

# **Computational Thinking Learning Platform for SAGE**

**COMS 6901 Section 014**

**Sajal Khandelwal**  
**sk4226**

**Plaban Mohanty**  
**pm2878**

## Table of Contents

<b>1. Introduction .....</b>	<b>3</b>
<b>2. Related Work .....</b>	<b>5</b>
2.1. Coursera/MOOC.....	5
2.2. Scratch .....	5
2.3. React .....	6
2.4. GradeCafé .....	7
<b>3. Strategy and Planning .....</b>	<b>9</b>
3.1. Features.....	9
3.1.1. Instructor Dashboard .....	10
3.1.2. Student Dashboard .....	17
3.2. Architecture.....	21
3.3. Timeline .....	22
<b>4. Implementation .....</b>	<b>22</b>
4.1. Dashboard .....	22
4.1.1. DB Models .....	22
4.1.2. Instructor Dashboard .....	23
4.1.3. Student Dashboard .....	24
4.2. Node Server .....	25
4.3. Recommendation Engine .....	30
<b>5. Conclusion .....</b>	<b>31</b>
<b>6. Future Work .....</b>	<b>32</b>
<b>7. Acknowledgements.....</b>	<b>33</b>
<b>8. References .....</b>	<b>34</b>

# 1. Introduction

Education especially for young students has been a widely researched and debated topic involving multiple disciplines of psychology, human behavior, technology etc. With the vast progress in science and technology, it has become a trend in education systems around the globe to experiment in ways to perfect the approach towards teaching young children in a way that is both enriching for them as well as enjoyable. One of the major areas of education research is towards development of proper formats of teaching computer science and computational thinking concepts as these fundamental concepts are generally challenging to be taught in a regular classroom environment.

With the advent of research in human behavior and conclusive results from human responses, it has been shown that visual interaction methods including but not limited to games, storytelling etc. capture the interest of young audiences and help them grasp the concepts better. Rather than general lectures and regular theory based assessments, a gameful approach of education piques the attention of students and leads them on a path of discovery and learning. Visual interaction methods usually supplemented by incentives and rewards have been widely applauded as a highly useful tool in helping students learn. This has been widely supported by experiments that students tend to remember or understand concepts clearly when accompanied by visual tools and when they themselves are involved in a construction/creation based learning environment. Games, especially simple educational games have been shown to be both efficient in capturing the attention of students and teaching them in an engrossing and enjoyable manner.

While these visual interactive techniques are in general a useful instrument for education, they are especially helpful in teaching computational concepts and computer science fundamentals. Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science<sup>[1]</sup>. Teaching computational thinking to students at young age could prove to be highly helpful even if they don't pursue computer science as a subject later. It equips them with skills on approaching problems in general and ways to tackle them. It inculcates a sense of algorithmic thinking to help them draw solutions in any field they choose to pursue. It is more about conceptualizing than programming and involves learning about various tools like recursion, parallel processing, abstraction and decomposition. As this is more conceptual and fundamental, it is difficult to teach these by theoretical means. Rather a more hands on

approach is required to teach students their utility and benefits. This is where the interactive gamified approach shines through.

Teaching computer science fundamentals through games have always been regarded as efficient <sup>[2]</sup>. However, the same could also be used for computational thinking concepts. As these are derived from the very same basics as computer science fundamentals, the way in which students perceive this would also be similar. Using tools like Scratch<sup>[3]</sup> and other design tools, simple games could easily be created to teach students the CT concepts that we earlier talked about. But, just having these games as a teaching tool would not be sufficient. The classroom teaching mode has worked for generations as it involves instructors, education institutions that ensure the student's growth by tracking their progress, providing help when needed, recognizing the student's weaknesses in certain areas and helping them strengthen those. For efficient education, we need to create a platform where gamified learning could be combined with a similarly interactive and intuitive interface to help a student learn the CT concepts, provide valuable feedback, suggest customized recommendations to improve and provide proper incentive to progress along with retaining the instructor's ability to create customized courses, learning paths for students.

SAGE (Socially Addictive Gameful Engineering)<sup>[4]</sup> intends to create the perfect platform by combining all these elements to form a collaborative game based learning system which infuses computational thinking into young students in a highly intuitive and integrated manner. In SAGE, not just the way of teaching CT concepts by games, but the platform which exposes students to the same is gamified as well. Games or assessments in SAGE is built on Scratch which provides a highly user friendly way of creation of interactive ways of teaching students by gamified assessments. Scratch has been widely used as educational tool for teaching computation and CS theory and programming because of its graphical and self-explanatory tone.

Our primary motivation behind this project was to build an integrated platform for all of SAGE's components and to make it as intuitive as possible for both students and instructors to use this as a medium for education. We wanted to develop an online scalable interface that would essentially be a one stop solution for instructors to design courses, build learning paths for students progression<sup>[5]</sup>, create assignments and write assessments for the same and for students to play the courses, get valuable recommendations, earn badges as incentives etc. We also wanted to make it scalable in a way that any future work or development would be able to fit in seamlessly. We built SAGE dashboards to include all components, their functionalities, use cases and integrations and presented it to the users in an intuitive, graphical interface. The dashboards were designed individually for students and instructors with possibility of development of other dashboards for extra users like researchers. The entire purpose

of the dashboard framework built around SAGE was to encompass all the functionalities of the platform under a single umbrella to be easily accessed and make it globally explorable.

## **2. Related Work**

### **2.1. Coursera/MOOC**

Recently there has been a massive surge in the world of online education through Massive Open Online Courses or MOOCs. This has opened up the previously constrained education from some of the finest institutes in the world to a global audience. One of the major organizations that launched from Stanford was Coursera. It is a MOOC platform designed to make education in a wide variety of topics globally accessible. Instructors and institutions can design courses with video recordings of lectures and other forms of coursework. Students can access this material from anywhere in the world globally. The courses are designed to have assignments customized for an online platform. Coursera also has a progression of courses with increasing difficulty culminating in mastery over a subject. The platform provides an all-inclusive package of course, recommendations, adaptive progress etc. We wanted to make SAGE globally accessible and hence wanted to create a similar platform to coursera for our dashboard .Many of the features implemented such a Learning paths/quests have been inspired from their counterparts in Coursera

### **2.2. Scratch**

Scratch is a visual tool designed especially for programming games, stories etc. in a highly creative and presentable manner. It has been widely used as a platform to teach young students how to think creatively , learn programming fundamentals and collaborate globally. The platform has many educational games in its repository , most of which are designed keeping in mind young audiences. The versatility of the tool as well as its intuitive nature makes it an ideal interface both to design games and play them.

Scratch has a wide online community of designers and developers who lend valuable advice on creation of games and other visual tools for education. One of Harvard's most popular courses CS50 starts its computer science fundamentals using Scratch.

## 2.3. REACT

Real Time Evaluation and Assessment of Computational Thinking<sup>[6]</sup>, is an assessment tool to allowing teachers to see which high-level concepts students have mastered and which ones they are struggling with as students code in real time.

REACT has three technical objectives. Firstly, it aims to provide a web based interface for the tool. Secondly, it communicate students' progress information to teachers hierarchically, allowing teachers to quickly get a high level sense of the entire class. Lastly, it also provides teachers the most useful representations of class/individual progress allowing them to make effective instructional decisions.

An example dashboard showing every student's progress is shown below.

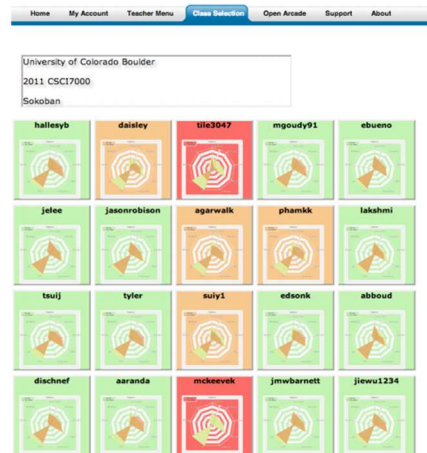


Figure 1. Example of REACT Dashboard

This can be further expanded to show a detailed Computational Thinking Pattern graphs illustrating student's skills and learning at the semantic level, as shown below.

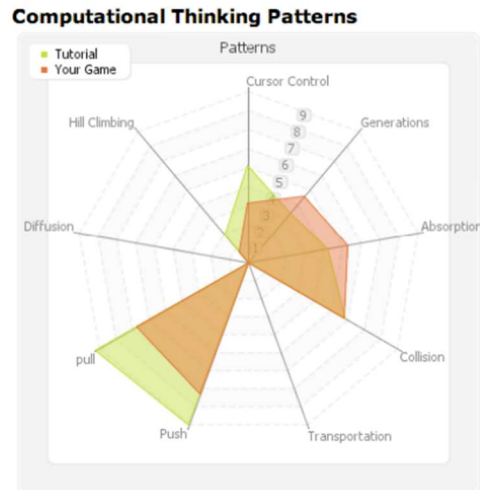


Figure 2. CTPA Graph

## 2.4. GradeCraft

GradeCraft<sup>[7]</sup> is a game-inspired learning management system. Researchers behind GradeCraft started with basic gameful mechanics, such as - using points and incremental levels instead of grades; awarding badges to recognize achievements and skill-acquisition; allowing students to redo assignments; giving students the choice of assignments; having students work together in both self-selected and pre-arranged groups and many more - and devised a interface in the form of GradeCraft, to build further on nuanced gameful functionality in the future.

GradeCraft, primarily, has 2 dashboards - Student and Instructor dashboards.

### 2.4.1 Student Dashboard

An example snapshot of the Student Dashboard, displaying comprehensive course progress, is provided below. The top portion contains information pertaining to student's own progress in terms of points won, badges earned, predicted points and more. This information has a motivational effect on the students, as preliminary research indicates that this type of display boosts user motivation to complete tasks.

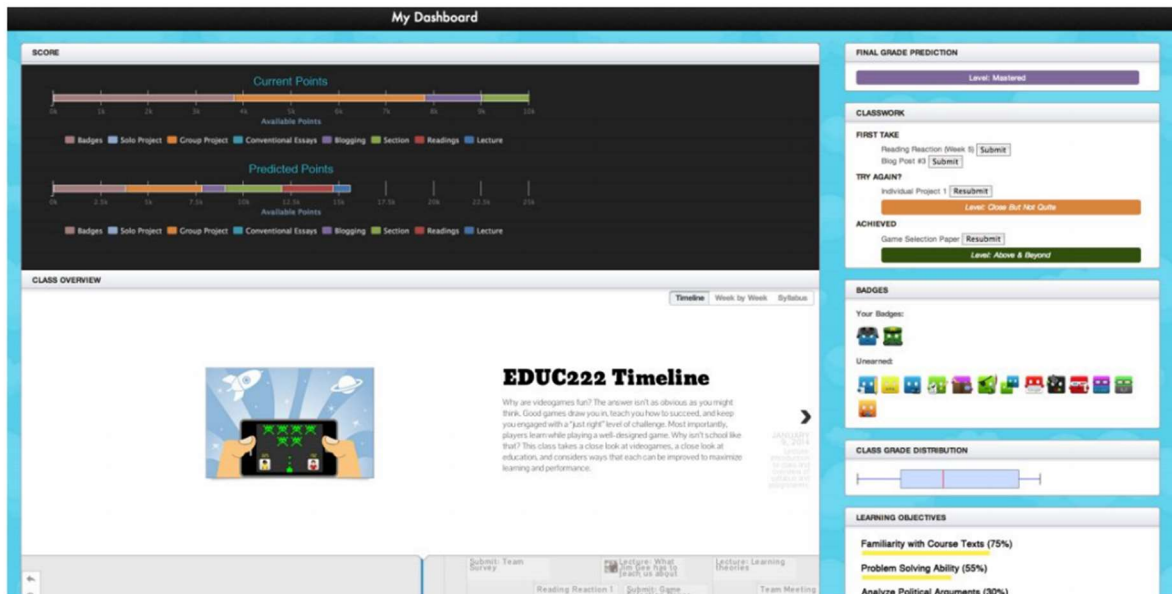


Figure 3. Student Dashboard

The dashboard also shows a To-Do list for upcoming assignments, assignments that could be re-submitted to improved performance, progress towards achieving the course learning objectives, and distribution of student's own performance against his or her own peers.

## 2.4.2 Instructor Dashboard

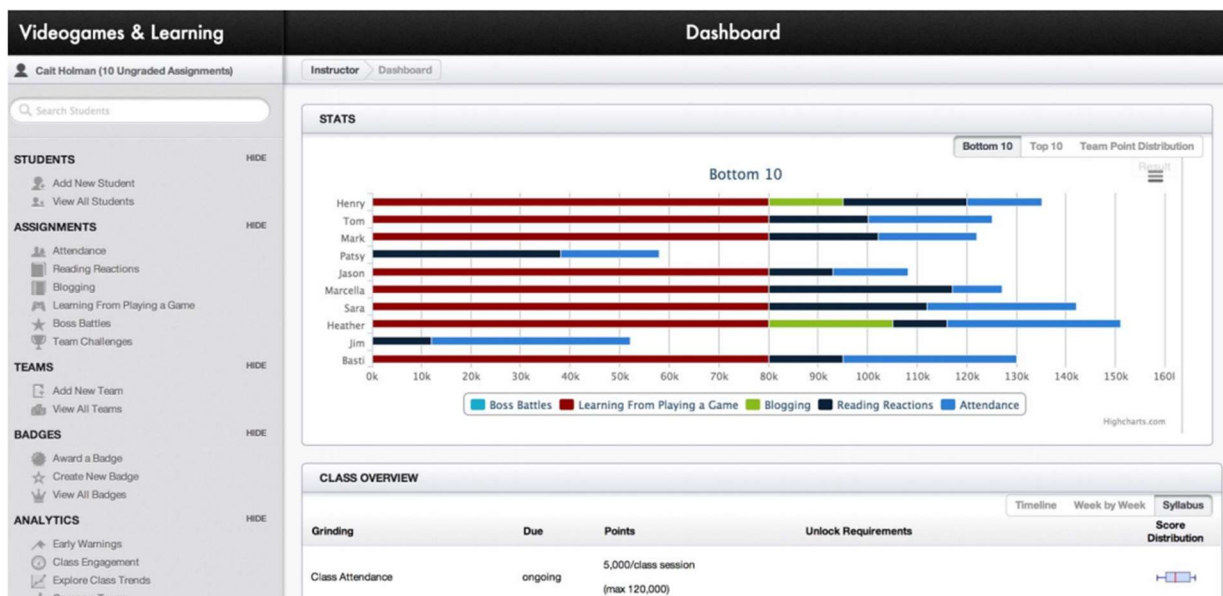




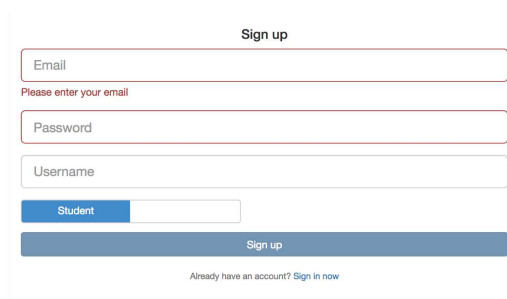
Figure 4. Instructor Dashboard

Instructor Dashboard, as shown above, provides the performance the class within a single view. Pedagogies that present more choice to learners and result in a broader variety of representations of learning are more difficult to manage than “traditional” didactic pedagogies (e.g., Crawford, 2000). Therefore, GradeCraft equips the instructors with tools and metrics to better manage the gameful structure of the class itself. It shows the top and bottom percentile of the class in form of the stacked bar charts. This assists instructors to identify the students in need earlier. 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.

## 3. Strategy and Planning

### 3.1. Features

The dashboard can be mainly divided into two parts - Students’ Dashboard and Instructors’ Dashboard. A common Login and Signup page, as shown below, serves as a point of entry to the respective dashboards.



The screenshot shows a 'Sign up' form with the following elements:

- Sign up** (header text)
- Email** (input field with a red border and the placeholder text 'Please enter your email')
- Password** (input field with a red border)
- Username** (input field)
- Student** (radio button selected in a role selector)
- Sign up** (blue button)
- Already have an account? [Sign in now](#) (link at the bottom)

Figure 5. SignUp Screen

The register page allows user to register for a student account or an instructor account. It asks 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. On successful registration, the user is redirected to the Login page as shown below.

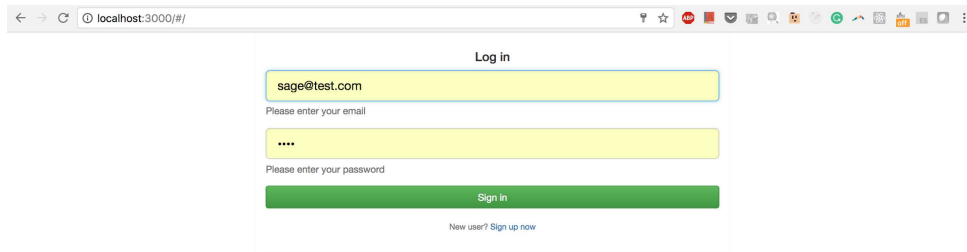


Figure 6. Login Screen

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.

### 3.1.1 Instructor Dashboard

To make the SAGE platform an integrated framework for learning, it is vital that the instructors use the same platform as students albeit with different capabilities. The application being centralized and well-integrated, all the data can be accessed in real time by the instructors for their feedback, which in turn creates an actual classroom environment with all its benefits in the online platform as well.

The instructor dashboard is the primary interface for the teachers/instructors to interact with SAGE. The instructor dashboard provides an interactive environment where instructors can login, create courses, learning paths, design assignments and assessments for students, view statistics etc. The dashboard provides a single gateway for instructors to access all of SAGE's functionalities. The most useful use cases for the instructors are described below:

#### 3.1.1.1 Creation of courses

Courses are the main unit of education in SAGE. Each course handles the teaching of one or two subjects at a particular mastery level. Each course is designed by an instructor, who can provide the required course materials which might include video lectures, tutorials, pdfs and other such course materials that provide a theoretical base to understanding the subject. The instructor can then create assignments specific to the course in increasing levels of difficulty. A guided tutorial can be provided on the way to approach the assignments.

In the instructor dashboard , an instructor can create courses by clicking on the Courses/Learning Path tab on the left pane :

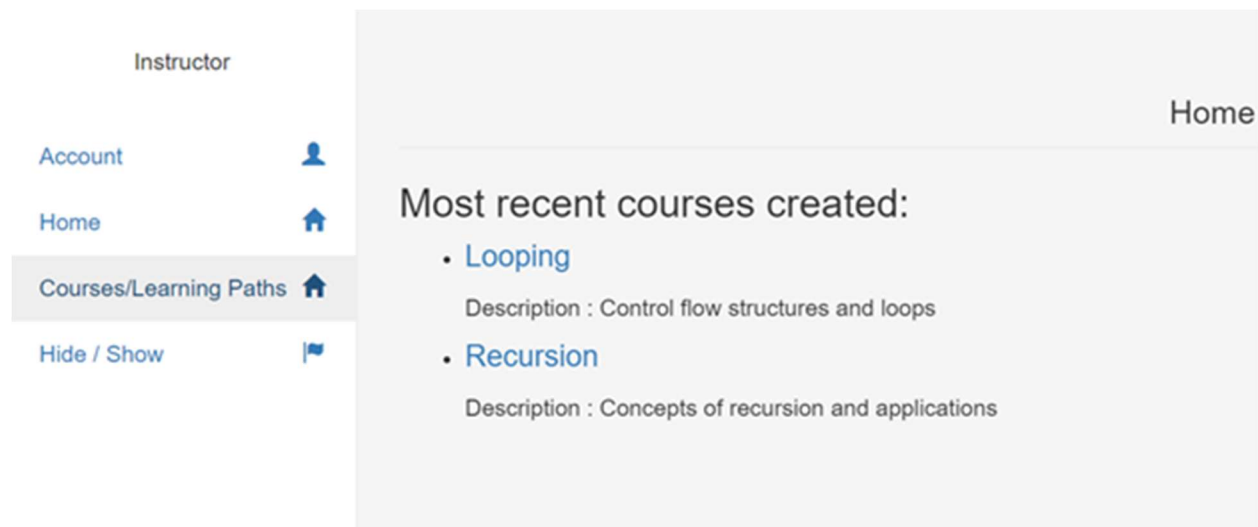


Figure 7. Home page for instructors

This opens a new view with all created courses and Learning Paths visible .

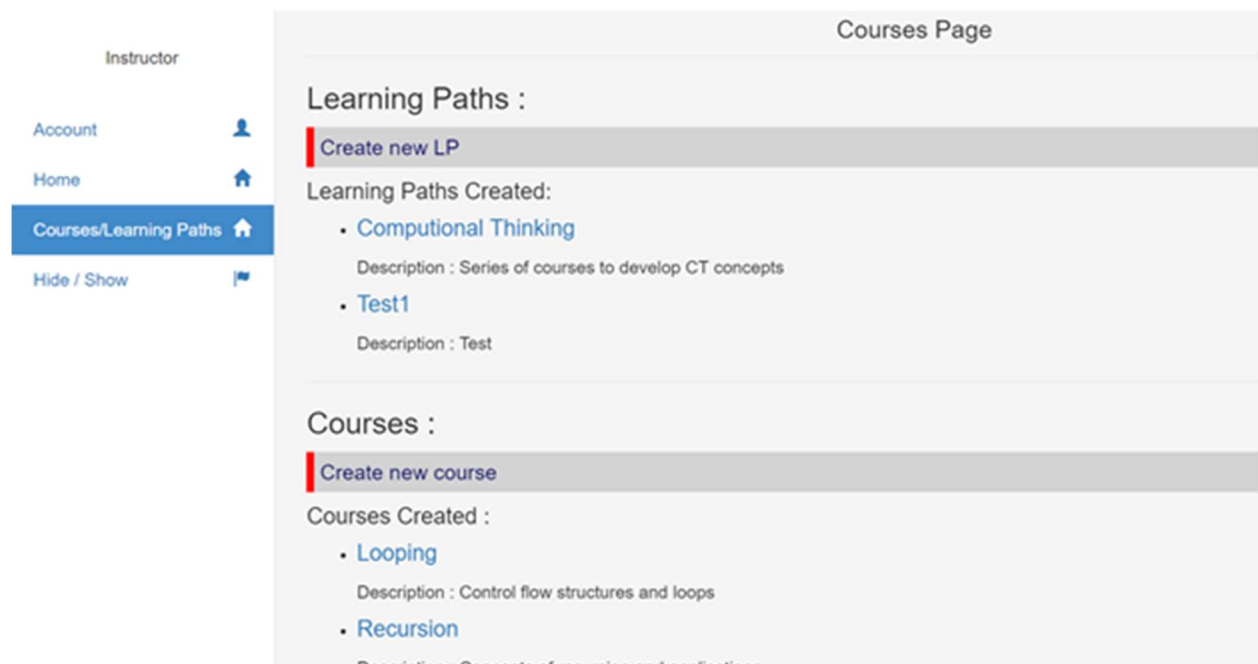


Figure 8. Courses Page

Clicking on Create new course under Courses in this page opens up the form to create new courses.

The image shows a web interface for creating a course. On the left is a sidebar with the following items: 'Instructor' at the top, followed by 'Account' with a person icon, 'Home' with a house icon, 'Courses/Learning Paths' with a house icon and highlighted in blue, and 'Hide / Show' with a flag icon. The main content area is titled 'Create Course' in a blue header. Below the header are five text input fields: 'Course Name', 'Course Description', 'Age Group (10-12)', 'Subjects', and 'Other features (separated by comma):'. At the bottom of the form are two blue buttons: 'Reset' and 'Save'.

Figure 9. Course Creation Form

The instructor can then fill in the necessary details and click on save which creates a course in the SAGE platform.

#### 3.1.1.2 Creation of assignments

Assignments are the way of evaluating the students' understanding of the course material. This is the prime application of using gamified way to provide students with a practical approach to applying the concepts. Instructors can create and design the assignments inside the dashboard itself.

After creating a course and clicking on Save from the last section, the dashboard redirects to an assignment creation page where an instructor can create an assignment by specifying the number as the order in which the assignment should appear.

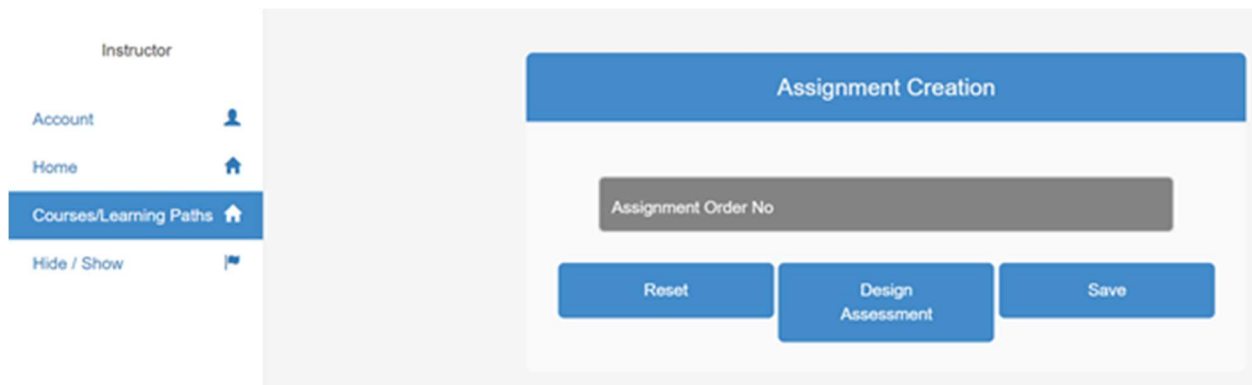


Figure 10. Assignment Creation Form

After specifying the order no, clicking on Design Assessment saves the assignment and redirects to a view containing a flash object wherein the assignment could be designed.

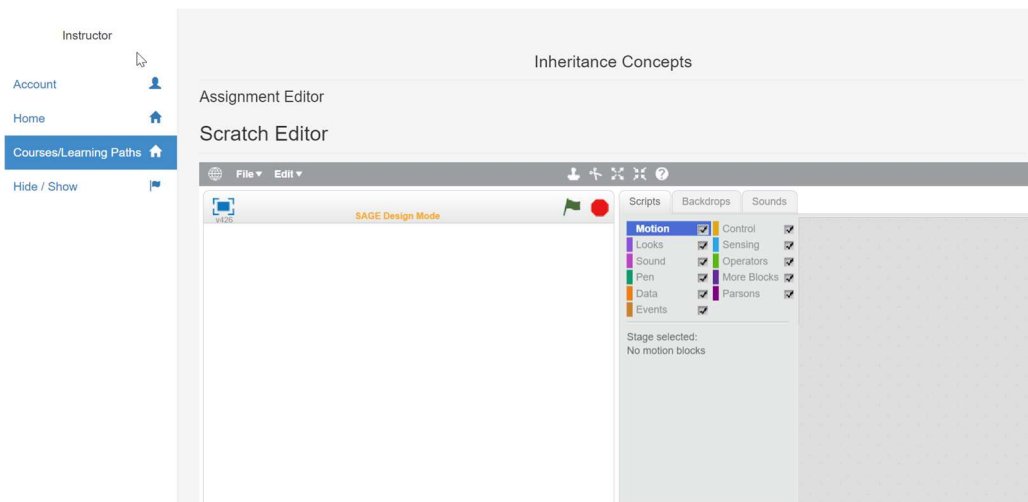


Figure 11. Assignment Designer

This can also be used to create a new assignment for an already existing course . From the courses/LP page in the previous section , click on any course under the courses created section under Courses. This leads to the course view which shows the course description , its resources as well as the assignments that are already a part of the course.

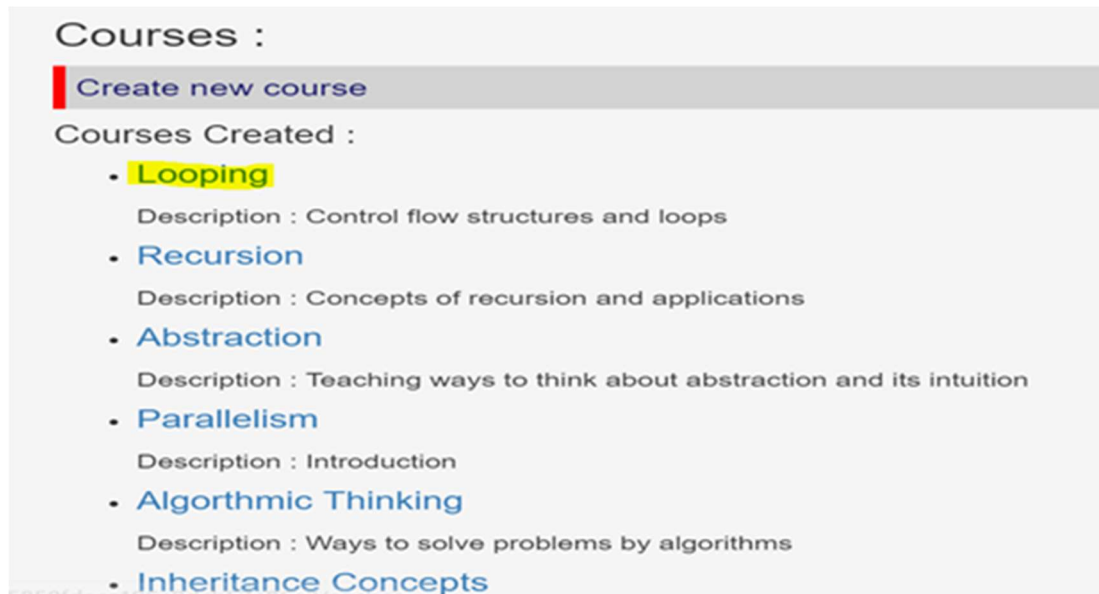


Figure 12. Course List

Clicking on Create new Assignment takes us to the same view as before.

### 3.1.1.3 Creation of Automatic Assessments

One of the prime motives to make the online platform for SAGE is to automate the entire process of gamified education so that students across the globe can access it to learn CT concepts. With the assignments in place, there has to be an automated assessment service that judges the students' performance in the assignments and provides a score accordingly.

For this we use what is known as the Visual Assessment Editor that lets the instructor design assessments for the assignments. The VAE was earlier a separate component that the instructors had to access with a different server . But with the dashboard integration , they can create these assessments in VAE from the platform itself.

Clicking on any course takes us to the course page with all the assignments under that course clearly listed. Clicking on any of these assignments takes us to the VAE where instructors could design assessments.

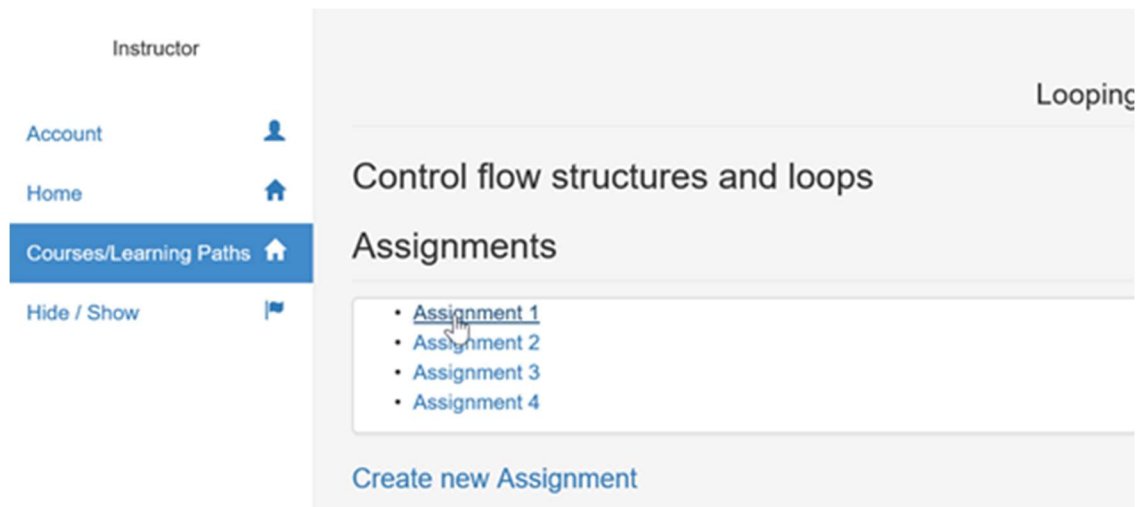


Figure 13. List of Assignments under a course

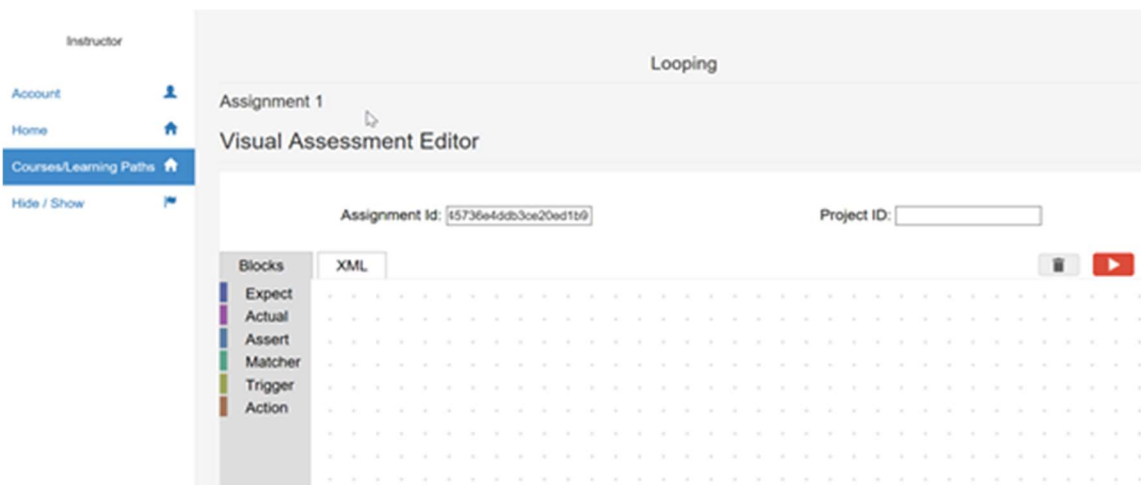


Figure 14. VAE integrated into dashboard

### 3.1.1.4 Creation of Learning Paths /Quests

In many scenarios , a single course can be connected to other course as a step in a progression. Progression models are highly useful in helping students gain fundamental mastery over one or more CT concepts. While one course might deal in the basics of a concept, another course might deal with advanced level applications of the same concept and yet another could involve constructing programs or applications from the concept. Hence, the dashboard provides the instructor the ability to create what we call as a learning path. This is essentially a progression model which consists of multiple

courses which are focused on a single concept or similar topics with each course being a step for the next.

The instructor can create the Learning Path by clicking on the courses/Learning Paths tab on the left pane. The view then displays the list of all Learning Paths created. to create a new learning path, the instructor can click on create new learning path as shown below.

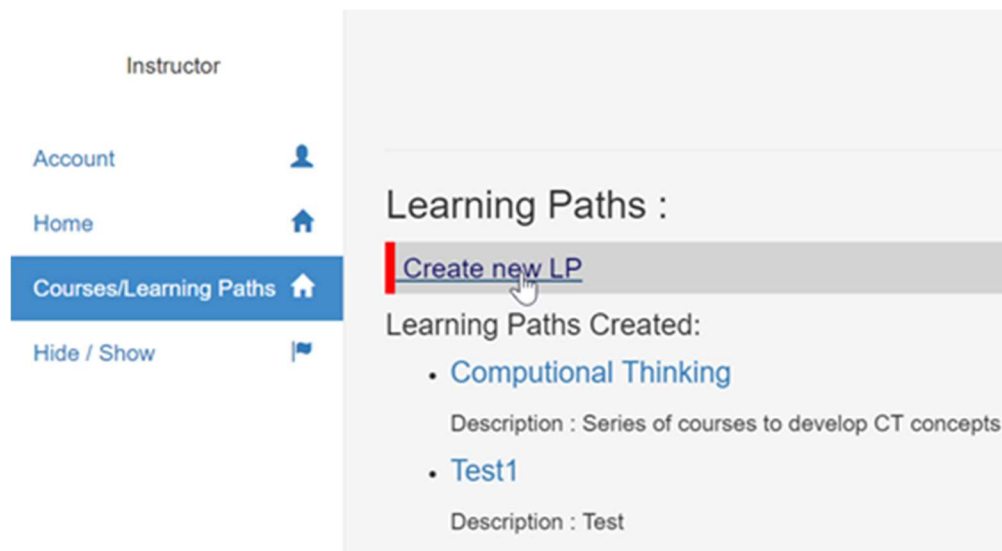


Figure 15. Creating new LP

This gives us a form which can be filled based on requirements and saved.

The screenshot shows a form titled 'Create LP' with a blue header. The form contains five input fields: 'LP Name', 'LP Description', 'Age Group (10-12)', 'Subjects', and 'Other features (separated by comma)'. At the bottom of the form, there are two buttons: 'Reset' and 'Save'.

Figure 16. LP creation form

On creating a LP, instructors can add courses to it by just clicking on the list of available courses.



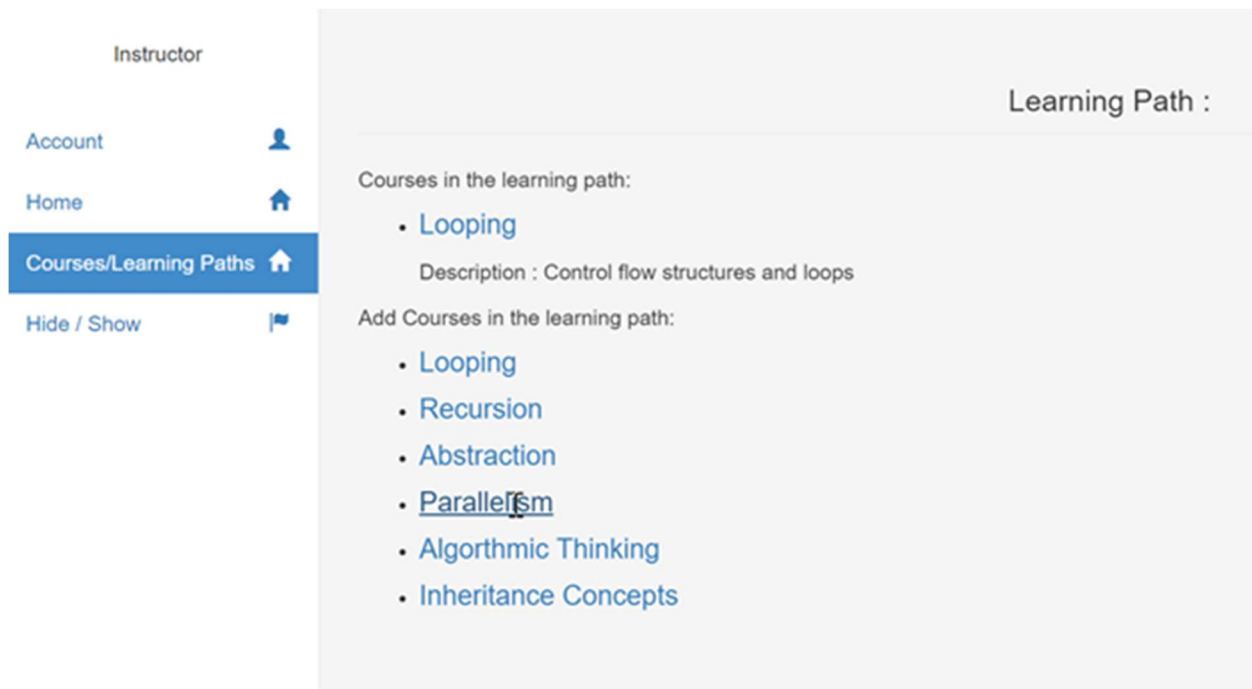


Figure 17. Adding courses to LP

Apart from these features, the dashboard provides the ability to view the courses and Learning Paths created and the provision to add statistics for the respective courses as well .

### 3.1.2 Student Dashboard

The student dashboard is a one stop platform for a student to browse and enroll in new courses, learning paths, track progress and evaluations, carry out assignments and other use cases explained in further detail below. Having all these features under one platform provides a much more integrated experience for a student than separate components serving these use cases would.

#### 3.1.2.1 Home Page

The student is directed to a Home page, as shown below, as soon as he or she logs in. The student is shown some recommended courses based on his or her and peers' enrollments. Below that the recent courses are displayed. Placeholders for displaying student specific metrics and badges earned are provided. These would provide relevant information when the models are implemented in future.

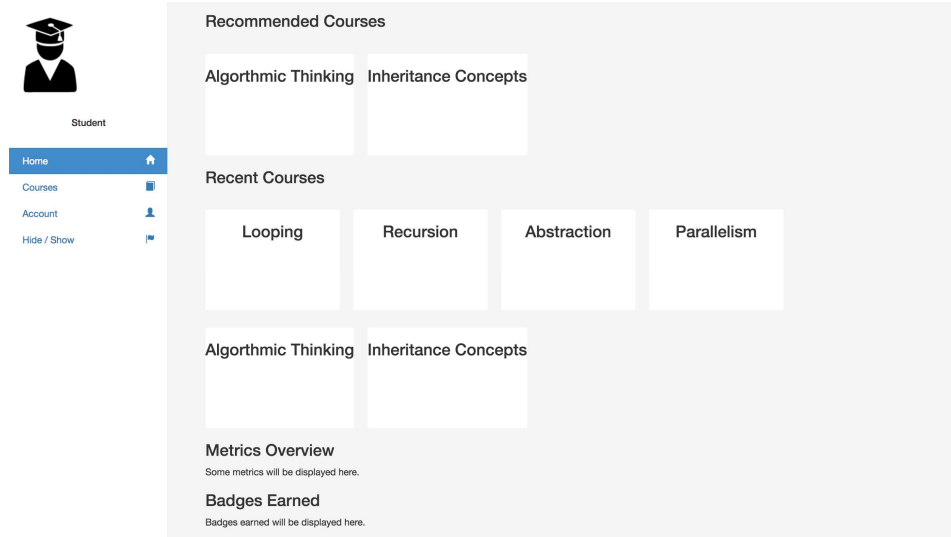


Figure 18. Student Home Screen

The student can navigate to appropriate screens or pages through the navigation bar.

### 3.1.2.2 Course Page

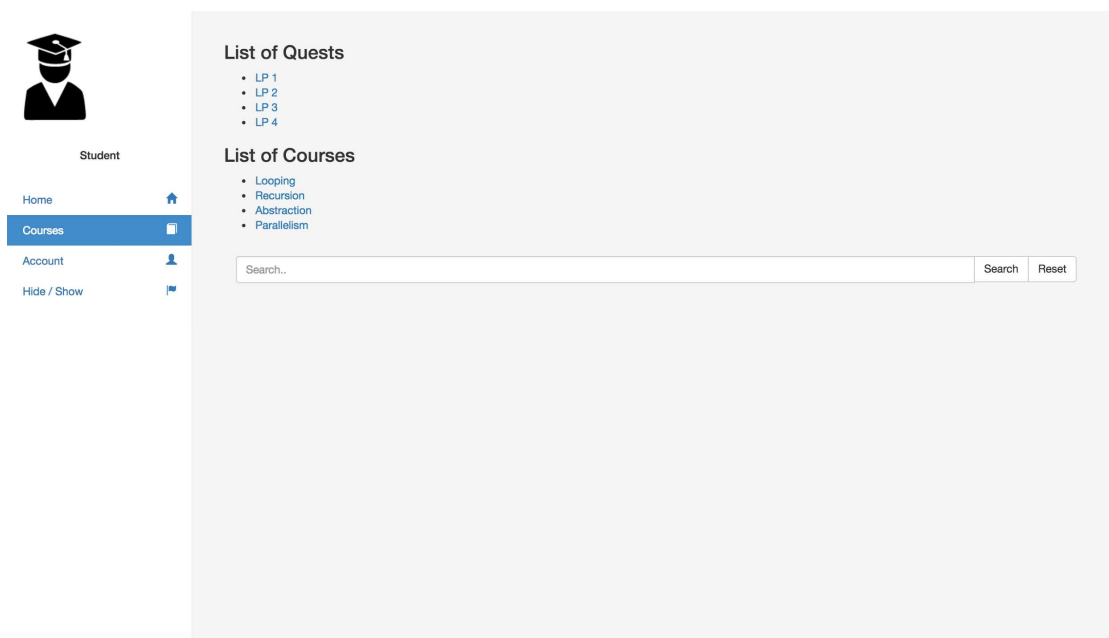


Figure 19. Student Enrollment Screen

On clicking on “Courses” on the navigation pane, the student is redirected to his or her own enrollments. These enrollments can be of “Learning Progressions”, otherwise called “Quests” (comprising of two or more courses), or simply of atomic “Courses”. An instance of such enrollments is shown in the figure above. A placeholder is provided to search for “Courses” or “Quests”. These would display the results when implemented.

On selecting one of the enrolled courses, the student is taken to a specific dedicated Course Page for that particular course, as shown below.

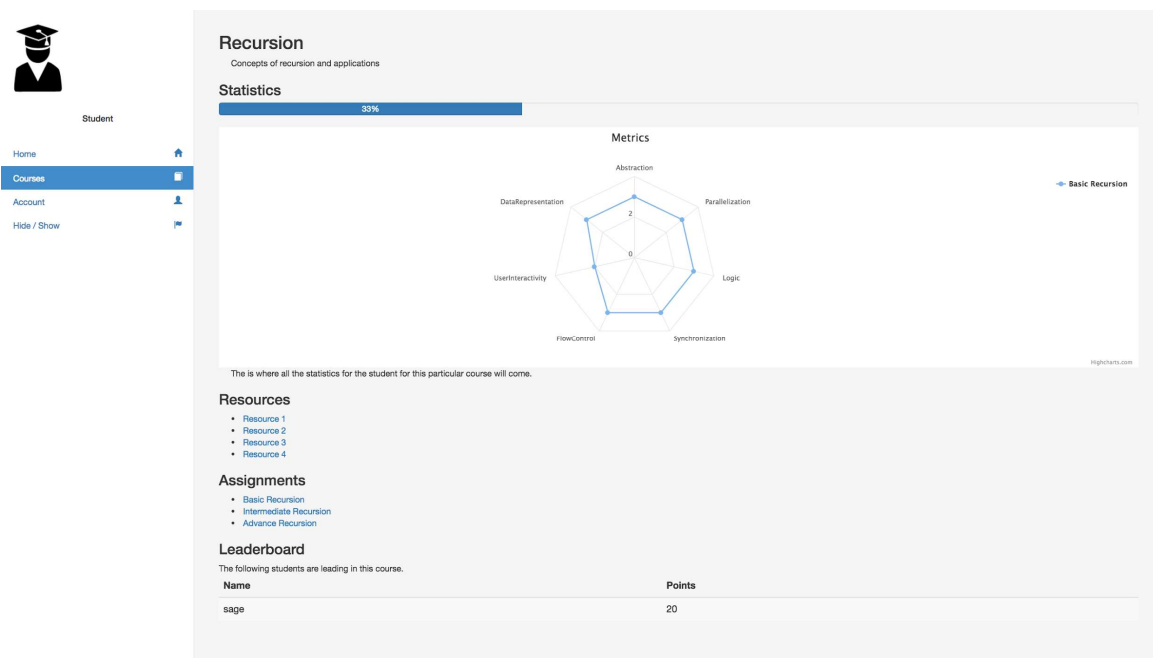


Figure 20. Course Screen

On top is the name of the course and its description.

Under “Statistics”, student is show his or her progress made in terms of the number of assignments completed in the form of a progress bar. The student is also provided with a “Spider Web”, to show the evaluation of different Computational Thinking concepts, namely - Abstraction, Parallelization, Logic, Synchronization, Flow Control, User Interactivity, and Data Representation - of the attempted assignments.

The student can check external resources provided by the instructor of the course.

At the bottom a leaderboard is provided of the top performing students to help the students engage in healthy competition, leading to better engagement.

### 3.1.2.3 Assignment Page

On selecting one of the assignments on the Course page, the student is taken to an Assignment Page, as shown below.

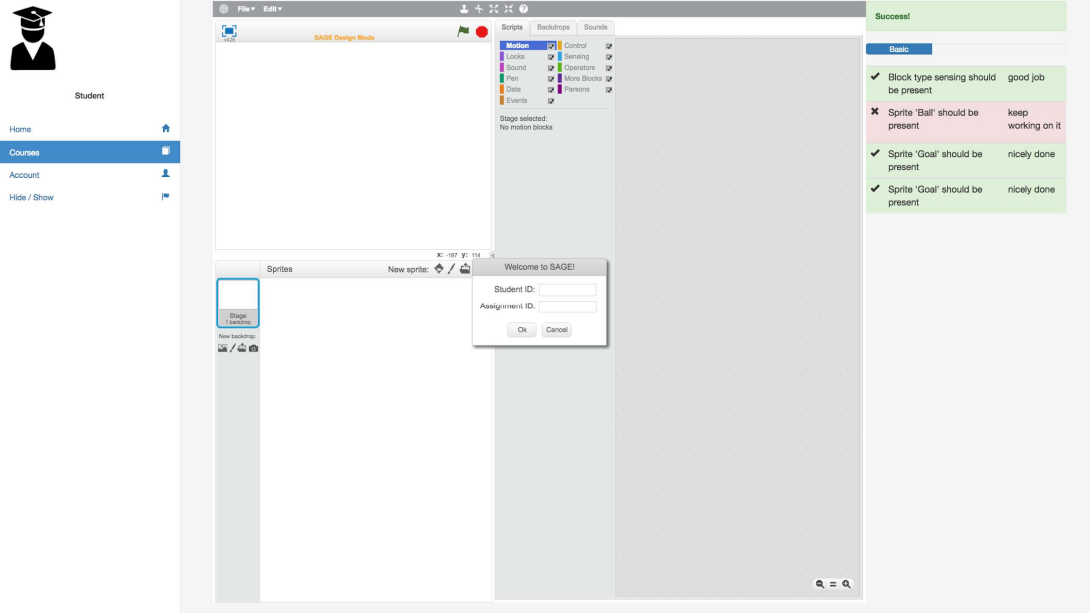


Figure 21. Assignment Screen

This page opens the Scratch Editor where the student can carry out the assignment. Next to it is a pane where there are placeholders for student to get real-time feedback. It displays any notification that may pop up, proficiency in CT concepts and the tasks with respect to that assignment that are complete/incomplete.

## 3.2.Architecture

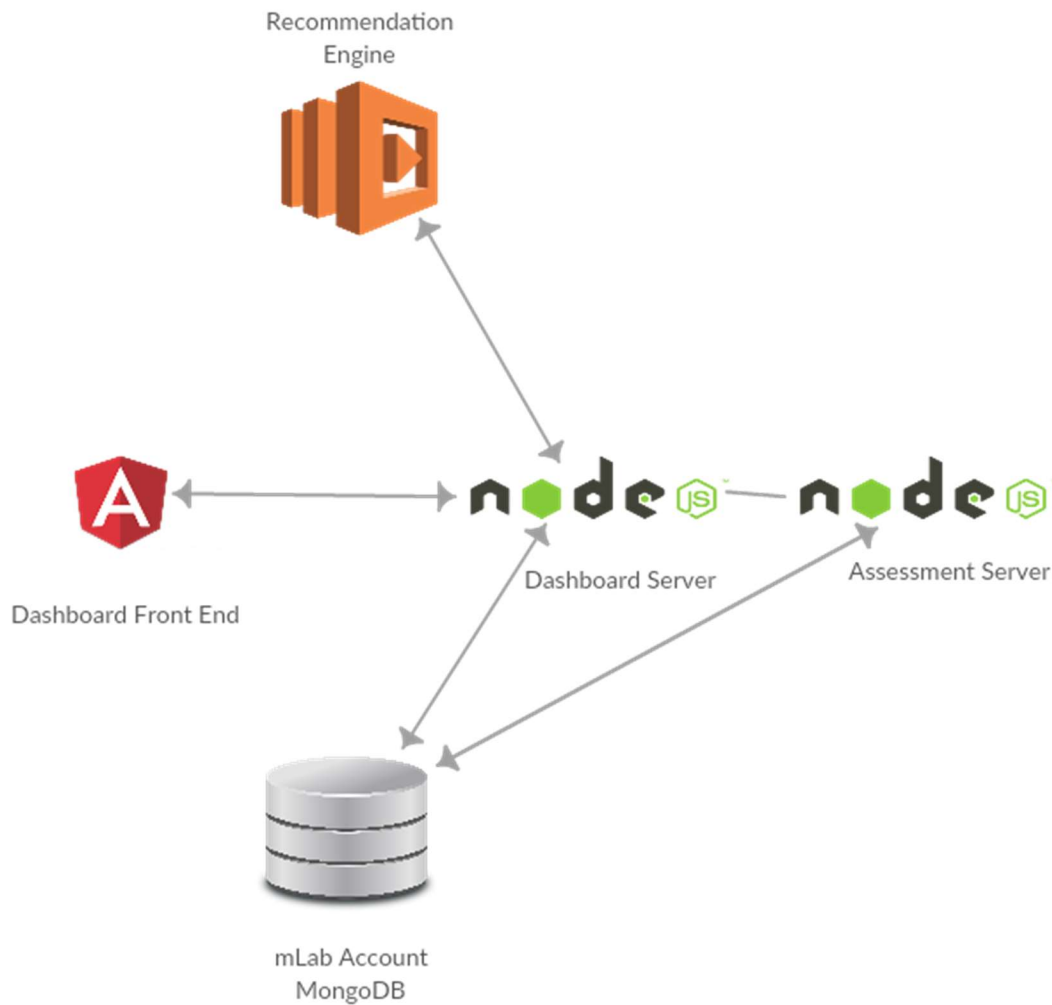


Figure 22. Architecture Diagram

The diagram above depicts a high level architectural diagram of the Computational Thinking Platform. It is a single page web application with front end developed in AngularJS. It receives the data in JSON format from Dashboard NodeJS server through series of AJAX calls REST APIs. The Assessment Server microservice handles the data associated with storing the files related to assignments' creation and design and students' progresses, and running evaluations using the Hairball plugin. Dashboard server receives the recommendations to through a separate Recommendation Engine deployed on AWS Lambda. All the structured data is stored in MongoDB.

### 3.3. Timeline

Tasks	Date Completed
Analysis of existing systems and installation of components	March 10th
Planning out the control flow and integration plan	Mar 25th
Building the dashboard structure for student and instructor	Mar 27th
Midterm Presentation	Mar 31st
Adding scores plugin, evaluation and other metrics to student dashboard	April 14th
Creating the recommendation engines	April 28th
Preparing documentation for SAGE platform and dashboard	May 4th
Final Report and presentation	May 5th

## 4. Implementation

### 4.1 Dashboard

The dashboard maintains its own server and database to serve the platform. Functionality has been added to support use cases such as courses, learning progressions and carrying out assignments on the same platform, rather than visiting a separate link.

GitHub repo can be found at - <https://github.com/cu-sage/sage-frontend>

Documentation detailing the instructions as to how to setup the server is submitted separately.

#### 4.1.1 Database Models

The database in MongoDB and Mongoose ORM has been continued from last time and the models have been expanded to support the aforementioned use cases.

There are six collections in the database -

1. assignmentModel
2. courseModel
3. learningPathModel
4. enrollmentModel
5. enrollmentLearningPathModel
6. userModel

The assignmentModel contains information relevant to it such as a unique identifier and assignment name. courseModel has information like unique identifier, course name, its description and the assignments a particular course entails. learningPathModel, in addition to identifier and name, contains the courses that make up a particular learning model. Both enrollment models - enrollmentModel and enrollmentLearningPathModel store all the enrollments of the students.

#### 4.1.2 Instructor Dashboard

The instructor dashboard has a single page web UI with a backend server powering its services .

##### 4.1.2.1 Front End

The front end is a single web platform with all use cases integrated. The features as described above have been exposed by HTML and Angular JS based web views. The default home page of Student Dashboard, provided one is authenticated, is

```
/instructor/#/home/:sid
```

##### 4.1.2.2 Node Back end

REST based API calls have been used to provide the required functionalities for the instructor dashboard . Detailed description of all the routes implemented can be found on the the project's Github wiki page

Some of the most useful routes are as follows:

HTTP Verb	Route
POST	/instructors/createCourse/:id
POST	/instructors/createLP/:id
POST	/instructors/:id/course/:cid/createAssignment
POST	/instructors/:id/LP/:LPid/addCourse/:cid
GET	/instructors/coursesby/:id
GET	/instructors/LP/:id

### 4.1.3 Student Dashboard

The Student Dashboard implementation can be thought of having 2 decoupled, but related, components - a single page web application front end, and a NodeJS backend server, supporting REST APIs.

#### 4.1.3.1 Front End

Front-end has been developed in AngularJS, HTML, CSS and Bootstrap. Angular router is used to navigate between different screens. The default home page of Student Dashboard, provided one is authenticated, is -

/student/#/home/:sid

Few external libraries such as HighChart.js have been used.

#### 4.1.3.2 Node Back End

The Node server back end is used to serves REST web APIs accessed by the front end through a series of AJAX calls.

A list of Student Dashboard specific routes are tabulated below. Detailed information pertaining to requests, headers and responses can be found on project's GitHub's repository's Wiki page.



HTTP Verb	Route
GET	/recommendedCourses/student/:sid
GET	/recentCourses/student/:sid
POST	/enroll/:cid/student/:sid
GET	/coursesEnrolled/student/:sid
GET	/course/:cid/leaderboard
GET	/course/:cid/student/:sid

## 4.2. Assessment Server

The assessment server has been revamped. This microservice only handles the data associated with storing the files related to assignments' creation and design and students' progresses, and running evaluations using the Hairball plugin.

A separate documentation is submitted on how to set-up this server.

GitHub repository can be found at - <https://github.com/cu-sage/sage-node>

### 4.2.1 Models

There are two primary models on this server.

#### 4.2.1.1 Assignment Model

```
let mongoose = require('mongoose');
let idValidator = require('mongoose-id-validator');
let Schema = mongoose.Schema;
let ObjectId = Schema.Types.ObjectId;

var AssignmentSchema = new Schema({
  assignmentID: {
    type: ObjectId,
    required: true
  },
  instructorID: {
    type: ObjectId,
    required: true
  },
  sb2FileLocation: {
    type: String,
    required: true
  },
  assessmentXML: {
    type: String
  }
});

AssignmentSchema.plugin(idValidator);

module.exports = mongoose.model('Assignment', AssignmentSchema);
```

Figure 23. Assignment Model Schema

This model stores the information related to the assignment created by the instructor.

- **assignmentID** : This is the unique identifier for the assignment.
- **instructorID** : This is the unique identifier for the instructor who created this assignment
- **sb2FileLocation** : This string contains the location of the .sb2 file created by the instructor while designing a particular assignment.
- **assessmentXML** : This is the stringified XML of the assessment written in Visual Assessment Editor.

#### 4.2.1.1 Progress Model

```
let mongoose = require('mongoose');
let Schema = mongoose.Schema;
let ObjectId = Schema.Types.ObjectId;

var ProgressSchema = new Schema({
  studentID: {
    type: ObjectId
  },
  assignmentID: {
    type: ObjectId
  },
  lastUpdatedsb2FileLocation : {
    type : String
  },
  progressJSON : [],
  'results' : {},
}, {
  toObject: {
    virtuals: true
  },
  toJSON: {
    virtuals: true
  }
});

var Progress = mongoose.model('Progress', ProgressSchema);

module.exports = Progress;
```

Figure 24. Progress Model Schema

This model stores the information related to the progress made by a particular student for a particular assignment.

- **assignmentID** : This is the unique identifier for the assignment.
- **studentID**: This is the unique identifier for the student attempting the particular assignment.
- **lastUpdatedsb2FileLocation** : This string contains the location of the latest .sb2 file created by while attempting the assignment.

- **progressJSON** : This array of objects contains the periodic update of student's progress with timestamp.
- **results** : This object contains the evaluations received from the Hairball plugin.

## 4.2.2 REST API Routes

There are primarily 5 routes to handle data associated with storing the files related to assignments' creation and design and students' progresses, and running evaluations using the Hairball plugin.

### 4.2.2.1 Creating new assignment

This route creates a new assignment in the assessment server and returns the created object.

Resource URL - `POST /assignment`

Headers - Content-Type : `multipart/form-data`

Body Parameters - `assignmentID`, `instructorID`, `sb2File`

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** localhost:8081/assignment
- Authorization:** (empty)
- Headers:** (1)
- Body:** (selected)
  - Form: form-data
  - Fields:
 

Key	Value
assignmentID	58d845736e4ddb3ce20ed1b9
instructorID	5850fdacc196e9dcb57e86e2
sb2File	Choose Files testFile.sb2
- Pre-request Script:** (empty)
- Tests:** (empty)
- Status:** 200 OK
- Time:** 132 ms
- Response:**

```

1 {
2   "_v": 0,
3   "assignmentID": "58d845736e4ddb3ce20ed1b9",
4   "instructorID": "5850fdacc196e9dcb57e86e2",
5   "sb2FileLocation": "uploads/assignments/1493864867856testFile.sb2",
6   "_id": "590a91a3a2e284335372e9c6"
7 }

```

Figure 25. New Assignment Creation Response

#### 4.2.2.2 Updating assessment XML

This route updates the assessment for a particular assignment and returns the updated object.

Resource URL - PUT /assignment/:id/update\_xml

Headers - Content-Type : application/json

Body Parameters - assessmentXML

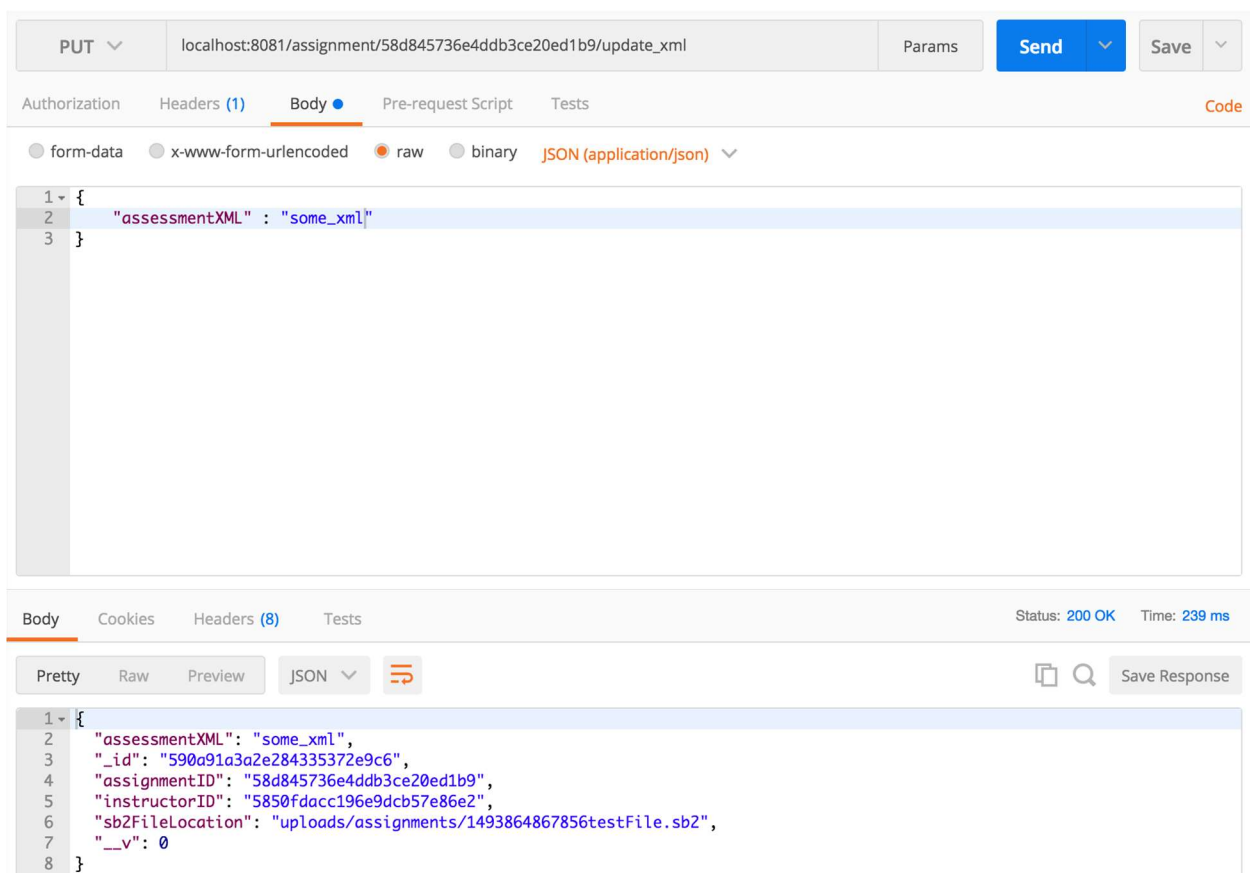


Figure 26. Updating Assessment Response

#### 4.2.2.3 Fetching progress for a Student

This route fetches the progress of a student for a particular assignment. If this is the first time a student is fetching an assignment, a replica of the starter design .sb2 file is created for this student

Resource URL - GET progress/student/:id/assignment/:id

Headers - Content-Type : application/json

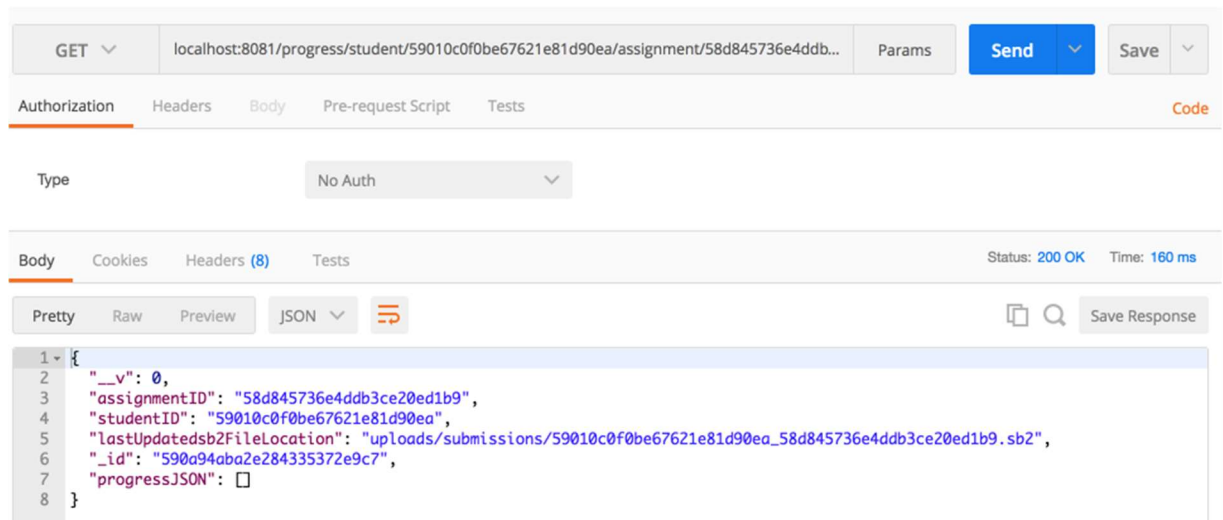


Figure 27. Fetching Progress Response

#### 4.2.2.4 Updating progress of a student

This route is called periodically to update the progress of a student.

Resource URL - PUT progress/student/:id/assignment/:id/updateJSON

Headers - Content-Type : application/json

Body Parameters - jsonString

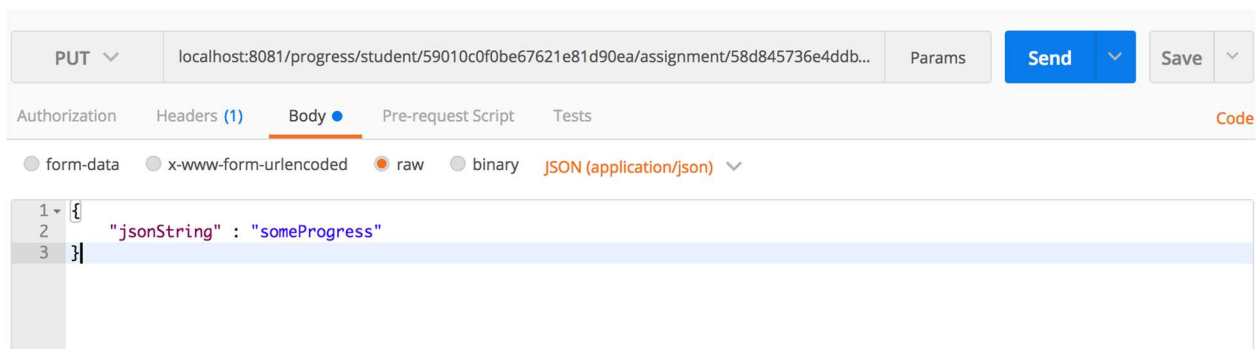


Figure 28. Updating Progress Response

#### 4.2.2.5 Submitting the assignment

This route is called periodically to submit the assignment. Hairball plugin evaluation is run on this .sb2 file submitted.

Resource URL - PUT progress/student/:id/assignment/:id/

Headers - Content-Type : multipart/form-data  
Body Parameters - sb2File

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** localhost:8081/progress/student/59010c0f0be67621e81d90ea/assignment/58d845736e4ddb...
- Headers:** Content-Type: multipart/form-data
- Body:** A multipart/form-data payload with a file named 'testFile.sb2'.
- Response:** A JSON object with the following structure:

```
1 {
2   "message": "Updated",
3   "lastUpdatedsb2FileLocation": "uploads/submissions/1493866083952testFile.sb2",
4   "results": {
5     "Abstraction": 3,
6     "Parallelization": 3,
7     "Logic": 3,
8     "Synchronization": 3,
9     "FlowControl": 3,
10    "UserInteractivity": 2,
11    "DataRepresentation": 3
12  }
13 }
```

Figure 29. Submitting Assignment Response

## 4.3. Recommendation Engine

The recommendation engine works on the principle of collaborative filtering. We have used two types of recommendation - user based and item based .

The user based recommendation engine provides recommendations based on user similarity. In this case, when two students take similar courses ,it is likely that a course taken by only one of the students would also be preferred by the other student. User based recommendation computes a similarity score between two students based on their common courses. The system then analyses the courses taken by the most similar students to the student in question and filters the ones which have not been taken by this student. These courses are then ranked on the basis of similarity score weights and the highest ranked scores are shown in recommendations.

Item based recommendation on the other hand computes similarity scores between items or in this case ,courses. All the courses which have not been taken by a student are analysed and assigned similarity scores. This assigns high scores to courses which are similar to the ones the student is currently taking.

The recommendation scripts have been written in Python and have been deployed in Amazon AWS Lambda functions and are available via API gateways.

The code is available in github.

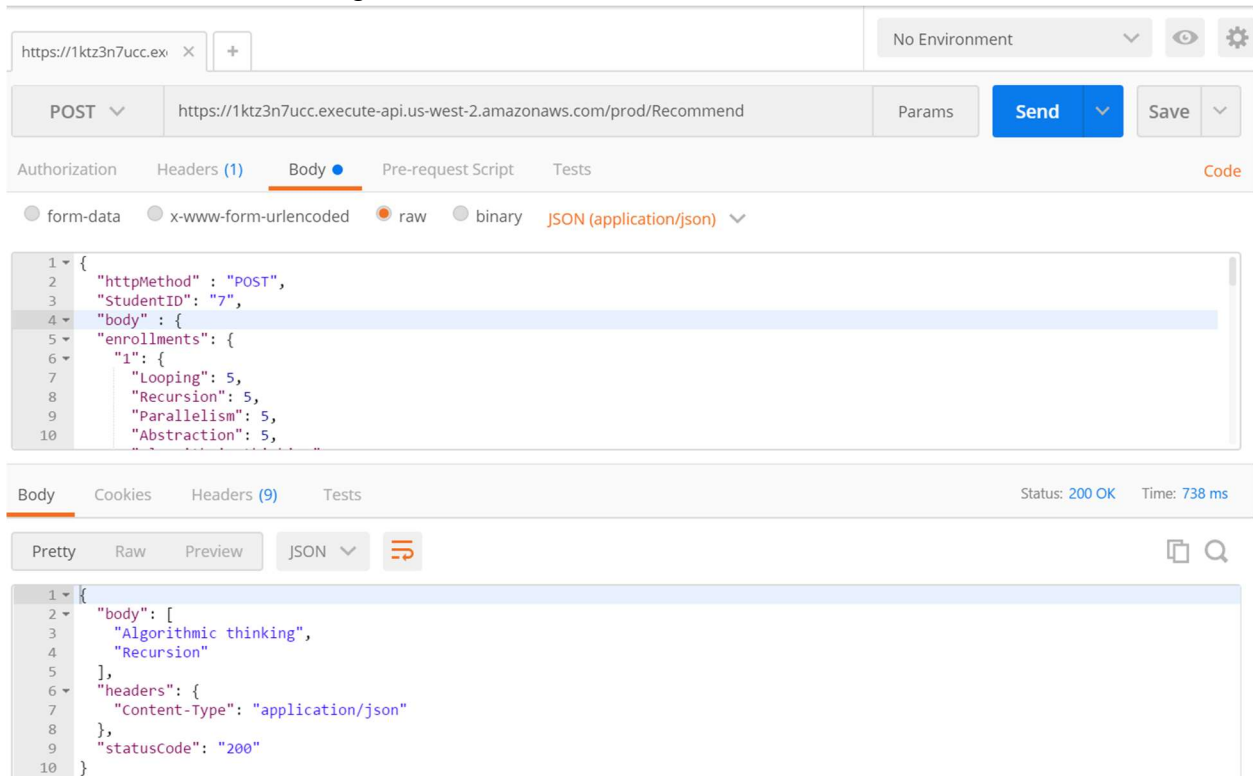


Figure 30. Recommendation Engine Response

As shown in the fig , the API gateway takes a JSON of course enrollments, student ID and returns in the response a list of courses which are to be displayed as recommendations

API Endpoint:

`https://1ktz3n7ucc.execute-api.us-west-2.amazonaws.com/prod/Recommend`

## **5. Conclusion**

The SAGE platform provides a highly innovative and interactive way to teach computational thinking concepts to young students. The entire process of interactive education is based on gamification and game based learning. Hence, it is imperative that the system is encompassed in a platform that is highly intuitive to use both for the students and teachers. The features and design has been done keeping in mind that SAGE could be used as a global platform for education of CT concepts. The recommendations and addition of learning paths/quests make it easy for students to navigate the platform and develop their understanding through the educational games. Further, to make SAGE accessible to everyone, the platform should be intuitive to instructors as well to prepare content ,assignments and assessments. Hence we have integrated all the components and added new features to make the SAGE platform a common web based interface to access all the capabilities of SAGE.

With the massive progress in technology and education, there is a need to evolve the general classroom based study into a more interactive and engrossing environment. The platform of SAGE built upon tools like Scratch make this possible for a global audience. As the research and technology in this field is never constant , we have made the platform as robust as possible to accommodate future changes and development with ease.

## **6. Future Work**

There are plenty of avenues for widespread research and development in Scratch .As we explored and added new features, newer avenues opened up which could be implemented in a proper way to increase the functionality and capabilities of SAGE as well as improve the existing systems

### **6.1. Learning Progressions**

While currently the instructor dashboard has learning paths which allows instructors to create progression models with multiple courses, it can be improved significantly. To hold true to its game based origins, the Learning Paths could be implemented as quests for the students where each quest would have multiple levels in the form of courses where students have to cross one course to access the other in increasing difficulty.



Also, the learning path/quest creation could be handled in an analytical way wherein the quest could start with easier instruction based games but later move on to more construction based games<sup>1</sup>.the instructor could be provided with guidelines or recommendations to choose courses in order to form the learning path which would best benefit the students.

## **6.2. Recommendation Services**

The current recommendation engine works in two ways - user based and course based collaborative filtering . While the current recommendations just provide the courses, these could be coupled with the mastery of CT concepts showing the computational concept associated with each course or the subject. This could again be supplemented with a reward based system wherein the recommendations would come with incentives to complete them

With increase in data , we could also improve the recommendation algorithm to include multiple features and implement machine learning techniques to generate hybrid recommendations comprising common elements from both item and user based recommendations.

## **6.3. Research Dashboard**

As SAGE is a tool for educating children about CT concepts , it would be highly useful in monitoring the statistics of the gameplay of students. The interest level of students along with their response to a type of game would be interesting to observe and trends can be recognized from such data. As the dashboard platform is a common environment for all of SAGE components ,we planned to include a research component that would have all the statistics from gameplay that could be used as a basis for research and improvement.

## **7. Acknowledgements**

We are extremely thankful to Jeff Bender for his constant support and guidance throughout the project. It was with Jeff's help that we managed to scope the project and do proper research to build this dashboard framework. We would also like to take this opportunity to thank Prof Gail Kaiser for her constructive feedback and guidance. We would also like to acknowledge the help of Allison Sawyer and Sambhav Anand who worked on various aspects of SAGE and helped us with integrating the Scratch file. We would also like to thank Jairo, Johann and other members of the team who have always been a great help to us.

## 8. References

- [1] Wing, Jeannette M. "Computational thinking." *Communications of the ACM* 49.3 (2006): 33-35.
- [2] Harteveld, Casper, et al. "A design-focused analysis of games teaching computer science." *Proceedings of Games+ Learning+ Society* 10 (2014).
- [3] Resnick, Mitchel, et al. "Scratch: programming for all." *Communications of the ACM* 52.11 (2009): 60-67.
- [4] Bender, J. (2015). *Developing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula*
- [5] Seiter, Linda, and Brendan Foreman. "Modeling the learning progressions of computational thinking of primary grade students." *Proceedings of the ninth annual international ACM conference on International computing education research*. ACM, 2013.
- [6] *Real Time Assessment of Computational Thinking*, Kyu Han Koh, Ashok Basawapatna, Hilarie Nickerson, Alexander Repenning
- [7] *What Game Are You Playing? Affordances of Tools for Incorporating Game Elements into Classrooms*, Caitlin Holman, Stephanie Wooten, Barry Fishman, University of Michigan
- [8] *GradeCraft*, Aguilar, S., Holman, C., Fishman
- [9] *Stay on the ball! an interaction pattern approach to the engineering of motivation*, Kirstin Kohler, Sabine Niebuhr, Marc Hassenzahl