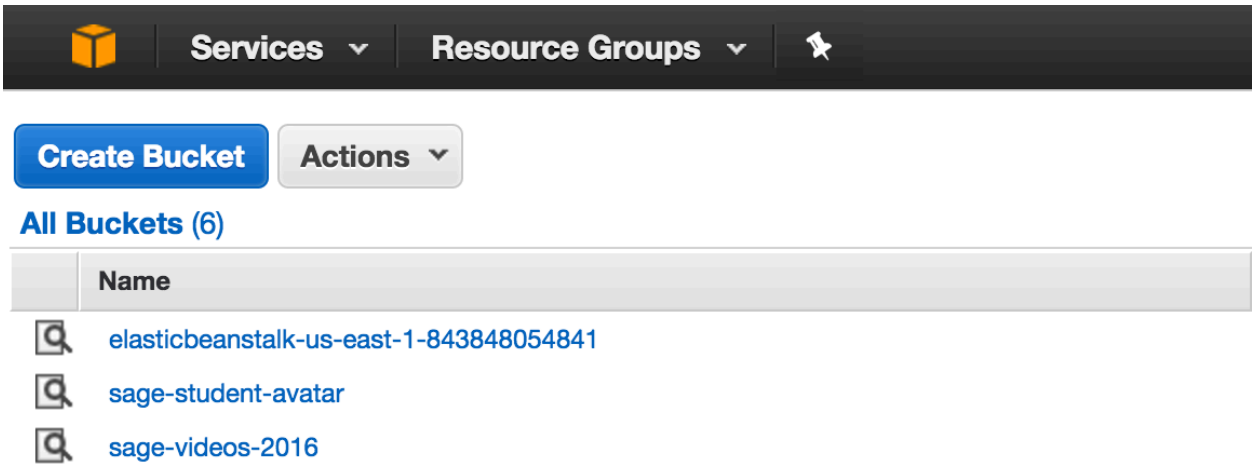

Figure 28. Amazon S3 Cloud Storage

We created two bucket in Amazon S3 service, sage-student-avatar and sage-videos-2016. The student avatar and guidance video will be storage separately in this two bucket. Inside the student avatar bucket, each avatar is name by the username plus "-student", e.g. if the student's username is "tianci", then the stored avatar will be "tianci-student.jpg". The video storage by the instructor will be name as the assignment name the instructor entered before they upload the video.

5.4 Student Dashboard

The student dashboard is implemented with HTML, CSS, Angularjs and Bootstrap. We also imported library "animate.css" for the hide / show animation. Student dashboard uses the single page page website design. The student dashboard itself is a standalone single page website. We use the angular router for routing the path inside the student dashboard.

The student overview page can be access by the path:
http://localhost:3000/student/#/overview/{sid}

The two graphs displayed on the Student Overview page are using HighChart. The drawn graphs can be saved as images into database if needed.

The student tasks page can be access by the path:
http://localhost:3000/student/#/tasks/{sid}

The tasks page's round progression bar is implemented using HighChart. The embedded video uses the "vjs-video" library to display the .flv video storage on cloud in S3.

The student account page can be accessed by the path:
http://localhost:3000/student/#/account/{sid}

The student account page allows student change their avatar and the uploaded image will be save on S3 storage in the bucket sage-student-avatar.

5.5 Instructor Dashboard

The instructor dashboard is implemented with HTML, CSS, Angularjs and Bootstrap. We also imported library “animate.css” for the hide / show animation. The instructor dashboard also uses the single page page website design. The instructor dashboard itself is a standalone single page website. We use the angular router for routing the path inside the instructor dashboard.

The instructor overview page can be access by the path:
http://localhost:3000/instructor/#/overview/{id}

The instructor dashboard displays the table of information for each course and student’s score for each assignment. It is styled by Bootstrap and the data is served with Angular controllers that make HTTP request to the test server.

The instructor upload video page can be access by the path:
http://localhost:3000/instructor/#/uploadVideo/{id}

The instructor upload video page allows instructors to upload guidance video for students and it will be save on S3 storage in the bucket sage-videos-2016.

6.0 Development

6.1 Code Repository

Github is used as the code repository for the Dashboard. The link to the Dashboard code repository is below:

https://github.com/cu-sage/sage-frontend

6.2 System Requirement and Development

To run the program, you need to install node and npm first. If you are using a Mac OS system, they can easily install using the commend below:
$ brew install node
$ brew install npm

To install all the library and requirements specific for this repo, you can run the commend:
$ npm install
This commend will find and install all the requirement noted in the package.json file under the folder node_modules.

Then, you need to enter the AWS accessKeyId and secretAccessKey in the file aws-config.json inside the config folder.

7.0 Future Work

The SAGE editor is a flash object. Initially, we are thinking embedded the student dashboard and the instructor dashboard to SAGE editor and we find it's relative hard to do so. We also figure out that there is not an actual account login system in SAGE editor yet. We believe the best practice will be embedded the flash object into the current dashboards. We implemented the login and register system to match the username with the userid that is needed in the SAGE editor to login a current assignment. We have set the road to embedded the SAGE editor into our dashboard, but we do not have enough time to finish embed the SAGE editor into the dashboard. We would like to finish this part of the work and bring the whole SAGE system together.

The instructor dashboard should have links to the student dashboard page for each student who is enrolled in he / her classes, so the instructors can also view the detailed information about each student's assignment submission. It is useful for the instructor to think about designing assignment according to the computational thinking categories.

For the student dashboard, we would like to populate the badge system with a specify set of rules. Currently, it just displays the data served by the test server. We would like to implement the logic behind it with all the evaluation result we get for each student. Many of the data used by the test server does not collect or calculated by the backend server yet.

Finally, we would like to during the frontend server and backend server together and collect more data from the SAGE editor for computing useful information about each student. Since both frontend and backend server is written in Node.js, it might be ideal to bring both servers into one single repo to make it easier to maintain.

8.0 Conclusion

The dashboard implemented for this project is focused on the visualize of student's learning progression and keep student continuously inside their zones of proximal flow. For animation orientated programs, it is hard to evaluate using the traditional text-based program evaluating methods such as unit testing and output text comparison. Instructors need to specific attention and spend much more time on grading and evaluating each student's assignment. We used the Hairball and Mastery plugin to automate part of the evaluation of the assignment. Computational thinking is a really abstract concept and it's hard to quantity and been visualized. This now divided into seven more concrete subcategories: Data Representation, User interactivity, Flow Control, Synchronization, Logic, Parallelization, and Abstraction. These seven categories are also used by the Mastery plugin. With the dashboard, students can compare their learning progression with other students who are working on the same project or solving the same problem on the gameful learning platform SAGE (Bender, 2015). It also provides useful information of student programming activities that assist instructors by providing real-time help to the students. We tried to design the more colorful and more interactive, so it will be more friendly to young people. Future works should aim to finish the process of combined the SAGE editor and dashboard together and also considering combining the current backend server and frontend server together into the same repo. It might also be a good idea to create a share account in mLab and Amazon AWS so all the developer for this project can have access to these services.

9.0 Reference

Basawapatna, A. R., Repenning, A., Koh, K. H., & Nickerson, H. (2013). The zones of proximal flow. Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13, 67-74.

Bender, J. (2015). Developing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula.

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: Lint-inspired Static Analysis of Scratch Projects. SIGCSE (pp. 215-220). Denver: ACM.

Cooper, S., Dann, W., & Pausch, R. (2000). Developing algorithmic thinking with Alice. Information Systems Educators Conference, 506–539.

Holman, C., Fishman, B., Aguilar, S. (2013). Designing a Game-Inspired Learning Management System. Poster for Games+Learning+Society Conference 9.0. University of Wisconsin-Madison. Madison, WI.

Wing, J. M. (2006). "Computational thinking," Communications of the ACM, vol. 49, no. 3, p. 33, Mar. 2006.

Koh, K., Basawapatna, A., Nickerson H., & Repenning A.(2014). Real Time Assessment of Computational Thinking. 2014 IEEE Symposium on Visual Languages and Human-Centric Computing(VL/HHC)

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. ACM Transactions on Computing Education, 10(4), 1-15.

Moreno-León, J., & Robles, G. (2015). Analyze your Scratch projects with Dr. Scratch and assess your Computational Thinking skills. Scratch 2015 AMS.

Moreno-Leon, J., Robles, G., & Roman-Gonzalez, M. (2015, September). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. RED-Revista de Educación a Distancia.

Murphy, C., Kaiser, G., Loveland, K., & Hasan, S. (2009). Retina: Helping Students and Instructors Based on Observed Programming Activities. SIGCSE (pp. 178-182). Chattanooga: ACM.

Pane, J. (2002). A programming system for children that is designed for usability (Unpublished doctoral dissertation). School of Computer Science, Carnegie Mellon University.

Repenning, A. (1993). Agentsheets: A tool for building domain-oriented dynamic, visual environments (Unpublished doctoral dissertation). Department of Computer Science, University of Colorado–Boulder.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60–67.

Scaffidi, C., & Chambers, C. (2012). Skill Progression Demonstrated by Users in the Scratch Animation Environment. International Journal of Human-Computer Interaction, 28(6), 383-398.

Seiter, L., & Foreman, B. (2013). Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. International Computing Education Research. San Diego: ACM.

Siemens, G., Gasevic, D., Haythornthwaite, C., Dawson, S., Shum, S.B., Ferguson, R., Duval, E., Verbert, K., Baker, R.S.J.d. (2011) Open Learning Analytics: an integrated & modularized platform: Proposal to design, implement and evaluate an open platform to integrate heterogeneous learning analytics techniques. Athabasca, Alberta, Canada: Society for Learning Analytics Research.