VISVESVARAYA TECHNOLOGICAL UNIVERSITY,
JNANA SANGAMA, BELGAUM – 590014

A Project Report on

"Learning Analytics for Block Based Programming"

Submitted in partial fulfilment of the requirements for the award of degree of

Information Science & Engineering

Submitted by:

VAISHNAVI S	1PI13IS122
VANDANA S RAO	1PI13IS123
VINITHA RAJ	1PI13IS126

Under the guidance of

Dr Viraj Kumar
Professor,
PESU – CSE

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
PES INSTITUTE OF TECHNOLOGY,
100FT RING ROAD, BSK 3RD STAGE,
BENGALURU - 560085

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELGAUM – 590014

A Project Report on

"Learning Analytics on Block Based Programming"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Engineering

in

Information Science & Engineering

Submitted by:

VAISHNAVI S	1PI13IS122
VANDANA S RAO	1PI13IS123
VINITHA RAJ	1PI13IS126

Under the guidance of

Dr Viraj Kumar

Professor,
Department of CS&E,
PESU

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
PES INSTITUTE OF TECHNOLOGY
100 Feet Ring Road, BSK 3rd Stage, Bengaluru – 560085

January 2017 – May 2017

PES INSTITUTE OF TECHNOLOGY

100 Feet Ring Road, B S K 3rd Stage,

Bengaluru-560085

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"Learning Analytics For Block Based Programming"** carried out by **Vaishnavi S**, bearing USN **1PI13IS122**, **Vandana S Rao**, bearing USN **1PI13IS123**, **Vinitha Raj** , bearing USN **1PI13IS126**, are bonafide students of **PES INSTITUTE OF TECHNOLOGY**, Bangalore, an autonomous institute, under VTU, in partial fulfilment for the award of degree of **BACHELOR OF ENGINEERING IN INFORMATION SCIENCE & ENGINEERING** of **Visvesvaraya Technological University, Belgaum** during the year **2017**. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the above said degree.

Dr Viraj Kumar Professor Department OF CSE PESU	**Dr. Shylaja S S** Professor and Head, Department of ISE PESIT	**Dr. K. S. Sridhar** Principal PESIT

External Viva

Name of the Examiners	**Signature with Date**
1._____	_____
2._____	_____

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

P.E.S. Institute of Technology

Department of Information Science and Engineering

Bengaluru – 560085

DECLARATION

We, **Vaishnavi S, Vandana S Rao and Vinitha Raj**, students of Eighth Semester B.E., in the Department of Information Science and Engineering, **P.E.S. Institute of Technology, Bangalore** declare that the project entitled **"Learning Analytics For Block Based Programming"** has been carried out by us and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Engineering** in **Information Science and Engineering of Visvesvaraya Technological University, Belagavi** during the academic year **2016-17**. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

Vaishnavi S
1PI13IS122 **Signature**

Vandana S Rao:
1PI13IS123: **Signature**

Vinitha Raj:
1PI13IS126: **Signature**

ABSTRACT

The aim of this project is to analyse the development of computational thinking skills of elementary school students during the learning process and provide a visual feedback tool that provides insights into students' level of understanding thereby improving the personalised teaching experience for the both teachers and students. It also helps in providing an enriched experience in adaptive learning environment. Often students who are struggling to understand a topic are not very open in expressing their difficulties, making it hard for the educators to identify this challenge among students. To help students of this category and identify the process of struggle in learning is the main motive of this project.

When students are exposed to new concepts and ideas, there are many ways in which students learn these ideas and concepts. We have divided the learning process in three ways:

1. Students understand the concepts clearly.
2. Students may not understand the concepts clearly and openly ask questions to the educators about their concerns related to the topics.
3. Students may not understand the concepts clearly but are shy and hesitant to ask questions.

We focus on the third set of students and aim to enrich their learning experience. And most importantly, to identify what the struggle in learning looks like from a Learning Analytics perspective.

ACKNOWLEDGEMENT

Firstly, we are grateful to PES University for providing this opportunity to showcase this project.

We wish to express my sincere thanks to Dr Viraj Kumar, Professor, Computer Science and Engineering department for his expert guidance and support throughout till the completion of this project. I am extremely grateful and indebted for his positive encouragement extended to me.

We also thank Dr Srivatsa K for helping us provide necessary contacts for the conduction of a major part of this project.

We would also like to thank the teachers and students of Delhi Public School-North, Bangalore, for facilitating the conduction of the workshop at their premises.

We place on record, my sincere gratitude to Mr. Ashish Surekha, Principal Researcher, ABB Corporate Research Centre(India) for his initial guidance for this project and for providing an enthusiastic motivation to work on this project.

We thank the project co-ordinator Prof. V R Badri Prasad, Associate Professor, Computer Science and Engineering department for his timely assistance regarding the project evaluation and submission.

We place on record, my sincere gratitude to Dr Shylaja S Sharath, Head of the Department, Information Science and Engineering for her constant encouragement.

We also wish to express my sincere thanks to Dr K S Sridhar, Principal of the college, for providing me with all the necessary facilities.

We take this opportunity to record our sincere thanks to all the faculty members for their help and encouragement. We also thank my parents for their unceasing encouragement and support.

We also place on record, my sincere gratitude to one and all who directly or indirectly, have lent their helping hand in this venture.

Table of Contents

LIST OF FIGURES

CHAPTER 01
INTRODUCTION

1.Introduction

In the past few years, technology has become an integral part of people's lives, influencing things such as communication, and most importantly the methods of teaching and learning.

An important goal of education is that in addition to content, students develop cognitive process abilities. Learners are expected to use these abilities for sense-making and problem-solving across domains in educational settings and in the workplace. The availability, scalability, and flexibility of e-learning platforms, eliminates time and space restrictions, and provides an appealing and personalized learning opportunity. Adaptive e-learning provides an individualized, engaging and responsive learning environment which is difficult to accomplish with traditional learning techniques. Educators are constantly finding better ways for students to learn, and new technologies such as online educational games are becoming a part of this process. Our research project focuses on the following aspects:

- Assessment of the computational thinking among elementary school students
- Technology enhanced learning environments for such assessment
- Automated identification of students' progress while learning

Computational Thinking (CT) is a problem-solving process that includes several characteristics and dispositions. CT is essential to the development of computer applications, but it can also be used to support problem solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom. The set of audiences to this research project are elementary school students because it is found to believe that what students learn at this age tend to have a profound impact on their mind-set in future. And at this age the students start to develop their computational thinking skills.

'Hour of Code' and Scratch Application are few of those platforms which aim at developing computational thinking skills along with various logical and analytical abilities like problem solving, logical reasoning and pattern recognition, to all age

groups. They are based on block based programming. Block based programming is a technique of teaching introductory programming skills to students by drag and drop “blocks” of instructions onto a workspace. We have used Scratch as the environment for learning in this project as it happens to be the most popular pedagogical tool for block based programming by far. Using Scratch students code their own interactive stories, animations, and games. In the process, they learn to think creatively, reason systematically, and work collaboratively — essential skills for everyone in today’s society. A key goal of Scratch is to introduce programming to those with no previous programming experience. This goal drove many aspects of our project design. Hour of Code is a worldwide initiative by the non-profit organisation Code.org to teach computer science. They provide many tutorials that are based not only on block based programming but also on other programming styles and languages to imbibe problem solving techniques in students of all age group.

To reach full potential of technology enhanced learning, adaptive e-learning systems could trace learners’ activity, monitor their individual characteristics, and generate timely adaptive interventions according to effective pedagogical strategies. By using this trace data and presenting it in a meaningful format, the pedagogical tool can help educators to understand and assess the student's learning process better. This would lead to a better personalised teaching environment for those students who are struggling to understand a certain topic in a way improving the all-round development of classrooms.

After having referred to various tutorials on Hour of Code, we designed a problem statement to conduct workshops to observe how the age group react when they face difficulties with the problem and what difficulties they face. Subsequently we monitor the real time interaction of the students with the learning tool, use this data to mine and give valuable feedbacks on the student's progress.

1.1 Problem Definition

To investigate and build learning analytics based solutions for block-based programming to improve pedagogical tools and techniques. When students are exposed to new concepts and ideas, there are many ways in which students learn these ideas and concepts. We have divided the learning process in three ways:

1. Students understand the concepts clearly.
2. Students may not understand the concepts clearly and openly ask questions to the educators about their concerns related to the topics.
3. Students may not understand the concepts clearly but are shy and hesitant to ask questions.

Our main area of focus is towards the third set of students and improve their learning experience. And most importantly to identify what this struggle looks like from Learning Analytics perspective.

The main objective of this project is to identify what struggle in learning process looks like and to come up with ways to identify it. To help students learn better and teachers teach better with improved personalised teaching approaches.

1.2 Generic Proposed Solution

The proposed approach involves identifying the right version of the open source tool, Scratch, to modify to collect data. An appropriate problem statement needs to be designed for the students to work on. Simultaneously, the data that must be gathered while students are working on the tool should be identified. Also the behaviour of the students that had to be observed during the workshop should be listed out. This project demands a whole lot of field work in gathering sets of students to play with the tool to collect the necessary data. After fixing a time and date for conducting the workshop, we need to physically set up the computer laboratories of the venue where we plan to do the workshop. A feedback form must be designed to gather the user experience data.

Once all the data, log data, observation data and feedback data, is available we plan to understand the student's progress at specific timestamp to know his deviation from the right answer and correspondingly respond to factors which impede a student's learning process.

1.3 Acknowledgement

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PES INSTITUTE OF TECHNOLOGY

(An Autonomous Institute under VTU, Belgaum)

100 Feet Ring Road, BSK- III Stage, Bangalore – 560 085

Project ID: PW119

Project Title: Learning Analytics for Block Based Programming (LEAPER)

Project Team: 1PI13IS122 Vaishnavi S

1PI13IS123 Vandana S Rao

1PI13IS126 Vinitha Raj

This project report was submitted for review on _____. I acknowledge that the project team has implemented all recommended changes in the project report.

Guide signature with date:

Guide Name:

CHAPTER 02
LITERATURE SURVEY

2.Literature Survey

Lynne et al. described three methods he used to make ethnographic observations of 30 pre-school children: audio-recording of naturally occurring talk, participant observation, and key informant diaries [1]. One of the methods used was to silently observe the participant without the knowledge of the participant. Here the researcher remains away from the young children, but carefully observes their behaviours of interest, which is particularly suitable for data collection. They recorded field notes about the observations they made about the description of events, settings and people with notations of time and duration; techniques or approaches used and the reaction impressions of the observer.

V. Chaudhary et al. describe their experimental study aimed at investigating the application and effectiveness of Lego Mindstorms EV3 for teaching computational thinking, problem solving, programming, team-work and project management to elementary level kids [5]. Their research framework and methodology comprised of various steps including surveying students about their expectations and their prior knowledge, monitoring their behaviour when they understand a concept or face difficulty and evaluating the results of their summer camp. Their experience report on the summer camp they conducted, showed the effectiveness of a short-term course.

C. Williams et al. describe their study which involved the design and implementation of a computer game to meet the goal of not only introducing elementary students to the fundamentals of computer programming, but also helping them explore more complex concepts in an engaging way [6]. By instructing a virtual robot to complete obstacle courses, students became familiar with core programming concepts which included algorithms, repetition, conditional logic, debugging, and optimization.
They conducted several interactive sessions with a group of elementary school students in order to evaluate the effectiveness of the game. Their results showed the game effectively familiarized students with both computer language constructs and the essentials of algorithmic thinking.

Brennan K. et al. described the use of artifact-based interviews to study the development of computational thinking in interactive media design[2].
Their study focuses on computational thinking and the ways in which design-based learning activities help in the development of computational thinking in young people. Their paper states that being incremental and iterative, testing and debugging, reusing and remixing, and abstraction and modularization are various practices involved in computational thinking. They state suggestions for assessing the learning that takes place when young people engage in learning programming concepts.

S. A. Nikou et al described their study which examined the nature of student motivation during The Hour of Code activities across two different contexts: in a high school class and in a first-year University course [3].
The theoretical framework used by the study was Self-Determination Theory of Motivation, which distinguished four types of motivation, namely Intrinsic Motivation, Identified Regulation, External Regulation and Amotivation. The students were asked to report their four components of motivation during The Hour of Code activity using 7-point Likert scales. The overall motivation was calculated using a Self Determining Index (SDI), which is an approach to combine these subscales and is reported as a sum of the individual motivation subscales appropriately weighted.
Both groups rated their level of self-determined type of motivation pretty high. Intrinsic motivation and identified regulation were dominant in both cases while external regulation and amotivation also existed. High school students found the activities more intrinsically motivated. This was because the level of the educational activities is better suited to their lower age level.

V. Kumar et al presents an ethnographic study of students and Mathematics tutors to investigate requirements for designing applications which are beneficial for students who lack the access to tutors.[4]

The student - tutor interactions proved to be quite insightful in revealing the typical behaviour of students to skip video tutorials and directly solve the problems as they believe that this is an efficient way to use their time. Moreover, many students often overlook the basics and hence don't have a good understanding of those concepts. The chances of going wrong in problems that require an understanding of the basics is very high.

CHAPTER 03
SYSTEM REQUIREMENTS SPECIFICATION

3.System Requirement Specifications

3.1 High level block diagram of the solution

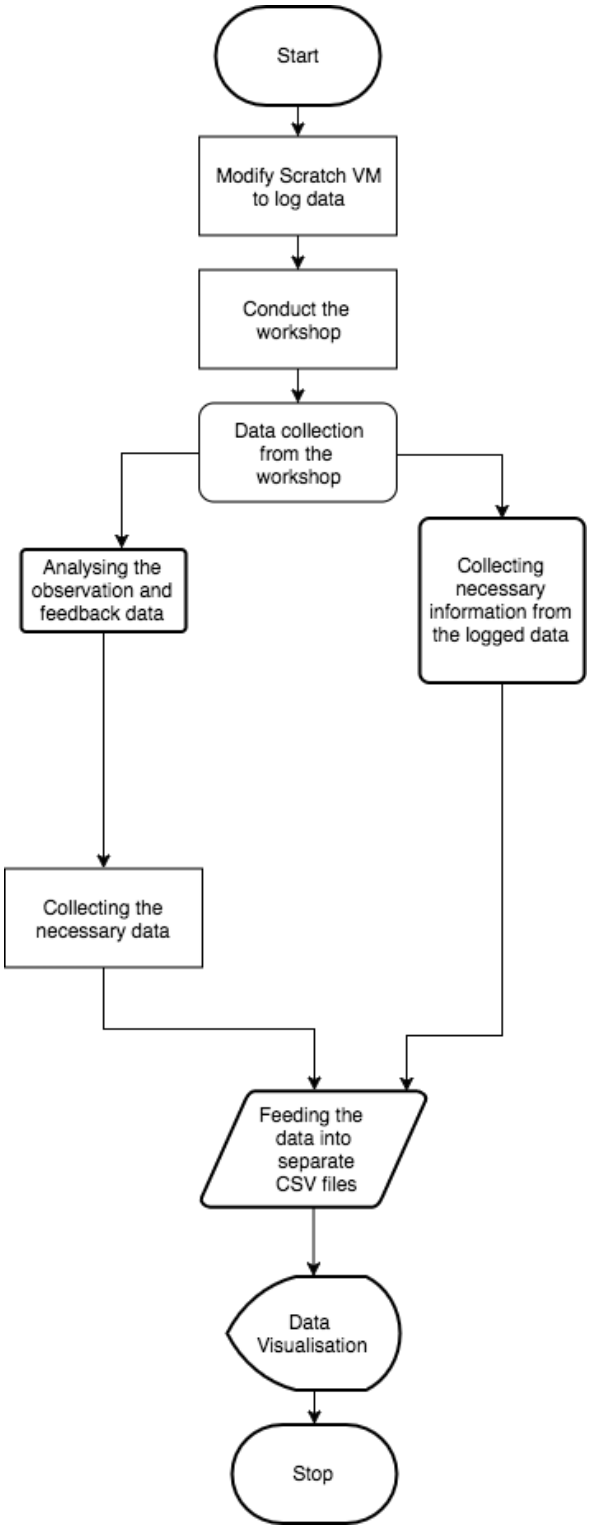


Fig 3.1 – High Level Block Diagram

Our project can be divided into five distinguishing modules or steps as shown in the above diagram.

The first step involved a thorough study of the way the Scratch VM tool works in order to modify it to the requirements of our project to log the user interaction data as the student plays with it. We had to modify certain files in the source code of Scratch VM and also write a code to monitor the activity of the student when the student plays on it.

The next step involved designing a problem statement for the students to play around with during the workshop and conducting the workshop. The problem statement had to be designed such that the students find it interesting and also help the students from an educational perspective. Then schools were contacted to fix dates for conducting workshops.

In the third step data was collected while the student worked on solving the problems on Scratch VM. This data was collected on the local system of every student. The bulk of data collected had to be refined and only the information necessary to us was to be taken from these log file.

In the next step, we had to accumulate all the information collected into a suitable format required as per the design of the user interface.

In the final step, a data visualization tool had to be developed to represent the collected data. We developed this using a CSS template. For every student we had to display the graph of each of the student's performance throughout the duration of the workshop.

3.2 Environment used in the project

The details of the specific environment used for this project are: Development environment for developing the project, a Test environment to test the features of the project, a QA environment for assuring quality when the project is deployed and a Production environment where the project was run.

The specifications of the environments used are as follows:

Development environment: 64-bit Ubuntu Operating System was used to develop the

codebase for the logging tool that we've used in the project. First the Scratch VM had to be installed on the system along with nodejs. Sublime Editor was used for coding purpose and run manually via terminal.

Test Environment: Testing was also done on the same environment as Development but without internet connection. Testing basically involved ensuring that the Scratch VM was running properly without throwing any errors. Then we had to ensure that the tool ran without internet connectivity. And also logged the interaction data while the tool was being used.

QA environment: This environment was a 64-bit Windows 8 machine without internet connectivity. All of the schools that we contacted had windows machines deployed in their computer labs. Hence, we had to test our tools on windows machine to make progress in the project.

Production environment: We used 32-bit machines with latest Windows 7 or Windows 8 Operating System installed on them. Nodejs had to be installed on every system. Basically, these were the systems in the computer laboratory of the schools that we had to conduct the workshop in. A few of these systems had internet connection and few of them did not. However, we had designed our tool to work either ways.

The specific kinds of softwares and tools in terms of hardware and software are explained in detail in the following section.

3.2.1 Hardware required

1. A computer having the following hardware configurations:
    a. RAM - 4 GB
    b. Hard Drive - 320 GB 5400 RPM hard drive
    c. Wireless (if laptops) - 802.11g/n (WPA2 support required)
    d. Monitor - 19" LCD - desktop only
    e. Operating System - Windows XP, 7, 8
    f. Backup Device - External hard drive, USB Flash Drive
2. Minimum internet requirements:
    a. Bandwidth/Speed: greater or equal to 4 Mbps on every system

3.2.2 Software required

Chrome 35 or later, Mozilla Firefox 31 or later, Internet Explorer 9 or later, or Microsoft Edge. You also need Adobe Flash Player 10.2 or later installed (which is only available on Windows 2000 or later or Mac OS X 10.4 or later). These software are used for experiencing the developed project tool. The main software required is the browser since that is used for working with the Scratch application.

Nodejs must be installed on the computer system since this starts the server where the application is running. Nodejs also provides the connectivity from the application to the logging engine that logs the user interaction data from the application.

3.2.3 Requirements for the project

Functional Requirements

F1 - Product logger and Process logger

We need to log the final code written by students for a given problem statement. Students are made to solve a problem on the offline scratch application and their final code has to be logged for further analysis. Also, user interaction data is logged. This data includes every interaction or every step taken by the student in order to accomplish the given task. This is logged as JSON objects. This data is collected when the students are working on the scratch offline application. Analysing the JSON string, extract the data regarding the user's interaction.

F2 - Perform analytics

On the data collected using either the process or product logger, machine learning algorithms are applied to identify the performance levels of the students. Information regarding the student's progress through the time he finds the solution, his deviation from the right answer at various intervals are planned to be calculated through this functionality.

F3 - Calculate edit distance

Once the data is collected, the number of various correct solutions needs to be identified and kept as a reference to calculate edit distance between the student's answers and the right answers. The deviation of the student's answers from the right solution is calculated and used for other functionalities.

F4 - Generation of Dashboard

After F3 is performed, the result is displayed in the form of dashboard which is given to the teachers as a feedback about the students performance. This is planned to contain graph representation of the student representing the performance rise or drop at different timestamps with respect to the right solution.

F5 - Advance analysis on dashboard

We aim at providing advanced functionalities which provides feedback to both students and teachers. Ex: If there is any concept in which most of the students fail to perform well then we provide a feedback to teacher to improve their teaching skills in a particular concept. If there is any concept a particular student is failing to understand then we provide feedback to teacher to facilitate personalised teaching to such students.

Non-Functional Requirements

NF1 - Scalability

Additional functionalities can be added to the product based on the requirements of the end user without having to change it entirely.

NF2 - Data Integrity

The data collected from students is kept confidential and the result about the student's performance is provided only to the teacher to enhance a student's skills.

NF3 - Usability

Feedback provided to the teachers has an interface which can be very easily understood and also additional analytics on the data is provided using graphs or dashboard.

UI Requirements

U1 - Dashboard

Represents the student's performance with respect to every concept the student was tested on. This provides the teacher with detailed information about student's performance.

3.2.4 Constraints and dependencies

The targeted audience is elementary school students whose age group is between 9 and 13 years. Since the learning process is quite effective among students who fall under this age group, we have chosen students of this age group as an ideal group for making observations related to our project. Thereby giving rise to a constraint of limited audience.

The hull of the project depends on the data that we collect from the students during the workshop. With project being held at a very crucial period between January and April, it was very difficult to arrange for sets of students for the workshop. Even if the students were available, the necessary resources availability became a constraint. We had to let go of the opportunity of collecting data from 100 students initially because of the scarcity in resources and also the resources failing to meet the our hardware and software specifications.

We are very thankful to Dr Srivatsa for co-ordinating with Delhi Public School - North, Bangalore to arrange for this workshop. The school was ready to provide us with 200 students. With the students and resources available, there was a constraint of getting enough volunteers to observe the students during the workshop due to which we were not able to tag all 200 students' data. However, we managed to tag 30-40% of the data that was collected.

3.2.5 Assumptions

There are two kinds of data that are collected during the workshop. Tagged data and untagged data. Tagged data consists of those student's data for whom observations could be made by the observer during the workshop. Untagged data consists of those student's data for whom observations could not be made during the workshop.

For understanding the effectiveness of learning via visual programming, we had to assume that the students did not have any prior knowledge on working with Scratch programming. And the observers had to make their observations on students based on this assumption that the students had least experience on working with the Scratch Tool.

The problem statements were designed in such a way that it required the understanding of basic introductory computer programming skills like straight line coding, simple loops and conditionals. We had to make an assumption that the students are already aware of these concepts and their usage in the world of computer programming to solve the problem posed to them during the workshop. The reason why this assumption is made is because the real time data is collected while the students are solving the problem. And if the students don't have prior knowledge of these concepts, the entire point of the workshop becomes null. More so, the targeted audience being elementary school students, it is safe to assume that they have the knowledge of these basic concepts of computer programming.

3.2.6 Use case diagrams for the requirements

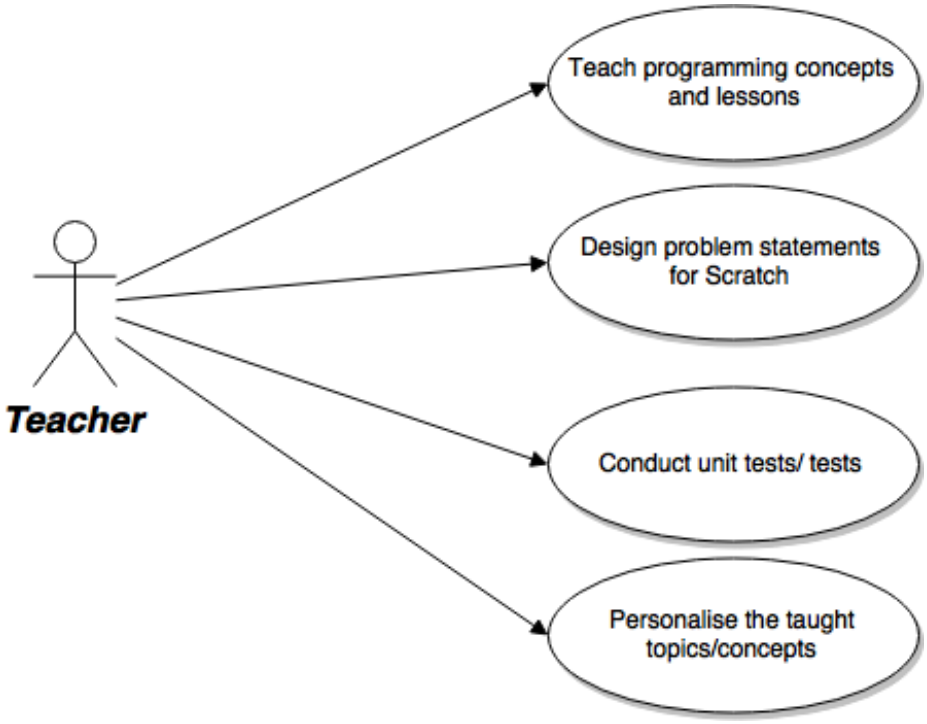


Fig 3.2.1 Use case diagram for Teachers

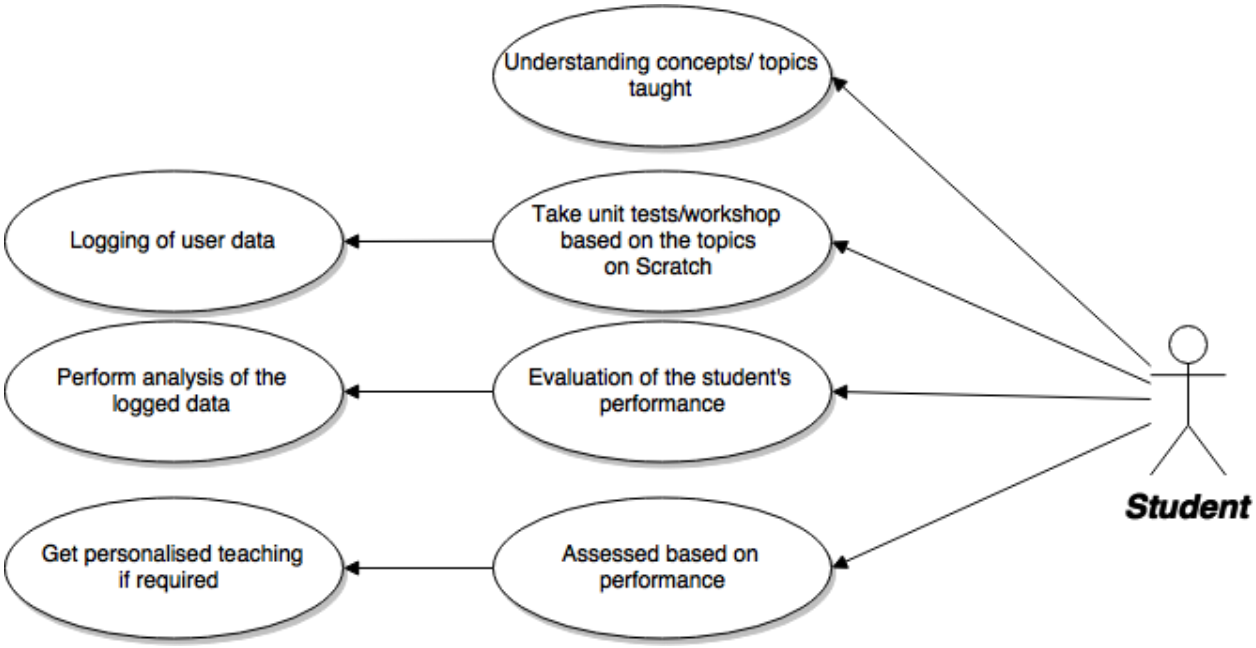


Fig 3.2.2 Use case diagram for Students

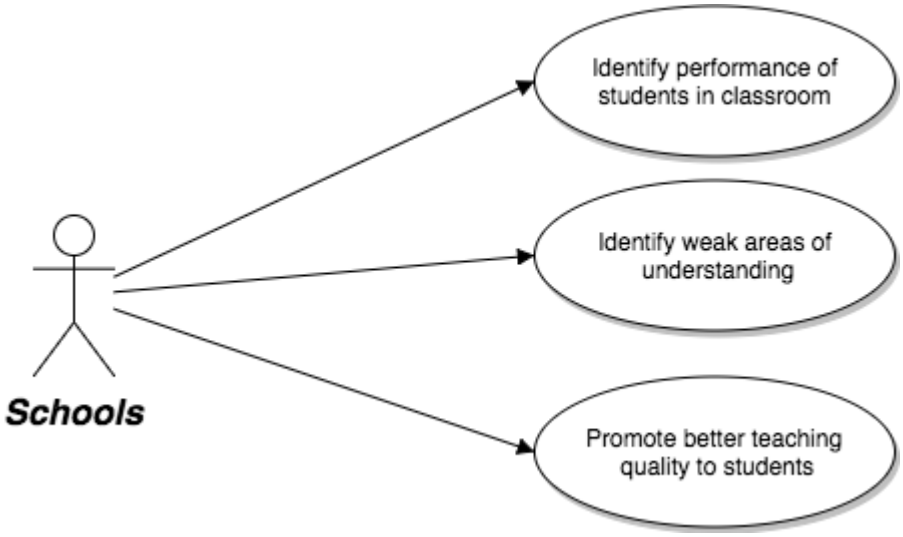


Fig 3.2.3 Use case diagram for Schools

Fig 3.2.4 Overall use case diagram

3.2.7 Requirement traceability matrix

3.8.1 Project Requirement Document (PRD)

PR#	Module Name	Applicable roles	Description
PR1	Hour of Code Workshop	Student, Observer	Student:Get used to Block Based paradigm application Observer:Observe how the student interacts with the problem manually.
PR2	Designing Scratch problem	Developer	Design a problem statement which includes basic type problem.
PR3	Scratch	Student	Student:Work with Scratch having previous hour of code familiarity
PR4	Logging Engine	Student, Developer	Student:Interacts with the system and logger logs the user data. Developer:Get the student interaction data using logging engine.
PR5	Analysis Dashboard	Teacher,Student	Teacher:Get the feedback about the students' performance Student:get the results about their performance.

3.8.2 Technical requirement Document(TRD):

TestCase ID	PR#	TR#	TestCase	TestData	Expected Result
1	PR1	TR1	Verify the student's first time introduction to block based programming	Observation forms, Feedback forms.	Positive feedback from students
2	PR2	TR2	Verify the problem such that difficulty level	Scratch problem developed	Basic concepts covered

			includes only basic programming concepts.		
3	PR3	TR3	Verify student is comfortable with the basic concept problems.	The interaction data collected while students interact	Log file of interaction data.
4	PR4	TR4	Verify the data is logged	JSON object	Log file is saved into the database.
5	PR5	TR5	Data mining on the log files	Log file	Mined results
6	PR5	TR6	Show the results to the user.	Dashboard	Get feedback about the student

3.9 Requirement Traceability Matrix:

PR#	TR#	TestCase ID
PR1	TR1	1
PR2	TR2	2
PR3	TR3	3
PR4	TR4	4
PR5	TR5,TR6	5,6

CHAPTER 04
SCHEDULE

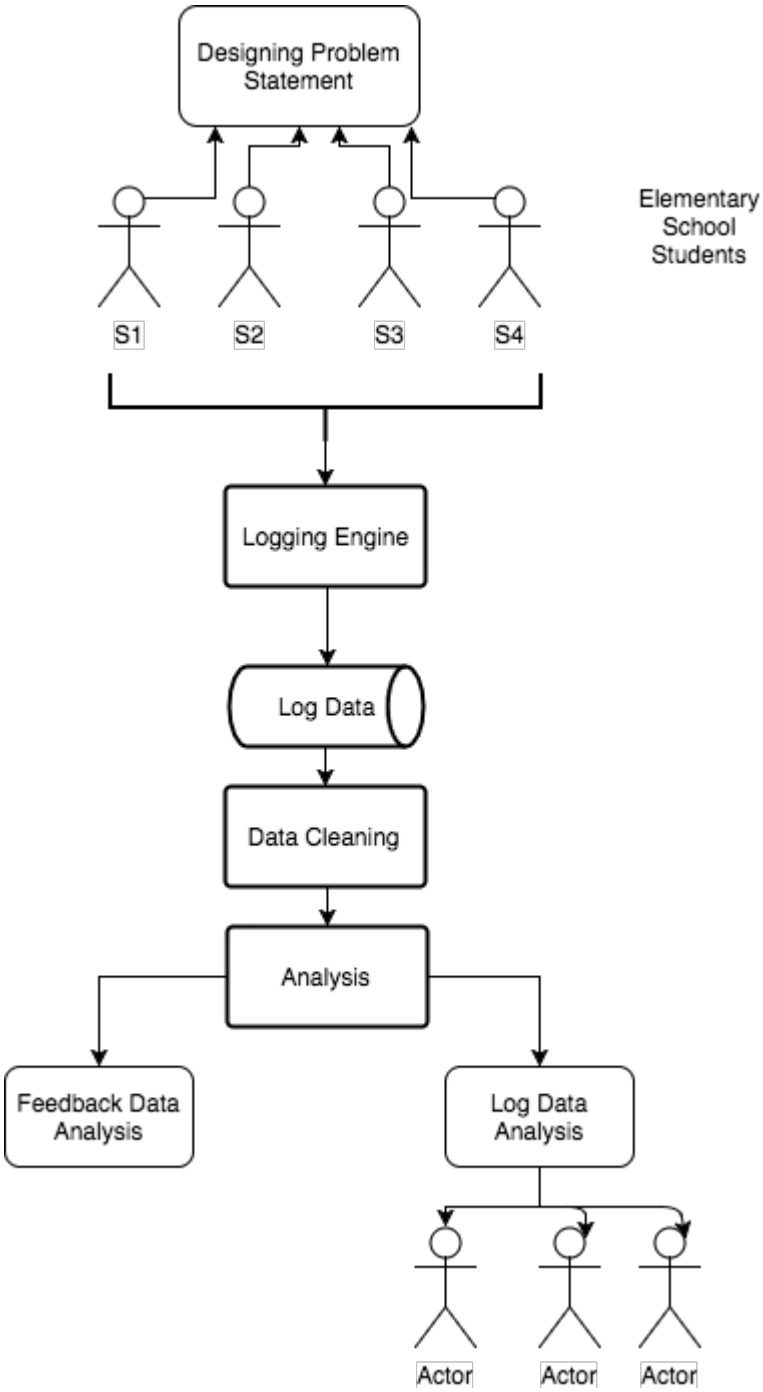
4. Schedule

Jan 2-Jan 6	Literature Survey
Jan 9-Jan 30	Designing the observation and feedback forms
Jan 30-Feb 3	Conduction of Workshop 1
Feb 6-Feb 10	Designing problem statement for workshop
Feb 13-Feb 24	Logging engine development
Feb 27-Mar 3	Conduction of Workshop 2
Mar 3-Mar 17	Cleaning the Logged data and matching the observation forms and feedback forms with the data
Mar 20-Mar 24	Extraction of opcodes from the JSON data
Mar 27-Mar 31	Extraction of opcodes from the JSON data
Apr 3- Apr 7	Calculation of Tree Edit distance scores
Apr 10- Apr 14	Development of UI - Dashboard
Apr 17- Apr 21	Testing

CHAPTER 05
SYSTEM DESIGN

5. System Design or High level design

5.1 Architectural diagram



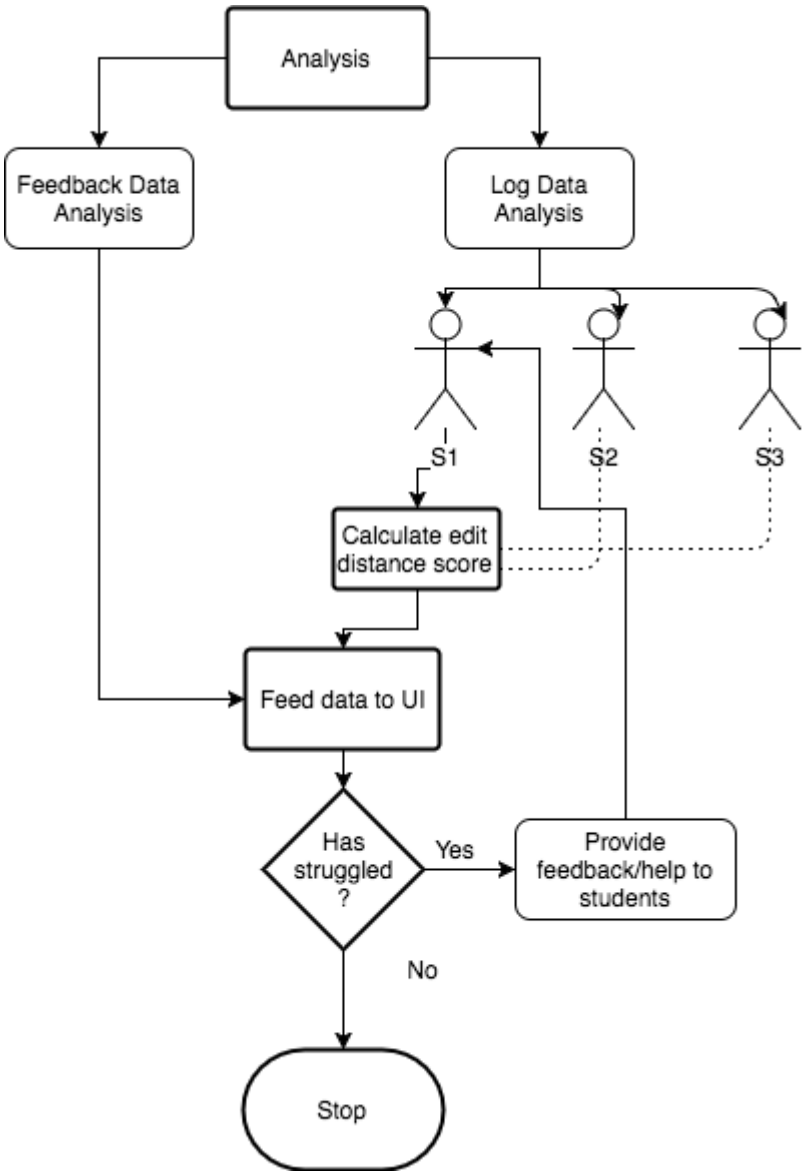


Fig 5.1 Architecture diagram

5.2 Sequence diagram or Interaction diagram

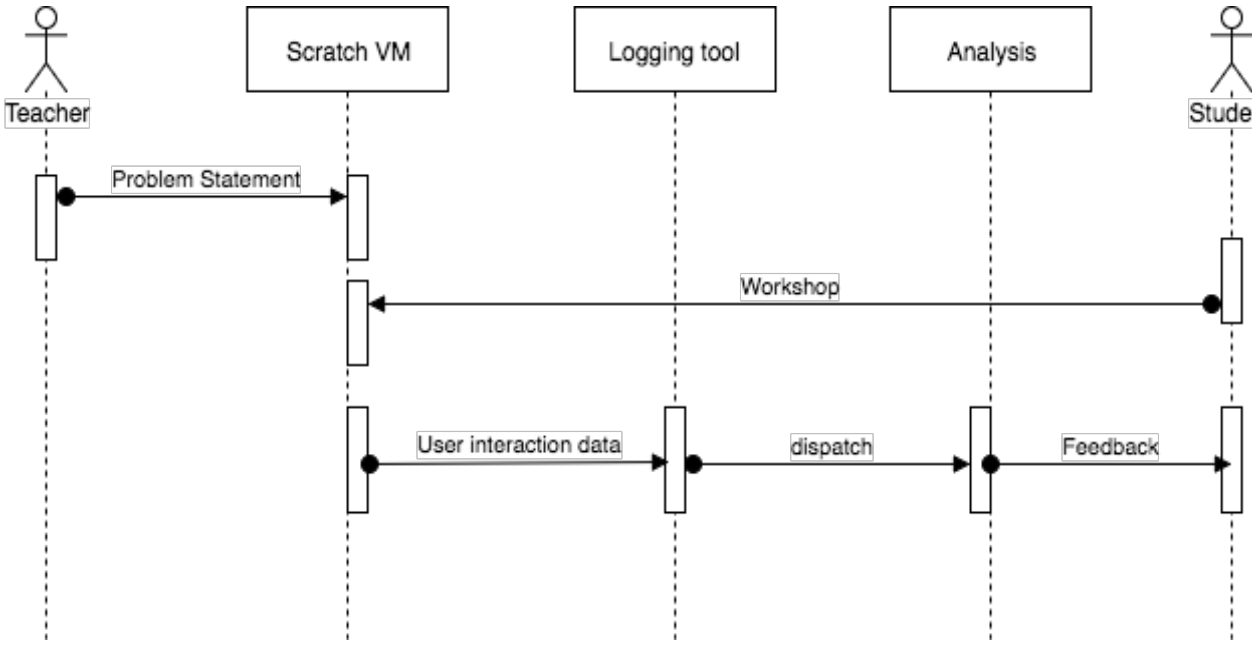


Fig 5.2 Sequence Diagram

The interaction diagram for the use case involving school is the same as Fig 5.1 with the difference that the feedback provided to the schools will be the cumulative performance of the students performed in the workshop/classrooms. Thus, helping the school understand where their students stand and what steps could be taken to improve the quality of learning in their curricular environment.

5.3 UI Design

The user interface comprises of a dashboard that provides the data visualisation mainly for two purposes.

- Provides the overall feedback of the workshop or classroom experience as stated by the students.
- Provides the student specific graph visual with respect the particular student.

Feedback forms were given to all the students who attended the workshop. This feedback form gave the observers/educators an idea about how the student found the workshop, whether the student had previous experience on working with block based programming etc. Using this feedback data we plotted a graph against the difficulty level the students faced and students.

We also provided student specific graphs related to the student's performance. This graph was based on the tree edit distance scores of the particular student against the right solution. The graph had scores of the students against timestamp. This graph helped in visualising how far the students' answer was from the right answer at particular time intervals.

5.4 Updated RTM

5.4.1 Project Requirement Document (PRD)

PR#	Module Name	Applicable roles	Description
PR1	Designing Scratch problem	Developer, Teacher	Developer: Design a problem statement which includes basic programming concepts. Teacher: Corresponding to the curriculum design a problem statement in scratch appropriate for the students to work on.
PR2	Scratch	Student	Student: Work on Scratch to solve the given problem statement.
PR3	Logging Engine	Student, Developer	Student: Interacts with the scratch application and logger logs the user data. Developer: Get the student interaction data using logging engine.
PR4	Analysis Dashboard	Teacher, Student	Teacher: Get the feedback about the students' performance Student: Get the results about their performance.

5.4.2 Technical requirement Document(TRD):

TestCase ID	PR#	TR#	TestCase	TestData	Expected Result
1	PR1	TR1	Verify the student's interaction with scratch application	Observation forms, Feedback forms.	Positive feedback from students
2	PR2	TR2	Verify the problem such that difficulty level includes only basic programming concepts.	Scratch project developed.	Basic concepts covered
3	PR2	TR3	Ensure that the problem statement loads without internet connection	JSON objects in a text file of the scratch problem statement	Loading of Scratch VM with the problem statement without internet
4	PR3	TR4	Verify student is comfortable with the basic concept problems.	The interaction data collected while students interact	Log file of interaction data.
5	PR4	TR5	Verify the data is logged	JSON object	Log file is saved into the database.
6	PR5	TR6	Data mining on the log files	Log file	Mined results

7	PR5	TR7	Show the results to the user.	Dashboard	Get feedback about the student

5.4.3 Requirement Traceability Matrix:

PR#	TR#	TestCase ID
PR1	TR1	1
PR2	TR2, TR3	2,3
PR3	TR4	4
PR4	TR5	5
PR5	TR5, TR6	6,7

CHAPTER 06
DESIGN

6. Design (Detailed design)

6.1 Data flow diagram

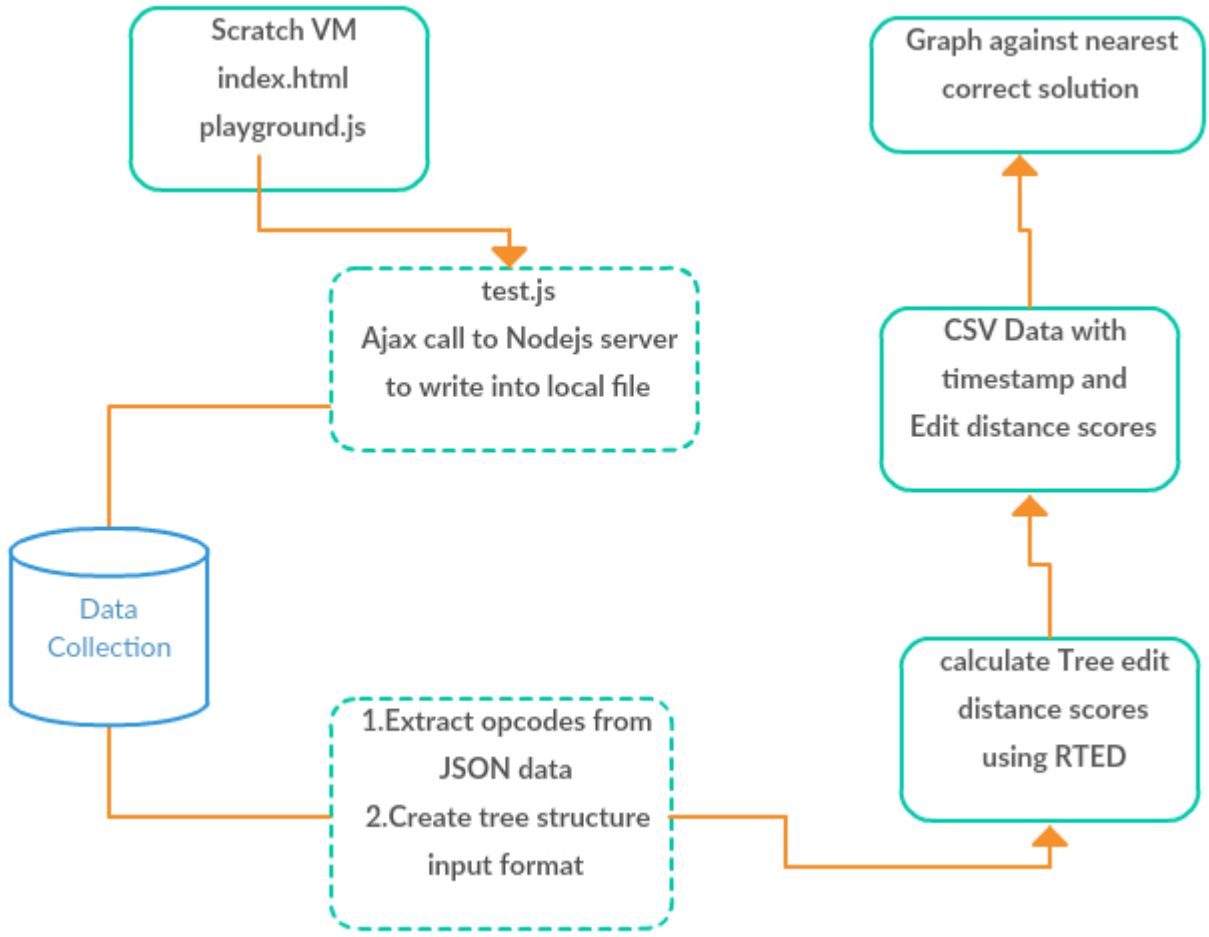


Fig 6.1 Data Flow in Leaper

6.2 Updated RTM

6.2.1 Project Requirement Document (PRD)

PR#	Module Name	Applicable roles	Description
PR1	Designing Scratch problem	Developer, Teacher	Developer: Design a problem statement which includes basic programming concepts. Teacher: Corresponding to the curriculum design a problem statement in scratch appropriate for the students to work on.
PR2	Scratch	Student	Student: Work on Scratch to solve the given problem statement.
PR3	Logging Engine	Student, Developer	Student: Interacts with the scratch application and logger logs the user data. Developer: Get the student interaction data using logging engine.
PR4	Analysis Dashboard	Teacher, Student	Teacher: Get the feedback about the students' performance Student: Get the results about their performance.

6.2.2 Technical requirement Document(TRD):

TestCase ID	PR#	TR#	TestCase	TestData	Expected Result
1	PR1	TR1	Verify the student's interaction with scratch application	Observation forms, Feedback forms.	Positive feedback from students
2	PR2	TR2	Verify the problem such that difficulty level includes only basic programming concepts.	Scratch project developed.	Basic concepts covered
3	PR2	TR3	Ensure that the problem statement loads without internet connection	JSON objects in a text file of the scratch problem statement	Loading of Scratch VM with the problem statement without internet
4	PR3	TR4	Verify student is comfortable with the basic concept problems.	The interaction data collected while students interact	Log file of interaction data.
5	PR4	TR5	Verify the data is logged	JSON object	Log file is saved into the database.
6	PR5	TR6	Data mining on the log files	Log file	Mined results

7	PR5	TR7	Show the results to the user.	Dashboard	Get feedback about the student

6.2.3 Requirement Traceability Matrix:

PR#	TR#	TestCase ID
PR1	TR1	1
PR2	TR2, TR3	2,3
PR3	TR4	4
PR4	TR5	5
PR5	TR5, TR6	6,7

CHAPTER 07
IMPLEMENTATION

7. Implementation

7.1 Pseudocode/algorithm

Scratch blocks are built on Blockly language. The Blockly blocks are internally represented as a set of compound statements in either Javascript, Python or other languages(Lua, Dart). The Scratch VM (Version 3.0) is an improvement over Scratch 2.0, which maintains state machine of Scratch code. An Abstract Syntax Tree is built internally when blocks are moved onto the workspace in the form of JSON objects.

That is the block definition and its information are defined in the form of JSON objects when the blocks are moved onto the workspace area. The block definitions at various timestamps gives the status of the block structure on the workspace. This information is useful in knowing what kinds of blocks are being used at which point of time into the workshop that we plan to conduct. We need to log this information that consists the JSON objects of the blocks present on the workspace. We then create an abstract syntax tree out of it for every timestamp when we collect data.

This JSON data has information about the hierarchy of the block structure along with the generate information regarding the blocks present on the workspace. The hierarchy consists of id, parent block, child blocks, inputs, fields and name of the block:

- Id - This corresponds to the id of the current block.
- Parent block - Id of the parent Node Block
- Child blocks - The child blocks are identified by the keys “next”.
- Input fields - These fields are the actual input alphanumeric values given to the blocks.
- Name of the block - The name of the block is also known as the opcode of the block. Specifies what kind of block it is.

Using these data from the log, we rebuild the block structure at every timestamp to view the progress of the student’s solution. Rebuilding consists of cleaning the collected data and extracting the opcodes of the blocks at every timestamp. Once this is done, we need to identify the number of correct solutions to the levels of the problem. For a given problem there could be many right solutions. Hence to assess the student’s performance it is necessary to identify all the possible right solutions to the problem.

Once we have the right solutions in hand, we divide these solutions into types and then start comparing the all the student's progress with these correct solutions. The number of steps he must take to reach the final solution is calculated using Tree edit distance algorithm.

We use the [8] Robust Tree Edit Distance (RTED) algorithm to calculate the tree edit distance between the trees generated out of the opcodes. The RTED computes the optimal strategy by performing an exhaustive search in the space of all possible strategies. The optimal strategy is computed in quadratic time and space; thus, the strategy computation does not increase the complexity of the tree edit distance algorithm, which is at least $O(n^2)$. The RTED calculates the scores for every student with respect to the correct solution.

With the scores generated, a graph is plotted to show the distance at every timestamp up till the final solution is reached or till the end of time limit of the workshop. Since there can be different solutions, each student's score will be with respect to the closest type of right solution the student has tried to implement. By doing a thorough study of the final graph, we can identify many aspects of the student's learning process and thought process. In this way, we are also able to decide where the student struggled and how far he/she is from the correct solution. For a few students we can also map to identify whether the student is open about the doubts or concerns he/she has. There are cases where the student struggled but got the right solution. Plotting the graph with the student's progress with various correct solutions provides the viewer an overview of where the student struggled and how he/she deviates from the right or wrong solution.

The algorithm of this implementation can be briefed as follows:

Step 1: Study the working of Scratch application

Step 2: Modify the source code of the Scratch VM to log the user interaction data

Step 3: Design a problem statement for the workshop

Step 4: Gather students to conduct the workshop

Step 5: Setup the application with the problem statement for the workshop

Step 6: Conduct the workshop and log the data

Step 7: Observe the students manually to identify if they are struggling to get the answer or not by making certain observations

Step 8: Collect the user interaction data and clean the data

Step 9: Identify the various correct solutions for the problem

Step 10: Run tree edit distance module for each of the student's data to get the scores at various time stamps

Step 11: With the scores generated for every student, plot the graph indicating the performance of the students throughout the duration of the workshop.

Step 12: Feed this to the Data visualisation tool for the UI that will be given to the students and teachers.

7.2 Codebase structure

Scratch VM uses Nodejs to render the Version 3.0 state machine of scratch. We run the development server through npm. When we start the server, the file playground.js gets executed and serves the content through the code written there. Once the server is started, the content gets served on localhost with port number 8077. Simultaneously we need to run the test.js file to start logging what goes on in the workspace to a file in json format. These two files play a very important role during the conduction of the workshop. Without them, collection of data cannot happen.

After collecting the data, we need to start to clean the data and start making sense out of the collected data for which we use readFile.js. readFile.js basically converts the collected data and appends the details of the corresponding block data at respective timestamp

Now that we have the partially clean data, we extract the required information from this text file generated by the readFile.js and create respective text files.

Now we have three set of files. One is the partially cleaned data that readFile.js generates. Next is the set of blocks that is used to replay the block structure at every timestamp. And lastly the format in which the block details need to be present in order to run the Tree Edit Distance algorithm. We have a jar file that runs this algorithm for us. We feed the student's data and the right solution to this jar and get the solution. For the files to be run with the jar file, we need it to be present in a certain format. getTreeFormat.py does this for us.

Finally, we have the getTreeEditDistanceScores.py that extracts the scores of the students with respect to the right solution. Once the scores are in hand, we need to feed it to the UI to get the graph of the student's performance.

The following are the list of files in the codebase that are important to this project:
Playground.js - Used to render the content of Scratch VM
Test.js - Writes the user interaction data into a local file in the form of JSON
readFile.js - Convert nested JSON data to normal text format.
getData.py - Get the required information from the text file.
getTreeFormat.py - Converts the text format data to treeEditDistance jar input format data
getTreeEditDistanceScores.py - Extract the edit distance scores between every student's data and right solutions
UI: Dashboard consists of details for every student. Each student gets his/her own dashboard. For every student, there is a feedback page and a performance analysis page.

- Feedback page will show the feedback given by that student
- Performance analysis page will show the edit distance score which helps the teacher/student analyse the student's progress.

7.3 Coding guidelines used

This section attempts to explain the basic styles and patterns that we used in the codebase.

Whitespace

No tabs. No whitespace at the end of a line.

Unix-style line breaks ('\n'), not Windows-style ('\r\n').

Line length

80 characters or less (for laptop side-by-side diffing and two-window tiling; also for Bonsai / hgweb and hardcopy printing).

Indentation

Four spaces in Python code and JavaScript.

Control structures

Use K&R bracing style: left brace at end of first line, cuddle else on both sides.

Note: Class and function definitions are not control structures; left brace goes by itself on the second line and without extra indentation for JavaScript.

Always brace controlled statements, even a single-line consequent of an if else else. This is redundant typically, but it avoids dangling else bugs, so it's safer at scale than fine-tuning.

Break long conditions after && and || logical connectives. See below for the rule for other operators.

Examples:

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

7.4 Sample code

JavaScript

```
var fs=require('fs');
fs.readFile(process.argv[2],'utf8', function(err, contents) {
    if (err) {
      return console.error(err);
    }
    var contentsList = contents.split("\n");
    for(var line=0;line<contentsList.length;line++) {
      var array = contentsList[line].split(/,(.+)/);
      var Timestamp = array[0];
      var time = "Timestamp: " + Timestamp + "\n";
      var Data = array[1];
      try {
        var jsonObj=JSON.parse(Data);
        var blockIDs=Object.keys(jsonObj['_blocks']);
        var opcodesList = [];
        for(var i=0; i<blockIDs.length; i++) {
            var opcode = [];
            var innerkeys = Object.keys(jsonObj['_blocks'][blockIDs[i]]);
            for(var j=0; j<innerkeys.length; j++) {
                if(innerkeys[j] == 'opcode') {
                    opcode.push(jsonObj['_blocks'][blockIDs[i]][innerkeys[j]]);
                }
                else if(innerkeys[j] == 'inputs') {
                    opcode.push(JSON.stringify(jsonObj['_blocks'][blockIDs[i]][innerkeys[j]]));
                }
                else if(innerkeys[j] == 'fields') {
                    opcode.push(JSON.stringify(jsonObj['_blocks'][blockIDs[i]][innerkeys[j]]));
                }
                else if(innerkeys[j] == 'id') {
                    opcode.push(JSON.stringify(jsonObj['_blocks'][blockIDs[i]][innerkeys[j]]));
                }
                else if(innerkeys[j] == 'parent') {
                    opcode.push(JSON.stringify(jsonObj['_blocks'][blockIDs[i]][innerkeys[j]]));
```

Python

```
#print len(solutionsListForLevel1)
def createCsvs(studentCsv1, studentInputFile):
        studentScores = studentCsv1
        #studLevel2 = studentCsv2
        #tree format input text file of student block data
        fileName = studentInputFile
        tempFile = '/home/vanrao/scratch-vm/LEAPER-master/WorkingLeaper/temp.txt'



        #for solution 1
        ff = open(studentScores, 'w')
        #level2file = open(studLevel2, 'w')

        def csvwrite(timestamp, score):
                csvwriter = csv.writer(ff, delimiter=',')
                global count
                global startTime
                if(count == 0):
                        startTime = timestamp
                        count = 1
                        #print timestamp
                        percentageTime = (int(timestamp)-int(startTime)) / 1000 ;
                        #print percentageTime
                        '''csvwriter.writerow([timestamp, score[0], score[1], score[2], score[3], score[4],
                        csvwriter.writerow([percentageTime,score])
                else:
                        #print timestamp
                        percentageTime = (int(timestamp)-int(startTime)) / 1000 ;
                        #print percentageTime
                        '''csvwriter.writerow([timestamp, score[0], score[1], score[2], score[3], score[4],
                        csvwriter.writerow([percentageTime,score])
```

7.5 Unit Test cases

The different modules in the project were tested separately to perform unit testing. The Python unit testing framework, sometimes referred to as “PyUnit,” is a Python language version of JUnit, by Kent Beck and Erich Gamma.

Test1 - createData.py

Test2 - getTreeFormats.py

Test3 - getTreeEditDistScore.py

Test4 - TreeScoresForAll.py

7.6 Metrics for unit test cases

Number of test cases executed : 75

Number of test cases passed : 70

Number of test cases failed : 5

Number of test cases blocked : 0

Test coverage : 93 %

7.7 Updated RTM

6.2.1 Project Requirement Document (PRD)

PR#	Module Name	Applicable roles	Description
PR1	Designing Scratch problem	Developer, Teacher	Developer: Design a problem statement which includes basic programming concepts. Teacher: Corresponding to the curriculum design a problem statement in scratch appropriate for the students to work on.
PR2	Scratch	Student	Student: Work on Scratch to solve the given problem statement.
PR3	Logging Engine	Student, Developer	Student: Interacts with the scratch application and logger logs the user data. Developer: Get the student interaction data using logging engine.
PR4	Analysis Dashboard	Teacher, Student	Teacher: Get the feedback about the students' performance Student: Get the results about their performance.

6.2.2 Technical requirement Document(TRD):

TestCase ID	PR#	TR#	TestCase	TestData	Expected Result
1	PR1	TR1	Verify the student's interaction with scratch application	Observation forms, Feedback forms.	Positive feedback from students
2	PR2	TR2	Verify the problem such that difficulty level includes only basic programming concepts.	Scratch project developed.	Basic concepts covered
3	PR2	TR3	Ensure that the problem statement loads without internet connection	JSON objects in a text file of the scratch problem statement	Loading of Scratch VM with the problem statement without internet
4	PR3	TR4	Verify student is comfortable with the basic concept problems.	The interaction data collected while students interact	Log file of interaction data.
5	PR4	TR5	Verify the data is logged	JSON object	Log file is saved into the database.
6	PR5	TR6	Data mining on the log files	Log file	Mined results

7	PR5	TR7	Show the results to the user.	Dashboard	Get feedback about the student

6.2.3 Requirement Traceability Matrix:

PR#	TR#	TestCase ID
PR1	TR1	1
PR2	TR2, TR3	2,3
PR3	TR4	4
PR4	TR5	5
PR5	TR5, TR6	6,7

CHAPTER 08
TESTING

8. Testing

8.1 System/Function test Specification for the project

System testing is done to test the entire system to ensure that it is working according to the specific requirements. It is an end to end testing done to verify the entire working of the system. System testing is conducted to test the integration of each module in the system and to find any discrepancies between the system and its original objective. It was found that there was an agreement between the original documentation and the system upon performing system testing. However, the following are the steps that were taken during system testing of the project.

The modules of the project to be tested can be divided into two parts. Firstly, the tests to check the real time data collection and secondly, the tests to check the python code which works on the data collected.

The most of the tests related to the real time data collection had to be done manually without the help of any external testing software tools. The following were tested in this part:

- The offline loading of the scratch project onto the Scratch VM
- The logging of data when events occur on the workspace of the Scratch VM

To test the modules written in python we fed sample data whose results on manipulation were known and then we checked if the results obtained on running the code with the sample data were same.

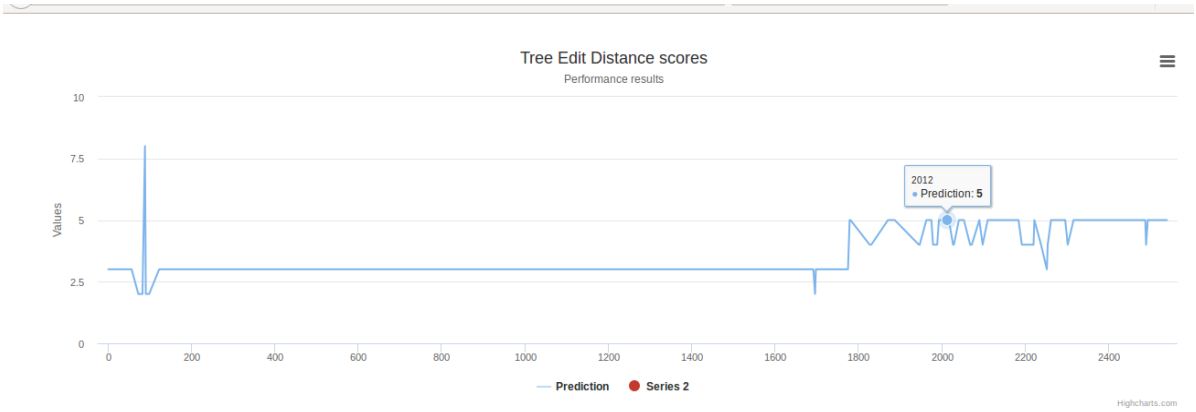
8.2 Test Environment Used

Unit test, the Python unit testing framework, sometimes referred to as “PyUnit,” is a Python language version of JUnit, by Kent Beck and Erich Gamma. Unittest supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The Unit test module provides classes that make it easy to support these qualities for a set of tests.

8.3 Test Procedure

Running the TreeScoresForAll.py will return the scores of each student’s data against each correct solution. The result is a CSV file with timestamp and score. Plotting the scores in CSV file against each timestamp gives the overview of the student’s performance.

8.4 Example test result



The legend of the graph has to keys: Prediction and Series 2.

Prediction indicates the closest distance score of the student to the right solution for the problem at that timestamp. Series 2 indicate another graph if there was another graph of another student to compare with.

8.5 Test metrics

Number of test cases executed: 75

Number of test cases passed : 70

Number of test cases failed : 5

Number of test cases blocked : 0

Test coverage: 93 %

8.6 Updated RTM

8.6.1 Project Requirement Document (PRD)

PR#	Module Name	Applicable roles	Description
PR1	Designing Scratch problem	Developer, Teacher	Developer: Design a problem statement which includes basic programming concepts. Teacher: Corresponding to the curriculum design a problem statement in scratch appropriate for the students to work on.
PR2	Scratch	Student	Student: Work on Scratch to solve the given problem statement.
PR3	Logging Engine	Student, Developer	Student: Interacts with the scratch application and logger logs the user data. Developer: Get the student interaction data using logging engine.
PR4	Analysis Dashboard	Teacher, Student	Teacher: Get the feedback about the students' performance Student: Get the results about their performance.

8.6.2 Technical requirement Document(TRD):

TestCase ID	PR#	TR#	TestCase	TestData	Expected Result
1	PR1	TR1	Verify the student's interaction with scratch application	Observation forms, Feedback forms.	Positive feedback from students

2	PR2	TR2	Verify the problem such that difficulty level includes only basic programming concepts.	Scratch project developed.	Basic concepts covered
3	PR2	TR3	Ensure that the problem statement loads without internet connection	JSON objects in a text file of the scratch problem statement	Loading of Scratch VM with the problem statement without internet
4	PR3	TR4	Verify student is comfortable with the basic concept problems.	The interaction data collected while students interact	Log file of interaction data.
5	PR4	TR5	Verify the data is logged	JSON object	Log file is saved into the database.
6	PR5	TR6	Data mining on the log files	Log file	Mined results
7	PR5	TR7	Show the results to the user.	Dashboard	Get feedback about the student

8.6.3 Requirement Traceability Matrix:

PR#	TR#	TestCase ID
PR1	TR1	1
PR2	TR2, TR3	2,3
PR3	TR4	4
PR4	TR5	5
PR5	TR5, TR6	6,7

CHAPTER 09
RESULTS AND DISCUSSIONS

9. Results and Discussions

The data collected in the second attempt were around 25 in number. The data were skewed and we couldn't perform any analysis on it. Finally, the data collected from third attempt was successful and around 75 students' data was collected.

Since there are many ways of approaching and solving a problem, we would expect number of different solutions from the students. Among the logged results, after performing the edit distance scores, results were pretty interesting. Around 28 different solutions were found for Level-1 and 5 different solutions for Level-2. Most of the students didn't attempt Level-3, so it wasn't considered into the analysis. Among the different solutions, we clustered them into different types. For every student, a graph is plotted against nearest correct solution. If the graph hits 0 on X-axis, then the student successfully solved the problem.

To achieve this, student's solution must be verified with every unique correct solution and minimum of the edit distance scores has to be plotted at every timestamp. By doing this, teacher can monitor the student's progress whether the student is struggling or not while solving the problem and will not have to depend on the result. Besides the field work and technical issues, we faced quite a few challenges while extracting the information from the logged data. One of them was to buffer the data while writing into the file.

Among the results, there were a few which were quite interesting. For a few students, feedback responses matched with their performances but for some, it didn't. After analysing the logged data, we realised that some students refreshed the page after playing level-1 because of which level-1 was logged again into the files which resulted in confusion. Some of the feedback responses were quite appealing. One of the students wanted the workshop to happen again since it helped him improve him his logical thinking.
Eg : Student who solved the level-1

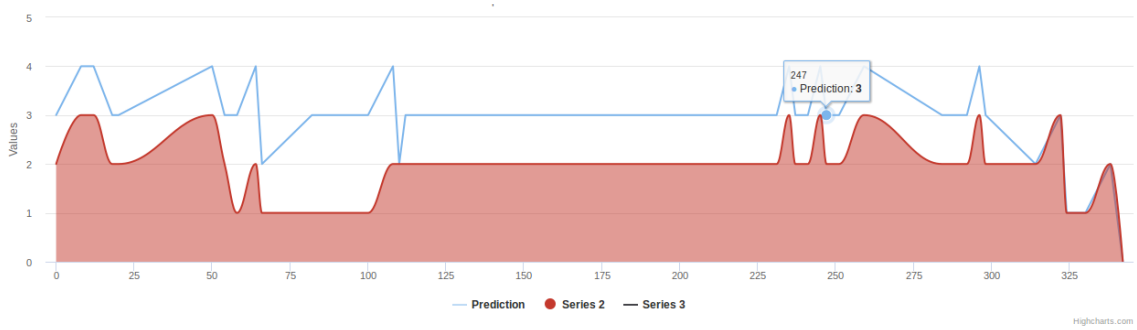


Fig 9.1 Graph-1

Eg: Student who didn't solve level-1

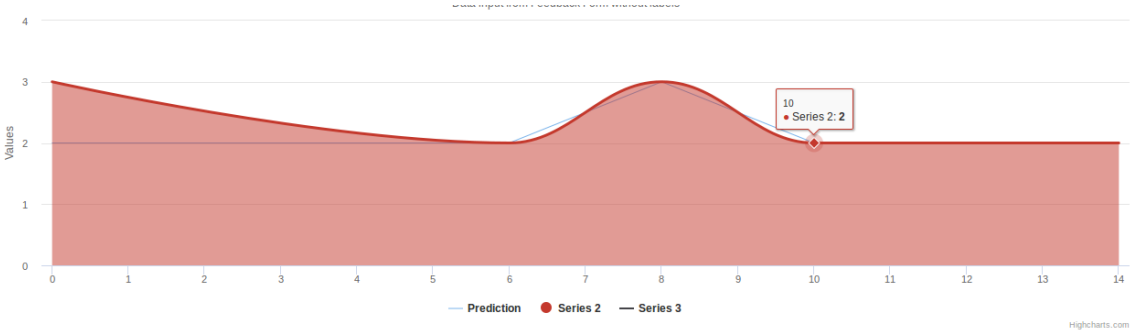


Fig 9.2 Graph-2

From the result of the graphs obtained above, we can understand that struggle can be identified nevertheless irrespective of whether the final right answer is known beforehand or not. 'Prediction' indicates the performance of the student when the final right solution the student was aiming at was available. 'Series 2' indicates the performance of the student when the final right solution was not available. And hence this graph was calculated from the min(S') where S' is the set of exhaustive right solutions available at the time.

CHAPTER 10
RETROSPECTIVE

10. Retrospective

The research work involving children is a very enthusiastic experience. Besides, it is extremely hard to achieve what we want while working with students to collect data. The necessary technical resources matching our requirement should be present in the school to conduct any analysis on the students' useful for our research. The scarcity of resources, difficulty in fixing a schedule for conducting the workshop in schools, managing the students during the workshop, getting our tool to work were some of the challenging hurdles we faced during this project.

A lot of field work was involved in successfully conducting workshops at different schools. We had to schedule meetings with the principal of the schools, explain the objective of the project and convince them to agree to give us slots for conducting the workshops. Initially, Sri Kumaran's Public School gave us the opportunity to conduct the workshop at their campus. Due to the Internet bandwidth issues this first attempt was not successful. Next, we approached Amaatra Academy with the detailed list of our requirements. They agreed to give us the time to conduct workshop. Yet again due to the Internet issues, we could only get a handful of data which wasn't enough to make the analysis. Finally, the workshop that we conducted at DPS North turned out to be a good success for our project where we could get enough data for the analysis. Data collection being the crucial part of the project, we had to try a lot to get this data from the schools. And every mistake that stopped us from collecting the data at a particular school was rectified in the next school we approached.

Coming to the positive side of this project, we were able to interact with the students directly during the workshop. We collected feedback from the students regarding their experience playing the game through feedback forms given to them at the end of the workshop. The remarks were quite overwhelming. Many of them loved the workshop and wanted us to conduct take more such workshops. Most of them found the game interesting and fun. One of them remarked about how the workshop helped him think logically. The technical aspect of interaction with kids was completely different. We get to know how students think and behave. We got to know the psychological aspects of how students react when they are interested and vice versa.

The overall experience involving, working with the students and the field work to get the workshops done and analysis on the data was very challenging one where we got know and understand some of the administrative policies of how schools function and other such general information. Overall, it was a very informative project which paved us way to learn many things

CHAPTER 11
REFERENCES

11. Reference and Bibliography

[1] Lynne (E. F.) McKechnie, “Ethnographic Observation of Preschool Children”

[2] Brennan, K., and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the American Educational Research Association (AERA) annual conference.

[3] S. A. Nikou and A. A. Economides, “Measuring Student Motivation during "The Hour of CodeTM" Activities”

[4] A. Sinha, A. M. Dhareshwar, L. Saloni Joshi and V. Kumar, “Mathematics Tutoring Apps for Low-Cost Devices: an Ethnographic Study of Requirements”

[6] M. Pawlik and N. Augsten. Tree edit distance: Robust and memory-efficient. Information Systems. 2016.

[7] M. Pawlik and N. Augsten. Efficient Computation of the Tree Edit Distance. ACM Transactions on Database Systems (TODS). 2015.

[8] M.Pawlik and N.Augsten. A Memory-Efficient Tree Edit Distance Algorithm. DEXA. 2014.

[9] M.Pawlik and N.Augsten. RTED: A Robust Algorithm for the Tree Edit Distance. PVLDB. 2011.

[10]Journal of the Learning Sciences, Volume 22, 2013 - Issue 4: Learning Analytics and Computational Techniques for Detecting and Evaluating Patterns in Learning

[11] Journal of the Learning Sciences, Volume 22, 2013 - Issue 4: Learning Analytics and Computational Techniques for Detecting and Evaluating Patterns in Learning

[12] Journal of the Learning Sciences, Volume 23, 2014 - Issue 4 : Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming

CHAPTER 12
USER MANUAL

12. User Manual

This project aims at identifying what a struggle looks like and helps a teacher monitor students without having to depend on a student explicitly asking her/him doubts regarding the concepts.

During workshop:

1. Install NodeJs with npm package manager
2. Install LEAPER code onto every system on which the student works during the lab/workshop
3. Traverse into the directory of the LEAPER folder, change directory to scratch-vm.
4. Open terminal and type ‘npm start’. This will start Virtual Machine of Scratch VM by loading the problem statement.
5. Open another window and type ‘node test.js’. This will start the NodeJs server on localhost which is used to log the user interaction data during the workshop.
6. Collect the logged data in text files of every student with a unique name to distinguish among the students.

After workshop:

1. Reorganize the JSON data collected by running createData.py module.
2. Finalise different correct solutions for the problem.
3. The tree formated data is fed to TreeScoresForAll.py module to calculate tree edit distance scores.
4. Feed the data score to the Graph visualisation tool which will give the performance graphs of every student.
5. X axis has timestamp and Y axis has tree edit distance scores.
6. This graph shows how far the student’s solution is from the correct solution and which correct solution is he approaching to.
7. Analysing this graph, teacher can analyse the student’s progress after the workshop is completed.
8. During the real time usage, a dashboard is provided to the teacher and all the students’ progress can be viewed on their dashboard. Viewing the progress graph, teacher can monitor student’s performance without having to observe manually.