A RECOMMENDER OF PHYSICAL GAMES FOR LEARNING PROGRAMMING AND COMPUTATIONAL THINKING

MOHAMMAD AHSAN HABIB

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

A RECOMMENDER OF PHYSICAL GAMES FOR LEARNING PROGRAMMING AND COMPUTATIONAL THINKING

MOHAMMAD AHSAN HABIB

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTERS IN COMPUTER SCIENCE (APPLIED COMPUTING, MIX MODE)

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Mohammad Ahsan Habib

Matric No: WOA160022

Name of Degree: Master of Computer Science (Applied Computing)

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

A Recommender of Physical Games for Learning Programming and Computational Thinking

Field of Study: Computers and Education.

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every right in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this work, I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

A RECOMMENDER OF PHYSICAL GAMES FOR LEARNING PROGRAMMING AND COMPUTATIONAL THINKING

ABSTRACT

Keeping students engaged while teaching programming courses is a big challenge for instructors in general, even with the availability of all modern facilities in a classroom. There are numerous approaches to teaching programming such as through online and offline software applications, digital games, Lego bricks, and robotic aids as well as physical activities such as board games, dancing, and unplugged computational thinking (CT) activities. This type of activities is called active learning approach. The objective of the present research is to design, develop, and evaluate a programming and computational thinking assistive tool to bridge the gap through physical and tactile games. The research methodology starts with qualitative preliminary study that was conducted by observing 121 students on different stages and arranged semi-structured interview sessions for 18 randomly selected students. In this study, CT elements are identified and have been mapped to programming codes. Following that, a prototype called Code Analyzer was developed which consists of four main components. i.e. user interface, Control, CT element calculator, and Instructors' window components. This prototype tool is intended to guide instructors to choose suitable physical or tactile activities by analysing students' code in relation to CT elements.

Keywords: computational thinking elements, tactile games, assistive tool

PENCADANG PERMAINAN FIZIKAL UNTUK PEMBELAJARAN PENGATURCARAAN DAN PEMIKIRAN KOMPUTASI

ABSTRAK

Mengekalkan keterlibatan pelajar sambil mengajar kursus pengaturcaraan adalah satu cabaran besar untuk pengajar pada umumnya, walaupun dengan adanya semua kemudahan moden di dalam bilik darjah. Terdapat banyak pendekatan untuk mengajar pengaturcaraan seperti melalui aplikasi perisian dalam talian dan luar talian, permainan digital, blok Lego, dan bantuan robot serta aktiviti fizikal seperti permainan papan, menari, dan aktiviti Pemikiran Komputasi (CT) tanpa plug. Jenis aktiviti ini dipanggil pendekatan pembelajaran aktif. Objektif penyelidikan ini adalah untuk merekabentuk, membangun dan menilai alat bantu pengaturcaraan dan pemikiran komputasi untuk merapatkan jurang melalui permainan fizikal dan taktil. Metodologi penyelidikan ini bermula dengan kajian awal kualitatif yang dijalankan dengan memerhati 121 pelajar pada peringkat yang berbeza dan mengatur sesi wawancara separa berstruktur untuk 18 pelajar yang dipilih secara rawak. Dalam kajian ini, elemen CT dikenalpasti dan telah dikaitkan dengan kod pengaturcaraan. Selepas itu, prototaip yang dinamakan *Kod Analyzer* telah dibangunkan yang terdiri daripada empat komponen utama. iaitu antara muka pengguna, (kawalan, elemen kalkulator CT), dan komponen tetingkap pengajar. Alat prototaip ini bertujuan untuk membimbing pengajar untuk memilih aktiviti fizikal atau taktil yang sesuai dengan menganalisis kod pelajar yang berkaitan dengan unsur CT.

Kata kunci: elemen pemikiran komputasional, permainan fizikal dan taktil, perisian alat bantu

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor, Raja Jamilah Yusof, lecturer at Faculty of Computer Science and Information Technology, University of Malaya. The door to Dr. Raja's office was always open whenever I ran into a trouble spot or had a question about my research or writing. She consistently allowed this paper to be my own work but steered me in the right the direction whenever she thought I needed it.

I would also like to thank the experts who were involved in the validation survey for this research project. Without their passionate participation and input, the validation survey could not have been successfully conducted.

I would also like to thank University of Malaya for granting RU009-2017M RU grand-UMCAS for conducting the preliminary study.

Finally, I must express my very profound gratitude to my parents and to my hall-mates for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author

Mohammad Ahsan Habib

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Background

There are many approaches to teach programming languages and one of them is through assistive tools by means of digital games (Papastergiou, 2009) or multimedia (Brown, 2007). Several such approaches are related to computational thinking (CT), such as Code.org (hour of code), MIT Scratch Programming, Serious Games, Robo games (Weintrop & Wilensky, 2013), The Tessera (Marcolino & Barbosa, 2017), and others (Brady et al., 2017; Farrell, 2014; Weintrop et al., 2016; Weintrop & Wilensky, 2013). CT on the other hand is the thought process required to be able to translate formulated ideas into data and instructions in a computer or even translating them into tasks to be carried out in the real world (Ortiz et al., 2017; Rosen, 1831; Wing, 2006; Wolfram, 2016). It is, therefore, a skill needed for programming. From a different perspective, programming is a subset of CT. A few practitioners view CT as an algorithmic process involving formulation of the thinking process to be executed by the computer (Wolfram, 2016), and a few others relate CT to elements such as decomposition, abstraction, automation, pattern recognition, sequential, recursion, and parallelism (Barr & Stephenson, 2011). Therefore, problems associated with learning to program are essentially problems related to CT (Ying Li, 2016).

1.2 Motivation

There are several motivations that drives this research project, one of which is related to active learning. It is an approach to teaching in classrooms in which the student is at the centre. In an active learning environment, the teachers' role closely resembles that of a facilitator, and students are engaged in activities to stimulate the learning of certain topics (as mentioned in Appendix A, Table A, column Game.). Common examples of active learning include classroom discussions, presentations, and hands-on experience

(Roehl et al., 2013). The traditional approach is teacher-centred, and knowledge is delivered to the students by the teacher in a classroom.

Contrary to popular belief, the brain is not designed for thinking. It is designed to save you from having to think, because the brain is actually not very good at thinking. Thinking is slow and unreliable. Nevertheless, people enjoy mental work if it is successful. People like to solve problems, but not to work on unsolvable problems. If schoolwork is always just a bit too difficult for a student, it should be no surprise that she doesn't like school much (Willingham, 2012). Moreover, they are distracted by other things such as smartphone in classroom (Anshari et al., 2017). As they need to keep focused continuously on the lecture board without having them involved actively in the lecture which is contradictory to the nature of our cognitive brain (Willingham, 2012). Therefore, something different is needed compared to conventional model of teaching. Learning environment should be reformed by making it enjoyable, competitive, fun, engaging and content rich. By employing some competitive physical activities in a class with specific targets, educators can resolve this and make students involved and get undivided attention from learners.

Another motivation is that there had not been extensively studies of the effectiveness of physical and tactile movements to teach programming concepts. This approach is motivated by learning style theories that students learn in different manners. For example, according to the Kolb learning style (Ateş & Altun, 2008), students learning can be classified as Concrete Experience (Feeling), Abstract Conceptualisation (Thinking), Active Experimentation (Doing), and Reflective Observation (Watching) (Campbell & Johnstone, 2010). Sprenger, on the other hand, defined learning styles as auditory, kinaesthetic, and visual (Ateş & Altun, 2008; Klement, 2014; Noor et al., 2014). Norwawi defined learning styles as active/reflective, sensing/intuitive, visual/verbal, and sequential/global (Norwawi et al., 2009). In context of teaching programming, it can be sensed that most teaching approaches minimally benefit the active and kinaesthetic types, and therefore, physical and tactile games can be used to promote understanding of fundamental and advanced concepts of programming.

1.3 Problem Statement

Computer programming courses play a major role in preparing programmers who can produce state-of-the-art software and applications. Currently, the social media landscape leads the direction of technology with the support of cloud and big data. Teaching programming courses have always been a challenge for many lecturers and instructors (Hegazi & Alhawarat, 2015; Ortiz et al., 2017; Yusof & Abdullah, 2005), even with the tremendous effort invested by institutions (Lethbridge, 2000). The success rate for teachers to teach students is generally low especially in introductory programming language classes. From this viewpoint, a few problems have been identified, i) difficulty in applying basic programming knowledge (Ab Hamid, 2004), ii) translating problems into solutions in the form of data and instructions (Ortiz et al., 2017), and iii) understanding existing codes and pseudocode (Marcolino & Barbosa, 2017). Problems related to programming can essentially be associated with problems related to CT (Ying Li, 2016).

It is envisioned that an assistive tool is needed to relate program codes to CT elements. Program codes written by students can be analyzed. Physical and tactile games can therefore be suggested based on the CT elements found lacking based on the analysis of the tool.

1.4 Objectives

The main aim of the present research is to bridge the gap between programming and computational thinking. A model tool (experimental technique) was designed and built to address this issue by which the CT elements can be connected with programming easily and subsequently be able to suggest CT activities through physical and tactile games. Therefore, the following objectives are needed to achieve the main aim.

i. To explore students' engagement in a programming classroom through computational thinking activities
ii. To associate CT elements in program codes
iii. To develop an assistive tool that gives output of CT elements from a program code
    a. To generate a reference sheet of standard program to identify CT elements
    b. To analyze students' code with the reference code
    c. To give suggestions for classroom activities on the basis of obtained CT elements
iv. To test and evaluate the assistive tool

1.5 Research Questions

Next, A prototype is sculpted and developed in web application platform named *Code Analyzer* by which CT elements can be calculated in terms of percentage value from a given programming code. *Code Analyzer* tool can compare CT elements values of a class or a student with its reference sheet. It can suggest classroom activities based on the performance and CT score of a student or a class. The following research questions have been addressed:

i. Do students enjoy classroom activities while it enhances their understanding of programming concept and improves their academic performance?
ii. How the CT elements relates to the code?
iii. Does the tool able to generate CT elements correctly?

iv. Does the tool able to compare CT elements correctly?

v. Is the tool able to suggest suitable activities based on students need?

vi. Does this tool able to help teachers/instructors to guide to choose CT class activities?

1.6 Scope of Research

Eight common activities namely Bubble Sort, Selection Sort, Quick Sort, Merge Sort, Insertion Sort, Shell Sort, Graph Theory has been covered in this study. And web platform is opted using PHP Laravel framework since this application needs to be accessible from anywhere from any computer through browser.

1.7 Methodology

CT concepts are the mental processes (e.g. abstraction, algorithm design, decomposition, pattern recognition, etc.) and tangible outcomes (e.g. automation, data representation, pattern generalization, etc.) associated with solving problems in computing. There are 11 concepts of CT (Jimoyiannis & Tsiotakis, 2017) as follows:

- Abstraction: Identifying and extracting relevant information to define main idea(s)
- Algorithm Design: Creating an ordered series of instructions for solving similar problems or for doing a task
- Automation: Having computers or machines do repetitive tasks
- Data Analysis: Making sense of data by finding patterns or developing insights
- Data Collection: Gathering information
- Data Representation: Depicting and organizing data in appropriate graphs, charts, words, or images
- Decomposition: Breaking down data, processes, or problems into smaller, manageable parts

- Parallelization: Simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal
- Pattern Generalization: Creating models, rules, principles, or theories of observed patterns to test predicted outcomes
- Pattern Recognition: Observing patterns, trends, and regularities in data
- Simulation: Developing a model to imitate real-world processes

First, an extensive literature review was done regarding classroom activities and CT elements to address the objectives. All the CT elements can be narrowed down to four elements based on literatures. They are namely decomposition, pattern recognition, abstraction, algorithm. As of preliminary study, six classroom activities have been designed and exploit in the classroom very carefully to collect comparative performance data of the students. These activities and the data went through evaluation and validation process by three lecturers and all the participants. A tool has been designed to get appropriate classroom activity suggestion by which CT elements have been linked with program based on the information from literature review. A novel relationship was established to translate a program into computational thinking elements. Having this facility, classroom activities can be redirected from these CT elements. Finally, usability of the tool will be evaluated and validated by at least seven university lecturers and seven postgraduate students.

The following flowchart in Figure 1.1 displays the steps to achieve the objectives anticipated for this research while adopting all the research questions stated in an earlier subsection.

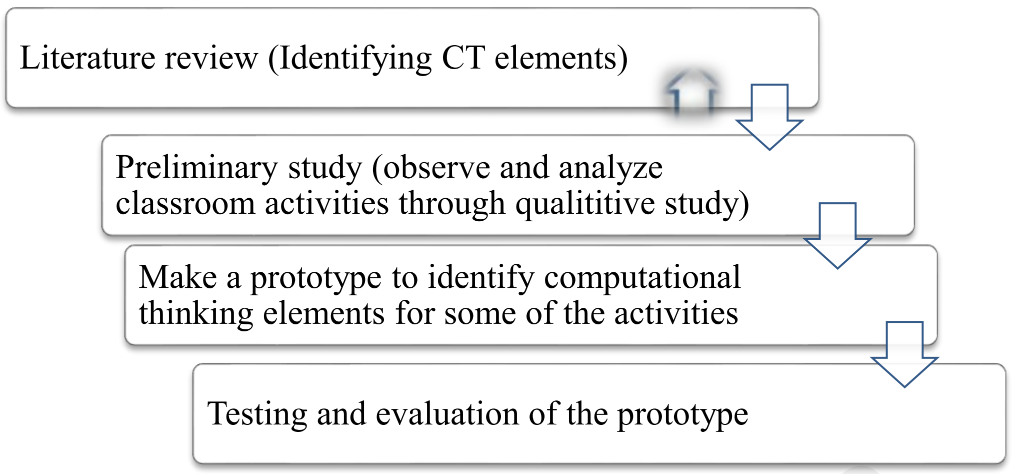

Literature review (Identifying CT elements)
Preliminary study (observe and analyze classroom activities through qualititive study)
Make a prototype to identify computational thinking elements for some of the activities
Testing and evaluation of the prototype

Figure 1.1 Flow chart of research methodology

1.8 Target Group

Target group for the present research are instructors who teach programming language in the universities.

1.9 Significance of the Research

The preliminary study will extend existing knowledge about the impact and effectiveness of nonconventional classroom activities especially using physical games. The present study will identify CT elements from program codes through an assistive tool. The novelty in this study is that for the first time a system is able to analyze python programming code and produce four CT elements. This idea can be scaled up by including more programming languages.

1.10 Overall structure of the Thesis

The following sections cover the literature review, research methodology, CT element identification, tool design and implementation, tool evaluation and validation, discussion and conclusion. Chapter one contains introductory research background, motivation, problem statement, objectives of the research, research questions, scope of the research, summary of methodology, target group, thesis structure, and the significance of the thesis.

Chapter two elaborates the literature review of the research finding related works, research gaps, and defining computational thinking. The methodology of this research has been elaborated in this chapter three in four steps. These four steps are illustrated in Figure 1.1. Preliminary study has been described in chapter four to address the first research question of the thesis. And the intended assistive tool has been modelled, developed and implemented in chapter five. The process of testing and evaluating the tool can be found in chapter six. Finally, all the research findings and outcomes have been concluded in chapter seven.

CHAPTER 2: LITERATURE REVIEW

2.1 Related Work

Many software applications have been developed to teach people to program such as BlueJ (David J. Barnes and Michael Kölling, 2013; Kölling et al., 2003) in Australia and Scratch (Resnick et al., 2009) in MIT, United States of America. Some are available online and some requires us to buy the licence to access the features and functionalities. These are software that covers fundamental topics of programming such as variables, if statements, loops, arrays, pointers and data structures (Farrell, 2014). This software is useful although the associated users did not claim any significant difference between those students using the software and not using them because of reasons associated to ethics and difficulties of setting the learning environment in using these software in their courses for experimental studies (Kunkle & Allen, 2016).

Another approach of teaching programming is through robotic technology (Liu et al., 2013) or other external devices that controls the environment such as Lego Bricks programming set, the Raspberry Pi microcontroller and the Arduino circuit board. These need the users to buy the hardware and software involved for the development of the programs and can be quite expansive in nature.

In order for students to be genuinely engaged in computational thinking, teachers need to facilitate an environment what they would be interested in. Researchers found that young people have intrinsic affinity towards playing games (Papastergiou, 2009). Games that incorporate scholastic objectives and themes have the potential to render learning of academic subjects easier, more enjoyable, more interesting, thus, more effective (Kafai, 2001; Malone, 1980). Specifically, games constitute potentially powerful learning environments for a number of reasons (Oblinger, 2004): (i) they can support multi-sensory, active, experiential, problem-based learning, (ii) they favor activation of prior knowledge given that players must use previously learned information in order to advance, (iii) they provide immediate feedback enabling players to test hypotheses and learn from their actions, (iv) they involve opportunities for self-assessment through the mechanisms of scoring and reaching different levels, and (v) they progressively become social environments involving communities of players. One of the two most important factors is affordability of pricing of the online digital contents, LEGO or robotic components for all the students, who are taking the programming courses. Second most important factor is opportunity/availability of interconnected facilities, i.e. electricity, computer or product shops. Therefore, these cannot be the best solutions considering many students are living outside the city area, who are less likely to have the prospect to use electricity let alone computers.

2.2 Assistive Tool for Teaching Programming

One of the most popular methods of teaching tool for programming is through games (Ab Hamid & Leong, 2007; Hamid & Ismail, 2007). Searching through literatures and resources from the internet, 12 games were found that are used as assistive learning tools in teaching programming. Table 2.1 shows the details of each game. Overall analysis shows that these games are suitable for all ages, and many of them are made for computer-based platforms, with only a few in the form of board games. Most of these games are to create an experience that kids, parents, and grandparents could share; and in the process allow children to exercise their immense learning capabilities through play. Most of these games cost less than USD$50. These digital game programs (such as the one by Al-Bow et al. (2009)) are highly rated, and they are offered on multiple platforms such as the web, mobile, and desktop. Such an approach raises interest because many young students themselves are regular players of digital games. The interest in playing games eases the learning process. However, in terms of non-game-based educational programming software, there is no hard evidence of effectiveness, although the interest showed in these games is already a significant finding to consider them as tools for teaching programming.

Table 2.1 Game based assistive tools

No	Name of Game	Age (years)			Platform		Pricing (USD)		
		<7	7 - 13	>13	Computer Board game	Board game	<10	10 - 50	>50
1	Lightbot and Lightbot Jr.	✓	✓		✓		✓		
2	Code Monkey Island	✓				✓		✓	
3	Kodable	✓			✓		✓		
4	Robozzle	✓	✓	✓	✓		✓		
5	Cargo-Bot	✓	✓	✓	✓		F		
6	SpaceChem		✓	✓	✓		✓		
7	Robot Turtles	✓	✓			✓		✓	
8	Code Combat	✓	✓		✓		F		
9	Ludos	✓	✓		✓				✓
10	Codemancer		✓	✓	✓			✓	
11	Machineers		✓	✓	✓				
12	Bee-Bot	✓	✓		✓		✓		

(Source: http://venturebeat.com/2014/06/03/12-games-that-teach-kids-to-code/view-all/)

Note: F == Free

2.3 Unplugged Computer Science

Unplugged Computer Science is a novel approach in the pedagogical realm of teaching computer science. It promotes CT. Nevertheless, it can be adapted a to range of other disciplines, for example, life sciences (Rubinstein & Chor, 2014). In addition, Unplugged activities are widely and freely available as materials and reference books, such as the book uploaded under the Creative Commons Licence (Bell et al., 2006).

Table 2.2 shows existing physical activities related to programming topics. Sorting algorithms such as bubble sort, insertion sort, and quicksort have been demonstrated using traditional Hungarian and Romanian dances. Other examples include the physical activity of students becoming boxes to explain variable concepts and passing batons to demonstrate flow control.

Table 2.2 Examples of physical activities to teach programming

No	Physical activity	Programming concept	Source of the content
1	Human sort	Bubble sort	https://www.youtube.com/watch?v=8QD-R_MfDsQ
2	Hungarian folk dance	Bubble sort	https://www.youtube.com/watch?v=lyZQPjUT5B4
3	Romanian folk dance	Insertion sort	https://www.youtube.com/watch?v=ROalU379l3U&list=RDlyZQPjUT5B4
4	Hungarian folk dance	Quicksort	https://www.youtube.com/watch?v=ywWBy6J5gz www.merlot.org/merlot/viewMaterial.htm?id=1132554
5	Students become boxes	Variables Swaps	https://teachinglondoncomputing.files.wordpress.com/2014/02/activity-boxvariables.pdf
6	Passing baton	Loops If-Else Flow control	https://teachinglondoncomputing.org/free-workshops/programming-unplugged-programming-without-computers/

However, these activities do not directly measure the understanding of the audience or the actors/performers but surely improve their understanding on the related topic (referencing to preliminary study). There can be several reasons for them to not measure the understanding, i) they didn't focus on actual understanding of students. ii) measuring understanding is quite counterintuitive normally, thus haven't been measured; on the other hand, we measured it by explicitly asking student feedback.

The primary goal of the Unplugged project is to promote computer science among young people as an interesting, engaging, and intellectually stimulating discipline (Bell, Alexander, Freeman, & Grimley, 2009). It can effectively convey fundamentals that do not depend on specific software or systems, ideas that will still be fresh in 10 years (Bell et al., 2006). This method can be useful where high-tech educational solutions are infeasible to implement (Curzon, 2013). Unplugged activities are not only a powerful way to familiarise children and students with computing concepts, and a study shows that they are a powerful way to introduce computing concepts to adult teachers (Curzon, McOwan, Plant, & Meagher, 2014). However, such games are not used in the mainstream standard educational structure, except for scattered personal efforts by a few educators.

2.4 Computational Thinking (CT)

Computational thinking allows us to take a complex problem, understand what the problem is and develop possible solutions. In this way, humans can present these solutions in a way that a computer, or a human, or both, can understand. It involves taking that complex problem and breaking it down into a series of small, more manageable problems (decomposition). Each of these smaller problems can then be looked at individually, considering how similar problems have been solved previously (pattern recognition) and focusing only on the important details, while ignoring irrelevant information (pattern generalization and abstraction). Next, simple steps or rules to solve each of the smaller problems can be designed (algorithms) (Curzon et al., 2014; Silapachote & Srisuphab, 2017; Weintrop et al., 2016; Wolfram, 2016). And therefore, through the literature review four major CT elements namely Decomposition, Pattern Recognition, Abstraction, Algorithm were found.

2.4.1 Decomposition

It merely indicates the method of breaking a bigger problem into smaller problems so that it can be conceived and managed easily (Barr & Stephenson, 2011).

2.4.2 Pattern Recognition

Once a complex problem have been decomposed into smaller problems the following step is to look at similarities they share (Sa Lorca, 2018). Patterns are shared characteristics that occur in each individual problem (Bishop, 2008).

2.4.3 Abstraction

"Abstraction" refers to focusing on the important information only while ignoring irrelevant features. In order to achieve a solution, a close look right through unnecessary traits is needed to focus on those that we do (Sa Lorca, 2018). The process of abstraction can be seen as an application of many-to-one mapping (Hazzan & Kramer, 2008).

2.4.4 Algorithm

An algorithm is a plan; a set of step-by-step instructions used to solve a problem (Sa Lorca, 2018). A program code that gives output correctly as intended is algorithmic. It relates to obtain an intended output by following a definite sequence. Interpolating this idea of solving a problem to a smaller regime, it can be found that functions are used to solve an apparent smaller problem. For a good solution, statements need to be placed in correct sequence hence the use of proper sequence of statements are algorithmic.

2.5 Connecting CT Elements with Programming

The ability to use the concepts of computer science to formulate and solve problems is CT. Coding is generally understood as a tool to teach CT, but CT entails a wider range of abilities (C. Lee et al., 1997). A link will be formed between CT elements and coding in the following sections.

2.5.1 Decomposition

In programming, each keyword is used to address a small problem. Likewise, calculating values, assigning values, or calling functions are the statements to resolve a small problem each and contribute to succeed a bigger problem. Hence, it can be said that use of keywords, assignments and functions have straightforward correlation to the idea of decomposition.

2.5.2 Pattern Recognition

While comparing two things, either some similarities or dissimilarities were considered that is pattern. In the same way, when a small problem of similar pattern being repeating repeatedly, it is better to put in the loop. Hence, every loop asserts a pattern to be solved.

2.5.3 Pattern Generalization / Abstraction

In programming, the act of calling a function induced the perception of abstraction. The process of abstraction can be seen as an application of many-to-one mapping (Hazzan & Kramer, 2008). i.e. the exact role of a function upon its usage.

2.5.4 Algorithm

An algorithm is not related to code itself but to achieve the desire output by ordering statements appropriately. An algorithm is a plan; a set of step-by-step instructions used to solve a problem (Sa Lorca, 2018). A program code that gives output correctly as intended is algorithmic. It relates to obtain an intended output by following a definite sequence. Interpolating this idea of solving a problem to a smaller regime, it can be found that functions are used to solve an apparent smaller problem. For a good solution, statements need to be placed in correct sequence hence the use of proper sequence of statements are algorithmic. The mapping layout can be found in Table 2.3.

Table 2.3 Mapping CT elements to programming

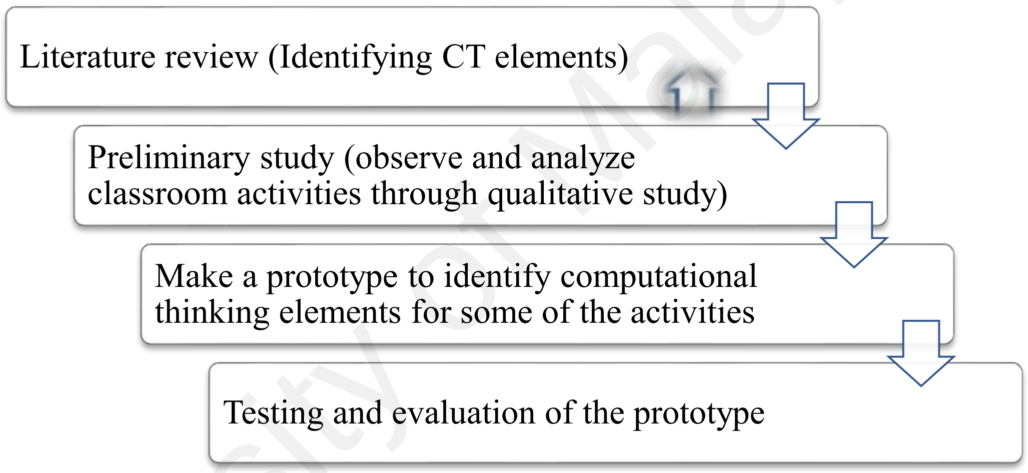
CT elements	Programming notation
Decomposition	Keywords, assignments
Pattern Recognition	Loops, comparisons, Conditions, function definitions
Abstraction	Function calls
Algorithmic	A correct sequence of statements that capable of produce output, defining a function, loop, statements

All the assistive tools we have found above including hardware-based gaming tool for learning, software-based learning applications, even the unplugged games have not really identified the computational thinking elements. But the approach should be able to address CT elements in gaming activities and enrich it with assistive functionalities.

CHAPTER 3: RESEARCH METHODOLOGY

This research study has been done in four steps. These steps are design so that all five research questions are addressed. Research questions are mentioned in section 1.5 of chapter one.

The following flowchart in Figure 3.1 shows the steps to achieve the objectives intended for the present research while addressing all the research questions mentioned above.



Literature review (Identifying CT elements)
Preliminary study (observe and analyze classroom activities through qualitative study)
Make a prototype to identify computational thinking elements for some of the activities
Testing and evaluation of the prototype

Figure 3.1 Flow chart of research methodology

3.1 Preliminary Study

A primarily qualitative research method was adopted that involves observing 123 students who participate in the aforementioned games during lecture time. Semi-structured interview sessions are conducted with and open-ended survey questions are administered to 18 students, and their opinions regarding their experiences of the lecture sessions are recorded. The recorded data are analysed using thematic coding based on the following research questions:

a) Do students show interests in participating in/playing the games?

b) How do the competitive physical games help the students understand computational thinking concepts more effectively?

c) Do the students show confidence in handling CT concepts?

d) Do the games improve the students’ understanding of programming concepts?

e) Did the students’ interests enhance their academic performance?

The overall result of the interviews was validated and then analysed through thematic analysis. This is to answer questions related to interest, understanding and confidence of the programming and to analyse whether computational thinking elements are activated in the thinking process.

In the thematic analysis, each of the themes was designed through the lens of the research questions. Students were asked specific questions based on these themes. These feedback-questionnaire can be found in appendix C. Their diverse responses were grouped into sub-theme categories aligned with the main themes.

Examination result was analysed of one of the students’ groups involved in the game activities and been compared to the previous year students’ group who did not go through such activities. This is to answer the fifth research question.

3.1.1 Activities

Six most popular and common unplugged sorting games (Bell et al., 2006; Curzon, 2013) available online are selected for the preliminary study. The authors were inspired by the unplugged computer science activities conducted by Curzon (Bell et al., 2006; Bell et al., 2012; Bell & Newton, 2013; Curzon, 2013) and CT video materials from Code.org. They are Quick Sort, Merge Sort, Selection Sort, Insertion Sort, Radix Sort, Bubble Sort. Details of the games attached in the appendix A. These activities were conducted on university students to measure their engagements in the classroom.

Evaluation of the activities were done in two steps:

3.1.1.1 Academic Test Score

The test score or academic result of the students will be considered and statistical t-test will be done on their scores to find if there is any significant difference between the groups. (Glen, 2016)

3.1.1.2 Feedback

Participants' direct feedbacks were taken by asking specific questions to measure their engagement in class. A thematic analysis has been done to elaborate the understanding of their participation.

3.2 Identify CT Elements

Through literature review the main elements of computational thinking are identified. Following that these elements are being connected with programming codes on the basis of their common essences.

3.3 Prototype (Assistive Tool)

3.3.1 Target Users

The targeted users were both the students and the teachers since this will allow more collaboration between them. Further it will allow the teachers to advise suggestions for the students.

3.3.2 CT Elements Calculation Mechanism

Using the model above four CT elements are calculated (detail of the calculation structure is shown in section 5.1.2) in relative percentage by analysing the code through regular expression and output obtained.

3.3.3 Activity Suggestion Module

Here, Instructors will be given classroom activity suggestions based on the average performance of any specific group of students in a classroom. Also, teachers will have the possibility to add new activities.

3.4 Tool Development (Experimental Technique)

It will require database to store information, programming language for development, backend framework support the structure of the application, frontend technology have user interface, and a server to maintain the protocols between users and system. There are four modules for the prototype. They are described as follows,

3.4.1 User Interface

There are four main components of the user interface (UI). The following things will be addressed in the UI developments, i) Code canvas, ii) CT element display, iii) Add new activity, iv) See activity suggestions for classroom.

3.4.2 User Type

Role selection: there will be two types of users— students and lecturers. Teachers (i.e. lecturers) will get all the access of the four components while student role is restricted to use the sections that are in teachers' window.

3.4.3 CT Elements Calculation

The proposed model for computational thinking elements in Figure 5.1 would be implemented in the model segment of the Model–View–Controller (MVC) architecture in Laravel[1].

[1] **Laravel** is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern and based on Symfony. It can be found here, github.com/laravel/framework

3.4.4 Business Logic

The business logic of the prototype will be scattered all throughout the MVC framework model, view, controller and services. Standard coding convention will be used as advised in the Laravel documentation.

3.5 Implementation

The following steps will be followed to implement. Figure 3.2 shows the major steps of implementation.

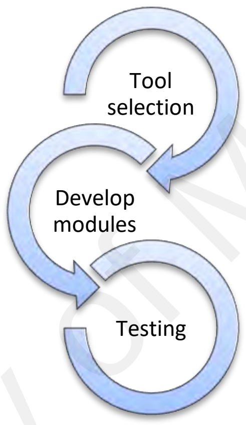


Figure 3.2 Steps of implementation

3.5.1 Tool Selection

Fast, reliable, well documented, and latest are the parameters that will be considered while choosing technologies or tool for the prototype.

3.5.2 Module Development

All the required modules of the tool will be built using PHP programming in MVC architecture of Laravel framework. Programming convention will be followed throughout the development.

3.5.3 Testing

In testing part (experimental technique), the prototype is tested to verify its functionality. This has been elaborated in detail in testing and evaluation chapter.

3.6 Testing and Evaluation of the Prototype

Two basic testing models will be used to test the prototype. For to test each part of the program and show that the individual parts are correct unit test will be conducted and to test overall functionality and interconnections of each of the module integration test will be performed.

University lecturers and educators will be asked to use the tool and evaluate the usability of the tool in three aspects. These are general aspect, structure and navigation, and system evaluation. Evaluation process is further explained in the Testing and Evaluation chapter.

3.7 Summary

This chapter describes overall method of the research. It shows that all four steps are done one after another while preliminary study and literature review was done back and forth. Then, it attempted to connect the CT elements identified through literature reviews. Then a prototype tool is developed to generate CT elements and calculate the mean CT elements of a class to assist teachers to choose a suitable activity for a specific class/group. Then it addresses the implementation of the model designed for tool finish it through testing. All the research objectives shall be achieved by addressing respective research questions.

CHAPTER 4: PRELIMINARY STUDY

This is a qualitative preliminary study that will provide the practical data of the research. Secondly, the design and implement a model to assist in objective number one by linking up CT elements with program. This chapter attempts to reveal preliminary study and the most suitable approach for the gaming activities by means of CT elements. This study aims to explore the students' engagements in a programming classroom through competitive physical and tactile games.

4.1 Game Design

We adopted a naturalistic observational research design. Naturalistic observation captures real-world activities and generates rich data (Johnstone & Kanitsaki, 2006) . The observations were videotaped. This was followed by structured interviews and a survey administered to 18 randomly selected participants; the interview questions included research questions that could not be answered by means of observation alone.

4.1.1 Study Setting and Participants

We proposed playing competitive physical and tactile games by using learners' body movements and a part of their motor control (such as hands, body, and legs) in contrast to purely computer-based or board games as tools for teaching programming. This part of the study was conducted in lecture rooms and computer labs. The gaming activities involved first to fourth year undergraduate students. 13 students from the fourth year in the Algorithm class, 44 students from the second year Algorithm class, 21 third year students from the Data Structure class, and nine first year students from an extra programming class, and 36 pre-university class, resulting in a total of 123 students. We randomly selected 18 students to be interviewed in a semi-structured manner. Only 18 of them were interviewed because of a few reasons:

- We invited all, however only 18 responded for the interview session
- For a qualitative data, 18 interviews (approximately 15% of the participants) would be sufficient to carry out the investigation inline with the research objective. As suggested by Creswell (2007) recommends 5 – 25 and Morse (1994) suggests at least six. For phenomenological studies.

Their ages were 20–24 years, and there were sis females and 12 males. Data were collected between September 2016 and January 2017. We arranged another programming session with 34 freshers (12 female and 22 male) aged between 19 to 20. In this session, we had a common presentation and lecture session on basics of programming. Following that they all sat for a pretest. At this stage, they were divided into two groups. They are Gaming Group (GG) and Lecture Group (LG) where only students from GG will participate in computational thinking activities. Following that both groups sat for a posttest. These test questions can be found in the appendix B.

Thematic analysis was adopted through the lens of our research questions. Students were asked specific questions based on these themes. Later, their diverse responses were grouped into sub-theme categories aligned with the themes.

4.1.2 Participant Observation

Students' behaviour during the game activities were observed in the class. A few sessions were videotaped to remind us how the games were conducted and how the students' interacted in and responded to the sessions. The statistics shown in Table 4.1 below.

Students' behaviour during the game activities were observed in the class. A few sessions were videotaped to remind us how the games were conducted and how the students' interacted in and responded to the sessions.

Table 4.1 shows, a total of 11 videos were recorded, and their total duration was 6 minutes and 5 seconds. The videos were recorded in the algorithm, programming, and data structures classes, and they were related to the sorting and graph topics.

Table 4.1 Participant observations and statistics

Observed class	Number of videos recorded	Duration of videos (Seconds)	Topic of focus
Algorithm	3	45, 65, 95	Sorting
Programming (1st year)	5	8, 8, 17, 21, 22	Sorting
Data Structure	3	22, 23, 30	Graph, Sorting
TOTAL	11	6 minutes 5 seconds	

4.1.3 Interview and Survey Procedure

Interviews were conducted in three different ways. First, a structured Google Form and a paper-based form were used to collect data from the participants before meeting or calling them to further clarify their answers. These forms comprised specific MCQs and short open questions. Second, we arranged a personal meeting with the participants to note down their feedbacks about the workshops. Each of the meeting sessions were recorded in audio format. Later, these recordings were transcribed into readable texts, after which, the transcripts were sent to the interviewees for verification. Last; one-to-one self-recorded video interviews were conducted to add to the flexibility of how students could express their answers. The questions were the same as those written in the form earlier. All transcribed interview data were merged before they were analysed.

4.1.4 Grade Performance Observation

Performance observation was conducted by comparing the examination results of two groups of students. The 1st group consisted of 8 students from the Algorithm class in 2016. The second group comprised 12 students from the Algorithm class in 2017. The 2017 group were exposed to the competitive physical games, while the 2016 group was not. Another observation on 36 first-year students from 2018/2019 batch was conducted.

4.1.5 Research Procedure and Data Collection

4.1.5.1 Game Procedure

Two types of activities were performed based on participant involvement. Activities that required only one participant were assigned to the type-I category, and activities that required a group of participants were assigned to the Type-II category. Type-I activities were conducted and observed several times for everyone, and Type-II activities were conducted with a group and observed once in a session.

Activity sessions were conducted in four parts: i) Introduction of things to do, ii) Demonstration, iii) Game among students, iv) Winner selection and present distribution.

4.1.5.2 Game Materials

Eight CT game activities were conducted, namely, swapping, bubble sort, quick sort, merge sort, selection sort, insertion sort, radix sort, and graph theory. Four major components of CT elements (i.e. abstraction, decomposition, pattern recognition and algorithm design) were incorporated while designing all activities/games. The game instructions were provided to two programming lecturers for validation, and positive and negative feedback were recorded for improving game description.

Table 4.2 illustrates the playing team size, materials used, and playable games for each category and gift items. Type-I refers to those games which a single player could play using cards. By contrast, Type-II refers to those games in which students formed groups consisting 5–15 members each.

Table 4.2 List of items based on activity type

Activity Type(s)	Type-I	Type-II
Experiment Sample Size	single student	5–15 students
Material	• Coloured paper cards with number written on it: 10–20 pieces • size: 15 X 10 cm^2	• Coloured paper cards with numbers: 8-12 pieces • size: A4 colour • Alphabet (A-Z) mat
Games	• Quick Sort • Merge Sort • Selection Sort	• Swap • Quick Sort • Bubble Sort • Graph Theory • Bubble Sort
Winners	• 3 Winners get special price • M&M chocolate, Mentos candy, note-books • All get a small bag of sweets for participating.	

4.1.5.3 Grade Performance Observation

Special permission was obtained to analyse the answer scripts of the participating students. This is because, exam scripts are classified for outsiders (i.e. examinees, examiners). Therefore, to get access to have statistical evaluation of those script required special permission. The topics considered in the present study were sorting and graphs. The students' marks remained anonymous so that the result could be presented without any controversies. The two sets of examination questions from two different years were set by the same lecturers, and their difficulty levels were the same. We have concluded the difficulty level being same mainly because of the following reasons, i) same course, ii) conducted by same lecturer, iii) exam question was prepared by same lecturer, iv) same type of question for same topic. The students' marks (max 10 points) were identified and recorded. A total of 20 students' marks were analysed over consecutive exams. It can be inferred from Table 4.3 that the standard deviation (SD) of both groups is almost the same, that is, 1.72 and 1.71, but the mean and median of marks of the 2017 group are higher than those of the 2016 group. Increment in mean (mark) indicates overall improvement of the entire class. Median (mark) indicates the mark of a student who represents the midpoint of the distribution.

Table 4.3 Comparative observation of SD, mean, and median marks of two groups

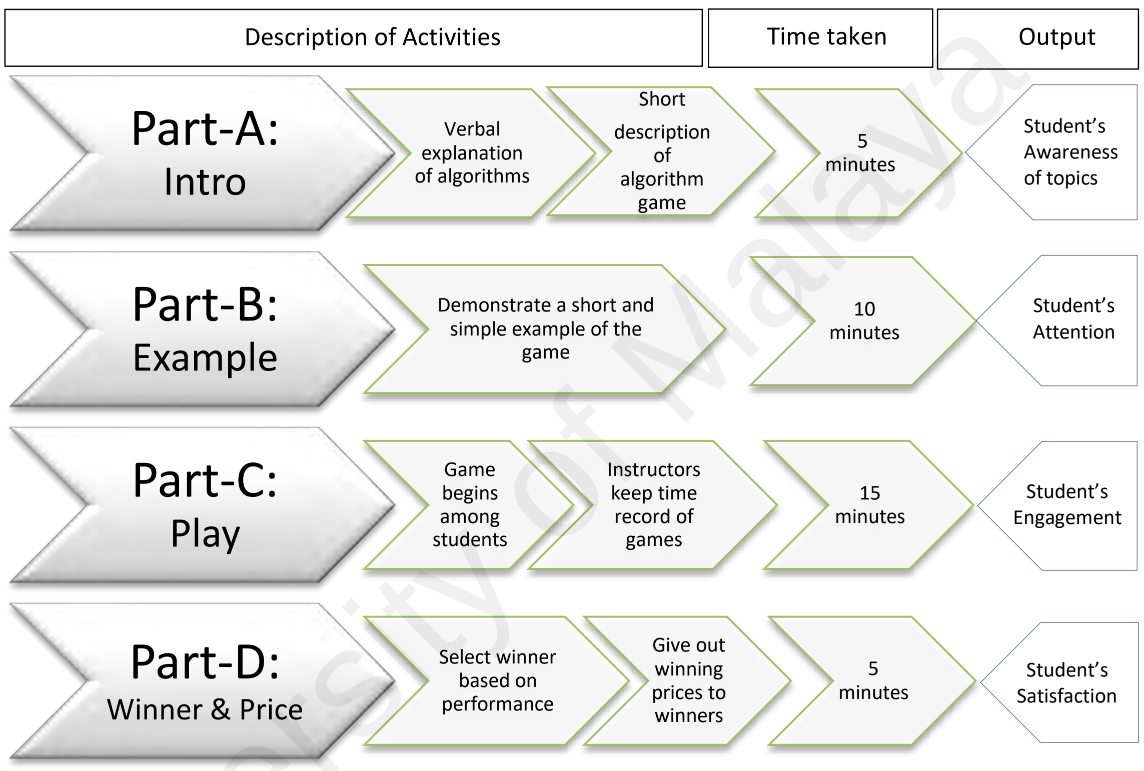
	Group 2016	Group 2017
Standard Deviation	1.72	1.71
Mean mark	5.55	8.13
Median mark	5.67	8.00

4.1.5.4 Learning Session Observation

A learning session on "Programming Basics" was arranged where 34 first-year students (admission year 2018) and 36 pre-university students participated. None of them had prior experience of participating in any programming class. Session consists of three parts. In the first part, a lecture on basic programming was delivered, a practical presentation of writing code in computer was exercised, and a pretest exercise was conducted for all 71 students. Then the students were divided into two groups namely GG (15 students, 15 students) and LG (19 students, 21 students). LG was asked to skip part two of the session and proceed to part three. In part two of the session, participants from GG had partaken in three Gaming Activities. Bubble Sort, Radix Sort, and Selection Sort activities were conducted among GG participants. All the above-mentioned gaming activity information can be found in Table A of Appendix A.

4.1.5.5 Game Procedure

Activity session was conducted in four parts (A, B, C and D). The first part (A) involved the verbal explanation to introduce and describe the algorithm involved and this takes approximately five minutes. Part A is to capture students' awareness of topics. The second part (B) involved a short demonstration of the competitive, physical and tactile games. Part B is to capture students' attention. This takes approximately 10 minutes. The third part (C) is the beginning of the actual game session among students. The instructors keep track of the time taken for each group to complete the game or just merely keeping record of which team completed the game first. Part C is to capture student's engagement. The last part (D) is to selector identify the winners of the game based on students' performance of the game. The winners were given acknowledgment of their achievement by distributing special prizes given. Figure 4.1 shows steps of the game sessions.



Description of Activities
Time taken
Output
Part-A: Intro
Verbal explanation of algorithms
Short description of algorithm game
5 minutes
Student's Awareness of topics
Part-B: Example
Demonstrate a short and simple example of the game
10 minutes
Student's Attention
Part-C: Play
Game begins among students
Instructors keep time record of games
15 minutes
Student's Engagement
Part-D: Winner & Price
Select winner based on performance
Give out winning prices to winners
5 minutes
Student's Satisfaction

Figure 4.1 Steps for game sessions

In the final part, both groups sat for the test. Table 4.4 shows the test result summary of the both groups. Complete result of the test results can be found in Table B(i) of appendix B.

Table 4.4 Comparative observation of mean, median and Standard Deviation.

Session	Group		Pretest	Posttest
Oct 2018	Gamming Group	N	15	15
		Mean	3.63	7.20
		Std. Deviation	2.408	4.161
		Median	3.00	7.00
	Lecture Group	N	19	19
		Mean	3.84	4.11
		Std. Deviation	1.756	2.622
		Median	3.50	3.00
Feb 2019	Gamming Group	N	14	15
		Mean	3.43	12.47
		Std. Deviation	2.593	7.090
		Median	2.50	17.00
	Lecture Group	N	21	21
		Mean	4.05	7.95
		Std. Deviation	1.949	5.054
		Median	3.50	6.00
Total	Gamming Group	N	29	30
		Mean	3.53	9.83
		Std. Deviation	2.457	6.309
		Median	3.00	8.00
	Lecture Group	N	40	40
		Mean	3.95	6.13
		Std. Deviation	1.839	4.479
		Median	3.50	4.00

An independent-samples t-test was conducted by using IBM SPSS Statistics application to compare test-marks for GG and LG. The t-test for equality of mean has shown below in Table 4.5.

Table 4.5 The t-test for equality of means

	t	df	Sig. (2-tailed) [p-value]	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Pretest	-0.77	49.51	0.446	-0.42	-1.502	0.671
Posttest	2.74	49.78	0.008	3.71	0.992	6.424

There was no significant difference in the scores for GG (M=3.53, SD=2.25) and LG (M=3.95, SD=1.84) conditions for pretest; t (50) = -0.77, p = 0.446. But There was a significant difference in the scores for GG (M=9.83, SD=6.31) and LG (M=6.13, SD=4.48) conditions for posttest; t (50) = 2.74, p = 0.008. These results suggest that gaming session really does have an effect on students' marks. Specifically, our results suggest that when students go through academic gaming activity, they score higher.

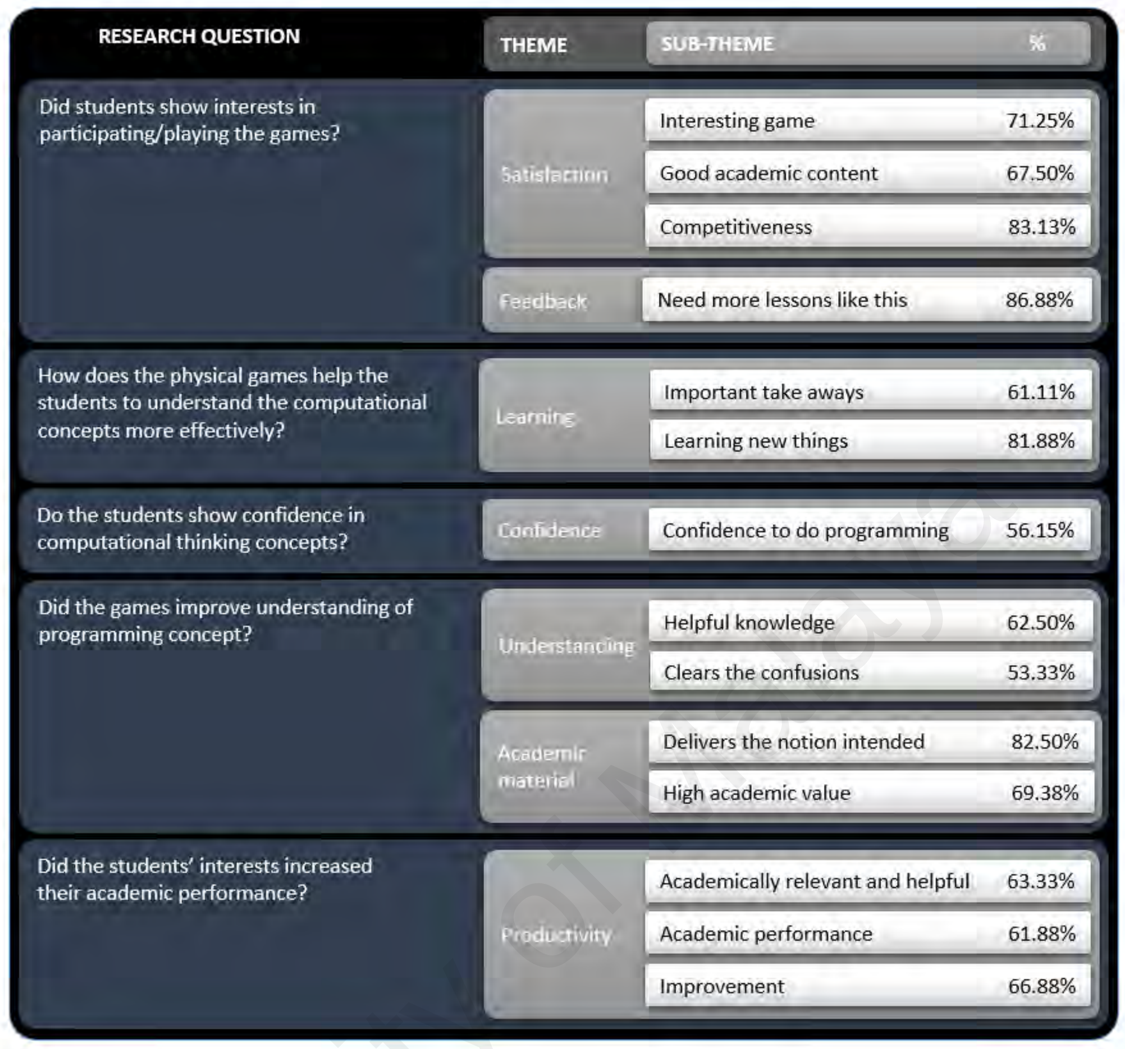
4.2 Validation

4.2.1 Game Activity Validation

First, all data from interviews were transferred into an Excel tabulation sheet. Some of the collected data were already in digital text format, while some were in audio/video/handwritten formats. The participants were asked to listen to/read transcriptions of their responses into digital text data. A follow-up validation process was conducted with 15 students who were representative of the study population to facilitate a member-check of the data.

4.3 Result

According to the data gathered from the activities conducted by the us, the students seemed enthusiastic when they were told that they would be playing an educational game. During their interaction with the game, they seemed very absorbed and interested in the task and exhibited high levels of engagement in their effort to maintain their competency toward winning the game or the completing it with high scores. Figure 4.2 shows the overall result of the interview through thematic analysis.



RESEARCH QUESTION	THEME	SUB-THEME	%
Did students show interests in participating/playing the games?	Satisfaction	Interesting game	71.25%
		Good academic content	67.50%
		Competitiveness	83.13%
	Feedback	Need more lessons like this	86.88%
How does the physical games help the students to understand the computational concepts more effectively?	Learning	Important take aways	61.11%
		Learning new things	81.88%
Do the students show confidence in computational thinking concepts?	Confidence	Confidence to do programming	56.15%
Did the games improve understanding of programming concept?	Understanding	Helpful knowledge	62.50%
		Clears the confusions	53.33%
	Academic material	Delivers the notion intended	82.50%
		High academic value	69.38%
Did the students' interests increased their academic performance?	Productivity	Academically relevant and helpful	63.33%
		Academic performance	61.88%
		Improvement	66.88%

Figure 4.2 Thematic Analysis with students' feedback

4.3.1 Students' Interest

Nearly 71.25% of the students reported that the game they were playing was "*really interesting*". One of the most essential part of this research is to study the feedback from students by which we can rate whether and how the participants benefited from these physical and tactile games. The interviewed participants (86.88%) pointed out that most of them loved and accepted this new method of teaching and are looking forward to more of similar activities on a range of different topics. Confidence to compete in CT exercises for solving a programming problem is one of the most vital things that are clearly noticeable among all participating students.

4.3.2 Confidence and Understanding

Around 56.15% of the participants thought that the games boosted their confidence their own programming abilities. Humans learn faster through experience by mapping their cognitive mind into reality or physicality (Bransford, Brown, & Cocking, 1999), which we aspired for in the present study. The majority of the students (67.50%) indicated conviction in understanding programming through games, cleared old confusions, and improved their understanding through game activities compared to learning only through slides (reported by 69.38% of the participants). Furthermore, they expect additional activities similar to the ones in this study for a wide range of topics in the future.

4.3.3 Learning Through Computational Thinking

It can be observed that most of interviewed participants (81.88% students) thought that the games boosted their confidence in programming and therefore improved their CT abilities as well. Although the issue of CT is a part of programming that has attracted considerable attention from the non-computer science community, the characteristics, practices, and perspectives have almost indisputably come from the analysis, development, and testing stages of software lifecycle (Brennan & Resnick, 2012; Lye & Koh, 2014). It is therefore acceptable from the viewpoint of the computer science community that programming practices improve CT or vice versa.

Most of the interviewed participants thought that the games were an effective method to cover certain topics in programming (82.50%) and important take ways (61.11%). The introduction of games in the classroom was thought of as a simple, short, step-by-step, and clear method for explaining sorting algorithms. The games inspired the participants to think about the set of rules and ignore irrelevant details. These findings are indicators of the presence of CT elements, as emphasised by Barr & Stephenson (2011) and Wing (2006), especially decomposition of complex tasks into simpler step-by-step solutions and manipulation of thinking through levels of abstractions by simplifying or ignoring details whenever appropriate. However, data on other elements of CT, such as pattern recognition, were not captured in the interview.

4.3.4 Academic Performance and Productivity

About 61.88% participants opined that their academic performance in the topics related to the games increased because of the games they played and 63.33% of the students found these games to be well related to academics and helpful for education. The exam results of the students in the group that played the competitive physical and tactile games (2017 batch) were better than those of the students in the group that did not play the games (2016 batch) in both the sorting and graph questions. This implies that the games had a positive effect on the learning outcomes and, consequently, the exam scores of the students. This is in contrast to the findings of a longitudinal study by Hanus & Fox (2015) .

- Longitudinal study on effects of gamification in the classroom.
- 71 students surveyed at four time points in gamified or non-gamified course.
- Over time, gamified students were less motivated, empowered, and satisfied.
- Gamified course negatively affected final exam grades through intrinsic motivation.
- Gamified systems strongly featuring rewards may have negative effects.

Most students (62.50%) admitted that the game concepts helped them understand the concepts and cleared their view of the problems (about 53.33%), as indicated by their exam results. These findings can be used to support efforts related to the use of active learning activities in classrooms for increasing academic performance (about 66.88%).

However, the competitive element applied in our study may be a crucial ingredient that should be included.

4.3.4.1 Comparing Groups

It has been observed that students enjoyed the class. The participants were more committed to finishing assignments in class (the game problem). Regarding the topic of sorting and graphs, the answer scripts had been analysed in the final exams and saw improvements in the students' examination scores. The students that were introduced to games scored higher marks compared to the previous batch in the course *WKES3311: Analysis of Algorithm* Class.

In Figure 4.3, the results of our analysis of students' academic records show that the average mark in the graph algorithms class in 2016 was around 60%, whereas in 2017

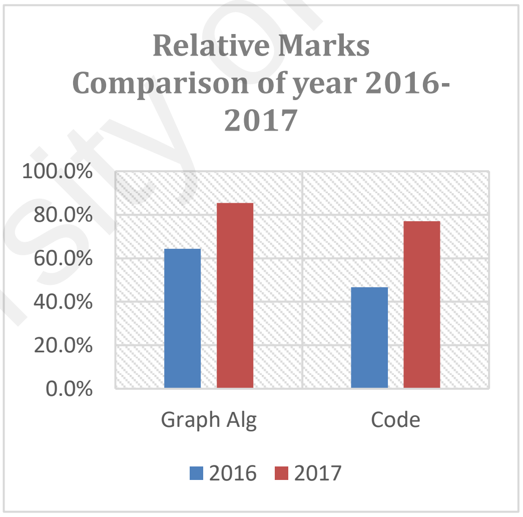


Figure 4.3 Average mark in final exam for the 2016 and 2017 student batches

batch the average mark was about 80%, representing an increase of 20% in the average mark. In the sorting codes class, the average mark was around 40% in 2016 and 80% in 2017, representing an increase of almost 40%. Figure 4.3 presents the relevant descriptive statistics.

4.4 Discussion

Six lesson games were designed and executed incorporating a scholastic approach to illustrate swapping; the sorting algorithm, that is, bubble sort, quicksort, and selection sort; and graph theory. The activities were conducted involving students from years one to four of a batch of undergraduate students. Thirteen student participants from WKES3311 2016/2017 semester-I managed to attain a noticeable change in their academic performance; 44 students from WIA2005, 21 students from Data Structure class 2016/2017 Semester-II, 9 students from extra-class semester-II, 36 first-year undergrad students that is, a total of 121 students, showed similar improvements. Our research questions were related to the participants' interests, understanding of the topic, CT, and academic performance.

It is been observed that most of the interviewed participants thought that the games boosted their confidence in programming and therefore improved their CT abilities as well. Although the issue of CT is a part of programming and that has attracted considerable attention from the non-computer science community as well as, the characteristics, practices, and perspectives have almost indisputably come from the analysis, development, and testing stages of software the lifecycle (Brennan & Resnick, 2012; Lye & Koh, 2014). It is therefore acceptable from the viewpoint of the computer science community that programming practices improve CT or vice versa.

Additionally, most of the interviewed participants thought that the games were an effective method to cover certain topics in programming. The introduction of games in the classroom was thought of as a simple, short, step-by-step, and clear method for explaining sorting algorithms. The games inspired the participants to think about the set of rules and ignore irrelevant details. These findings are indicators of the presence of CT elements, as emphasised by Barr & Stephenson (2011) and Wing (2006), especially decomposition of complex tasks into simpler step-by-step solutions and manipulation of thinking through levels of abstractions by simplifying or ignoring details whenever appropriate. However, data on other elements of CT, such as pattern recognition, were not captured in the interview.

The interviewed participants (81.88% of them) thought they learned new things, new skills, and algorithms, and that they were getting direct knowledge transfer. This indicates that the students not only learned new topics, but they were stimulated to acquire certain skills that should be thought of as CT skills.

4.5 Summary

In this chapter, the entire qualitative preliminary study has been done to address the first research question of the thesis. Some of responses were just above 50%, it means those students are sure that they have improved their understanding and grow confidence in those topics. And what it does not mean for the rest of the participant is that it did not cause anything to lose their confidence or understanding in those topics. From this chapter it has been demonstrated that how physical and tactile games are interesting and helpful for the students by comparing their feedback and academic performances.

CHAPTER 5: DESIGN AND IMPLEMENTATION

In this chapter, the prototype design and implementation of the tool is expanded in two sections.

5.1 Tool Design

The prototype design is elaborated through Model design, Formula design, System design, Interface and component design, User accessibility scope

5.1.1 Model Design

First, the user code is analyzed into keywords and syntax. These keywords, syntax, and output of the program is used to find the attributes for the CT elements. List of parameters of all layers are shown in Table 5.1.

Table 5.1 Keyword-syntax and attribute table for CT elements

Keyword-syntax analysis	Attributes	CT elements
Keywords Syntax	Function definitions Function calls Loops Comparisons Conditions Assignments Statements	Decomposition Pattern Recognition Pattern generalization/Abstraction Algorithm

The model diagram of CT elements from coding is shown below in Figure 5.1,

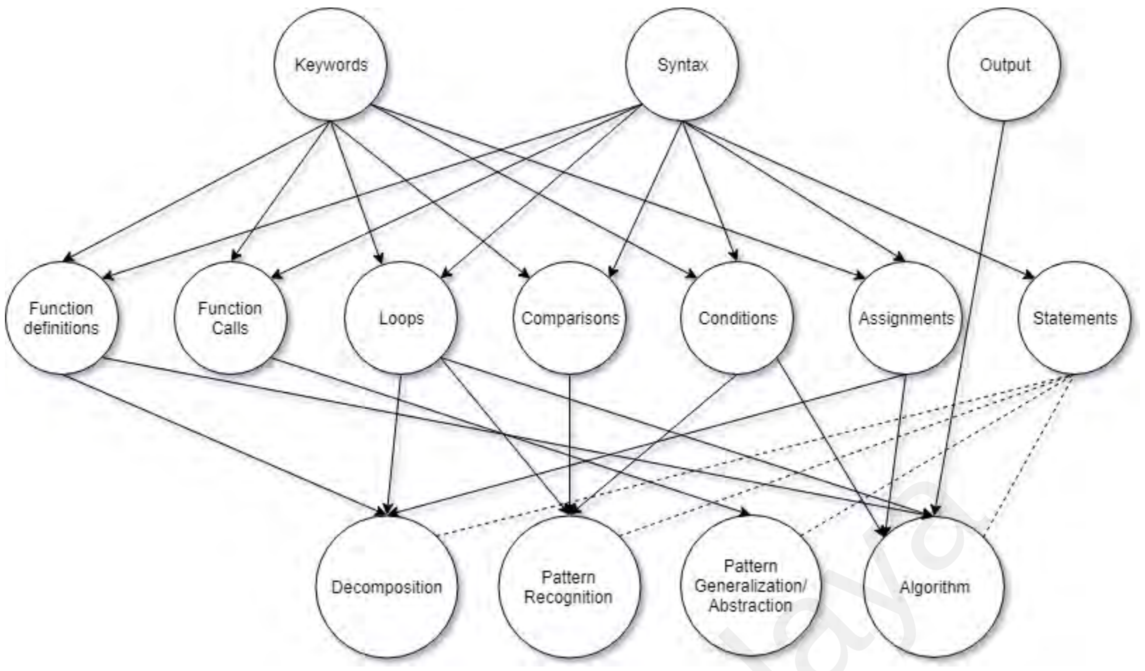


Figure 5.1 Modeling CT elements

5.1.2 Formula Design

The formula was designed for CT element identification and activity suggestions. The CT elements was obtained from python program, the calculations have been done in three layers.

5.1.2.1 Layer 1

In the first layer of calculation, the keywords and syntaxes are obtained by means of regular expression analysis of the code typed by the users. Here, keywords include all the python keywords.

$$Keywords \Rightarrow \{False, class, finally, is, return, None, continue, for, lambda, try, True, def, from, nonlocal, while, and, del, global, not, with, as, elif, if, or, yield, assert, else, import, pass, break, except, in, raise\};$$

And syntax includes mathematical operators, python operators, special operators. i.e. "",+,-,*,/, =, <, (), %, !, [] etc. PHP built-in functions were used to facilitate these calculations.

$$Syntaxes \Rightarrow \{(python\ operators), (mathematical\ operators), (special\ operators)\}$$

Output are obtained from online compiler that is directly linked to the code canvas module.

5.1.2.2 Layer 2

The results, obtained in layer one, are classified into seven parameters based on standard programming convention and python code standard. Parameters are function definitions, function calls, loops, comparisons, conditions, assignments, statements. Pseudocodes are given below,

```
$Function\ calls \Rightarrow counts\ the\ number\ of\ function\ calls$
$Function\ definitions \Rightarrow counts\ the\ number\ of\ function\ definitions$
$Statements \Rightarrow counts\ the\ total\ valid\ statements$
$Loops \Rightarrow counts\ number\ of\ loops$
$Assignments \Rightarrow counts\ the\ number\ of\ assignments\ to\ the\ variables$
$Comparisons \Rightarrow counts\ the\ number\ of\ comparisons\ has\ been\ done$
$Conditions \Rightarrow counts\ how\ many\ times\ the\ program\ checks\ the\ conditions$
$Runs \Rightarrow determines\ the\ output\ of\ the\ program\ through\ online\ compiler$
```

5.1.2.3 Layer 3

Following that, in third layer of calculation, CT elements are estimated in percentage on the basis of total valid statements by associating the parameters found in layer two. Following pseudocodes shows the computation of the CT elements.

```
$Decomposition \Rightarrow [\{(keywords), (assignments)\} : \{(valid\ statements)\}] * 100$
```

$$Pattern\ recognition \Rightarrow [\{(loops), (conditions), (comparisons)\}$$
$$: \{(valid\ statements)\}] * 100$$

$$Abstraction \Rightarrow [\{(library\ function\ calls), (user\ function\ calls)\}$$
$$: \{(valid\ statements)\}] * 100$$

$$Algorithm$$
$$\Rightarrow [(output\ of\ the\ program)\{(function\ definitions), (loops), (conditions)\}$$
$$: \{(valid\ statements)\}] * 100$$

5.1.3 Activity Suggestion Formula

This calculation has been done in two steps.

5.1.3.1 Step-1

Each activity has its four CT components. Based on these components, the tool generates four sorted lists of available activities in descending order. Pseudocodes are given below,

```
$Decomposition\_list$
        $= sort\ all\ the\ activities\ in\ descending\ order\ based\ on\ decomposition\ value$

$Pattern\_recognition\_list$
$= sort\ all\ the\ activities\ in\ descending\ order\ based\ on\ pattern\ recognition\ value$

$Abstraction\_list = sort\ all\ the\ activities\ in\ descending\ order\ based\ on\ abstraction\ value$

$Algorithm\_list = sort\ all\ the\ activities\ in\ descending\ order\ based\ on\ algorithm\ value$
```

5.1.3.2 Step-2

In step-2, the system calculates four mean value of the CT elements of a particular class/group from their corresponding program code. If expected mean value is less than class/group mean value, then activity lists obtained instep-1 are suggested for each of the respective CT elements. Here, default expected mean value is 50% for each of the CT elements. This expected mean value depends on the instructors' decision for a particular class. The four CT means are denoted as decomposition mean (DM), pattern recognition mean (PM), abstraction mean (AM), algorithm mean (AlgM) in the following pseudocode,

```
$Class\ DM\ =\ average\ decomposition\ values\ obtained\ from\ a\ class$
$Class\ PM\ =\ average\ pattern\ recognition\ values\ obtained\ from\ a\ class$
$Class\ AM\ =\ average\ abstraction\ values\ obtained\ from\ a\ class$
$Class\ AlgM\ =\ average\ algorithm\ values\ obtained\ from\ a\ class$

$If\ (class\ DM\ <\ expected\ DM)\ suggest\ Decomposition\_list$
$If\ (class\ PM\ <\ expected\ PM)\ suggest\ Pattern\_recognition\_list$
$If\ (class\ AM\ <\ expected\ AM)\ suggest\ Abstraction\_list$
$If\ (class\ AlgM\ <\ expected\ AlgM)\ suggest\ Algorithm\_list$
```

Bases on above formulas, suggestions are displayed in the teachers' window.

5.1.4 System Design

This chapter starts with the architectural design of Code Analyzer tool, the functional modules, interface, and Use Case diagram. Finally, the implementation and algorithm are highlighted according to the designed components.

5.1.4.1 Use Case

There are two different roles that can be performed in the tool. In Figure 5.2 user scope has been illustrated.

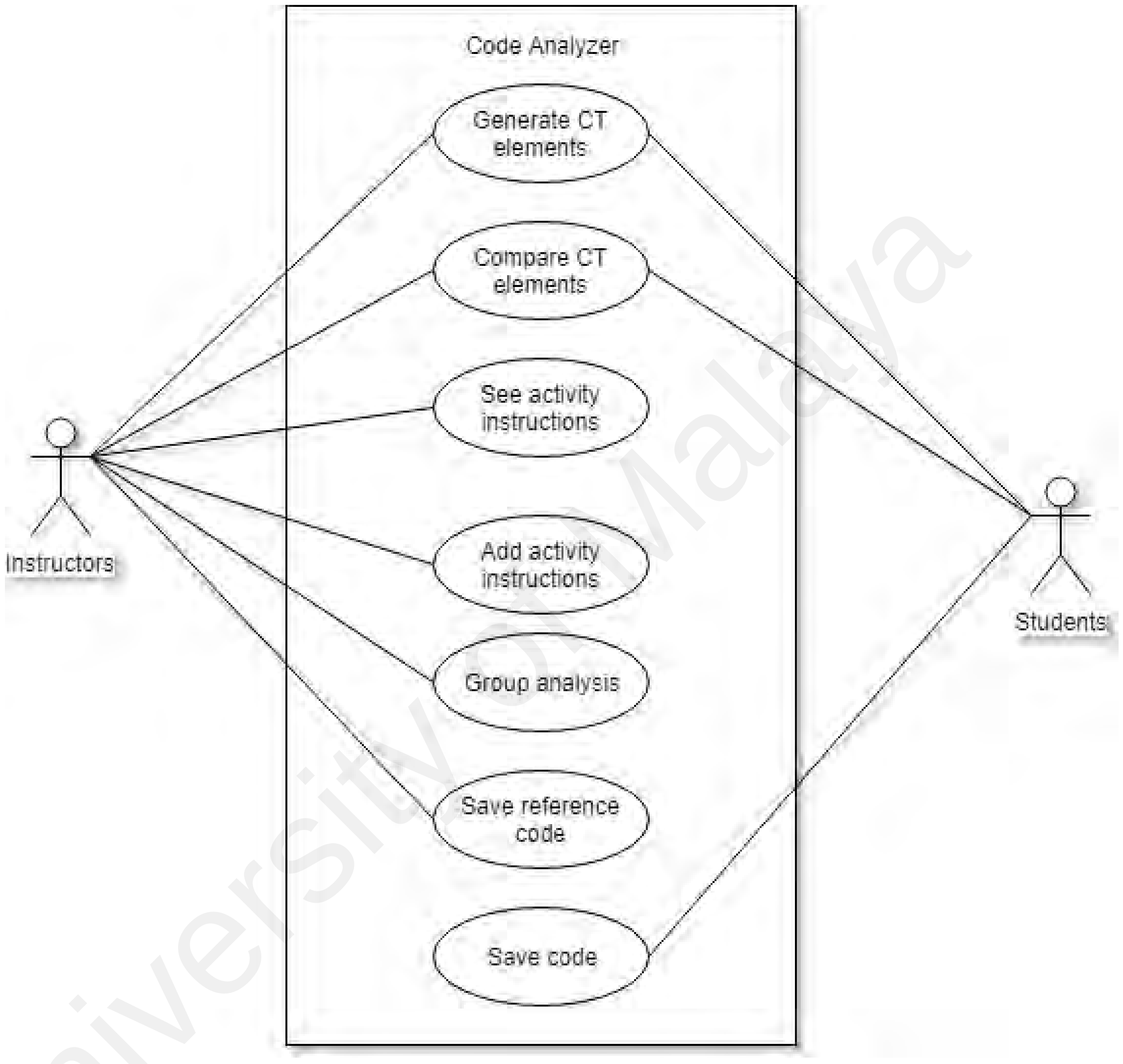


Figure 5.2 Use Case diagram for instructors and students

5.1.4.2 System Flowchart

The following Figure 5.3 illustrates flowchart of the tool. It starts with code canvas where students can save their code according to their associated group/class name.

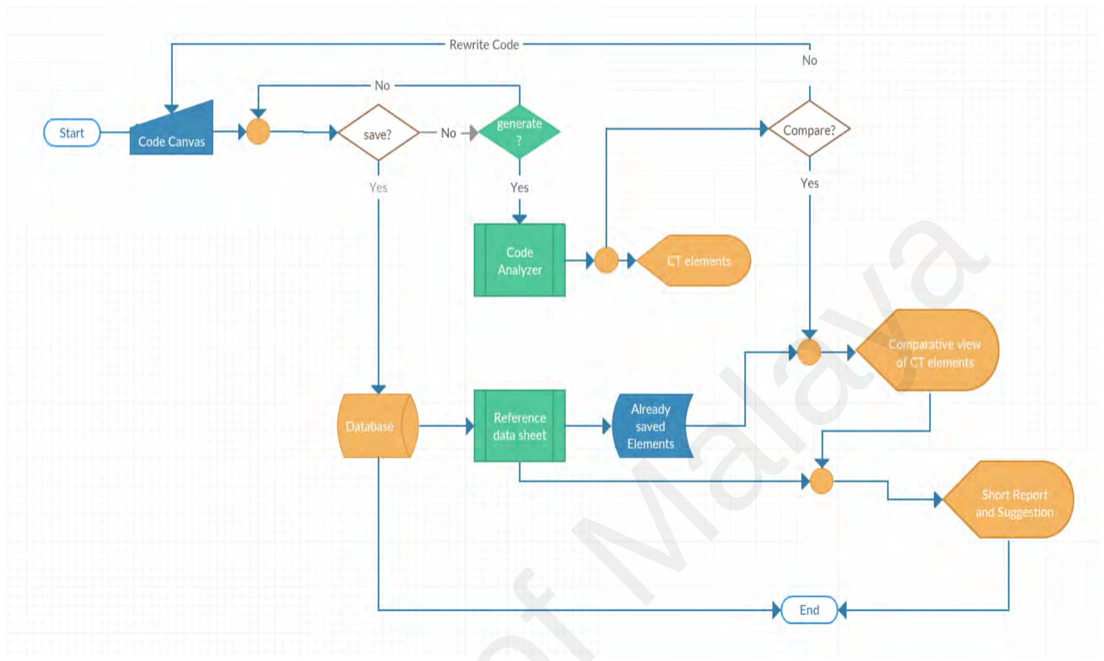
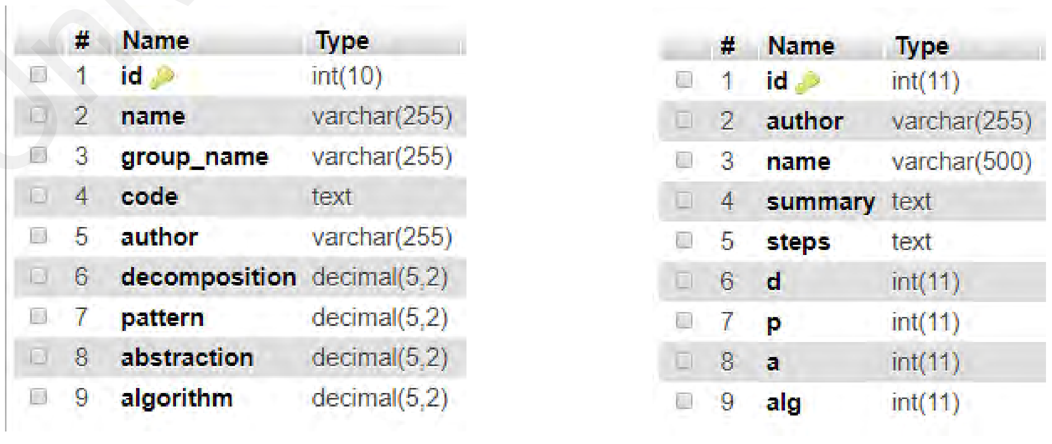


Figure 5.3 System flowchart for Code Analyzer

5.1.4.3 Database Overview

Tt requires a database containing two tables to manage all the data. Namely 'activities' and 'reference_sheet'. They are illustrated below in Figure 5.4,



#	Name	Type
1	id	int(10)
2	name	varchar(255)
3	group_name	varchar(255)
4	code	text
5	author	varchar(255)
6	decomposition	decimal(5,2)
7	pattern	decimal(5,2)
8	abstraction	decimal(5,2)
9	algorithm	decimal(5,2)

(a)

#	Name	Type
1	id	int(11)
2	author	varchar(255)
3	name	varchar(500)
4	summary	text
5	steps	text
6	d	int(11)
7	p	int(11)
8	a	int(11)
9	alg	int(11)

(b)

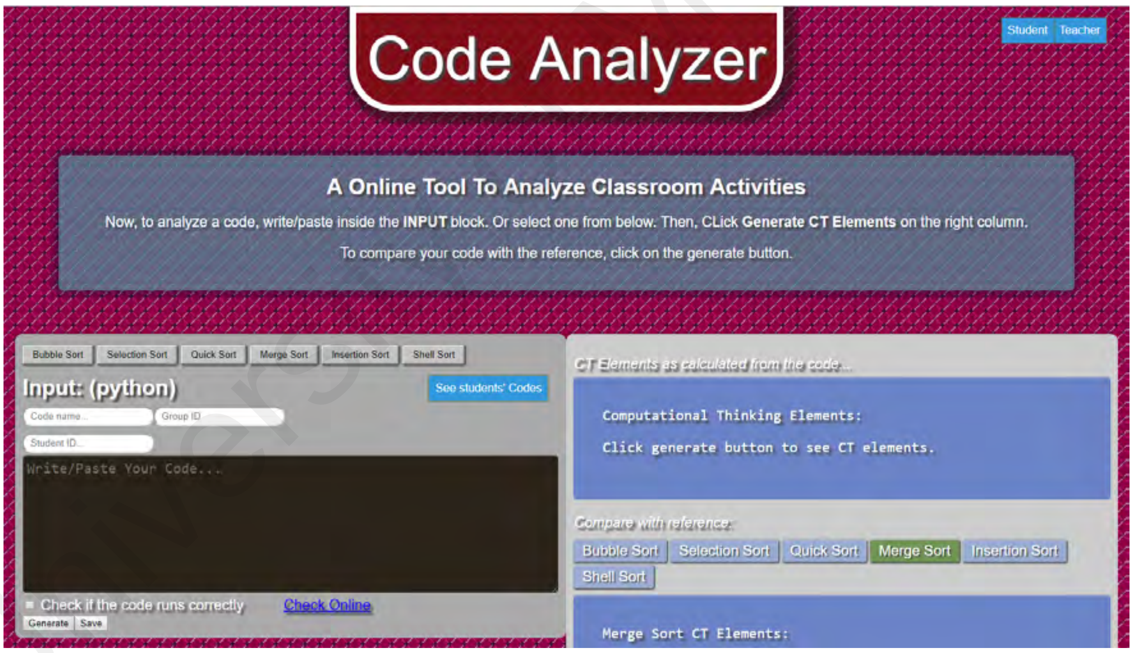
Figure 5.4 Table structures of (a) reference_sheet and (b) activities

In ‘reference_sheet’, program code, name of the program, author, group name, and CT elements are saved in eight columns. If author is a teacher, code will be saved as reference in this table of the database, as student code otherwise.

In the ‘activities’ table, instructions for classroom activities, author of the activity, activity name, short description of the activity along with four corresponding CT elements are saved in eight different columns.

5.1.5 Interface and Component Design

Figure 5.5 shows the complete view of the web application in which the tool was implemented in the backend.



A Online Tool To Analyze Classroom Activities

Now, to analyze a code, write/paste inside the **INPUT** block. Or select one from below. Then, CLick **Generate CT Elements** on the right column.

To compare your code with the reference, click on the generate button.

```
Computational Thinking Elements:
Click generate button to see CT elements.
```

```
Merge Sort CT Elements:
```

Figure 5.5 Front view of the tool

This tool has three basic components.

5.1.5.1 User Interface Component

It has two level of access. Level 1 for teachers and level 2 is for students. User interface consists of three subcomponents.

- Code canvas where user may write python codes. [Figure 5.6]

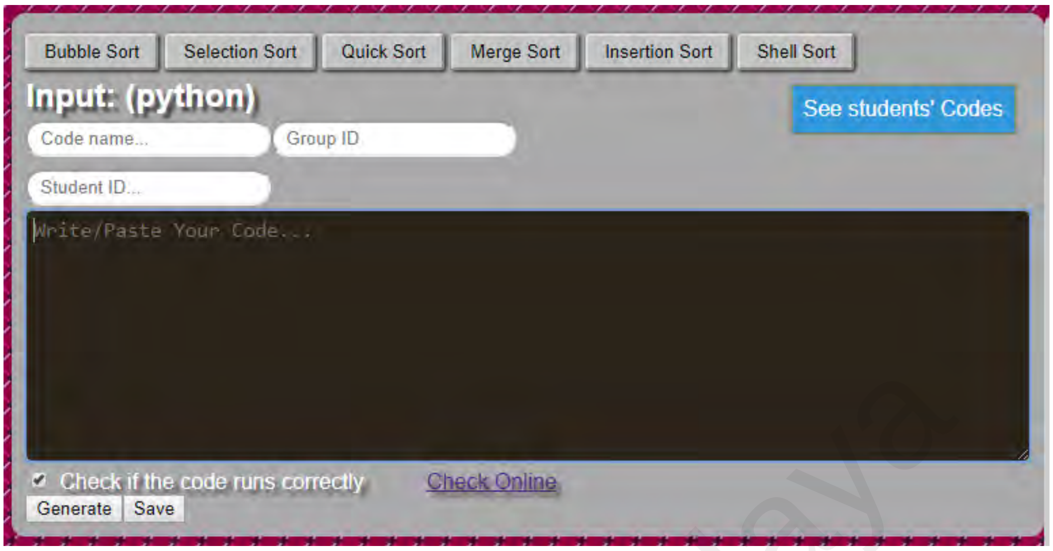
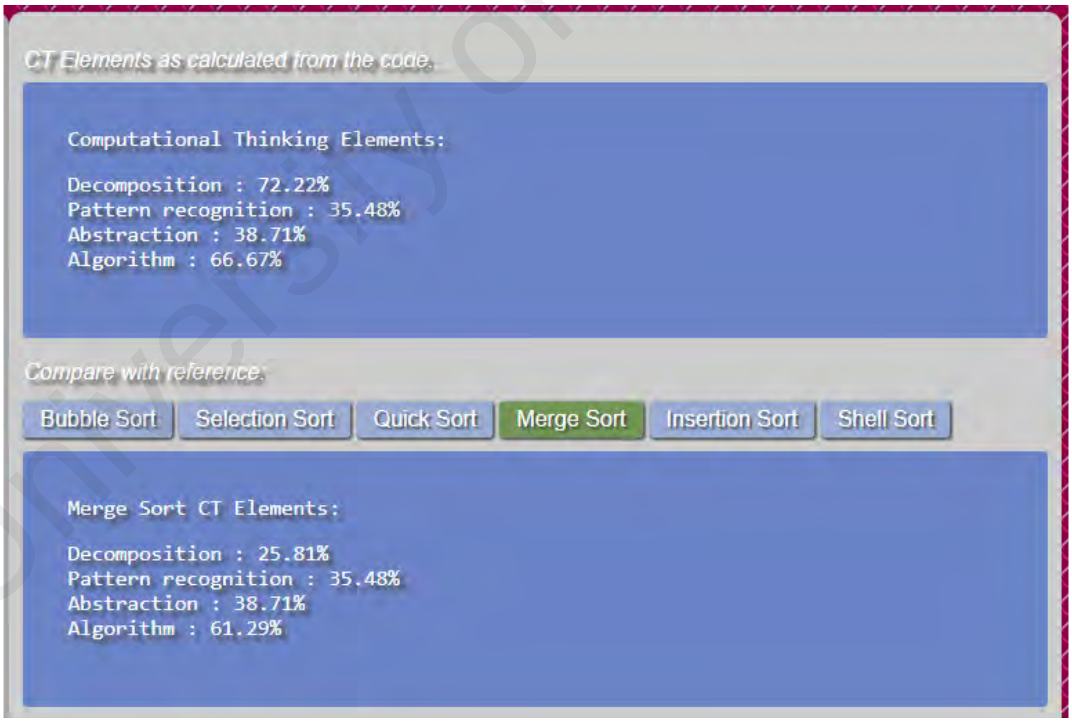


Figure 5.6 Code canvas

- CT elements view. [Figure 5.7]



```
Computational Thinking Elements:

Decomposition : 72.22%
Pattern recognition : 35.48%
Abstraction : 38.71%
Algorithm : 66.67%
```

```
Merge Sort CT Elements:

Decomposition : 25.81%
Pattern recognition : 35.48%
Abstraction : 38.71%
Algorithm : 61.29%
```

Figure 5.7 CT elements view and compare window

- Activity Suggestion, available for instructors only, where instructors can add new classroom activity and/or get classroom activity suggestion based of class performance. [Figure 5.8]

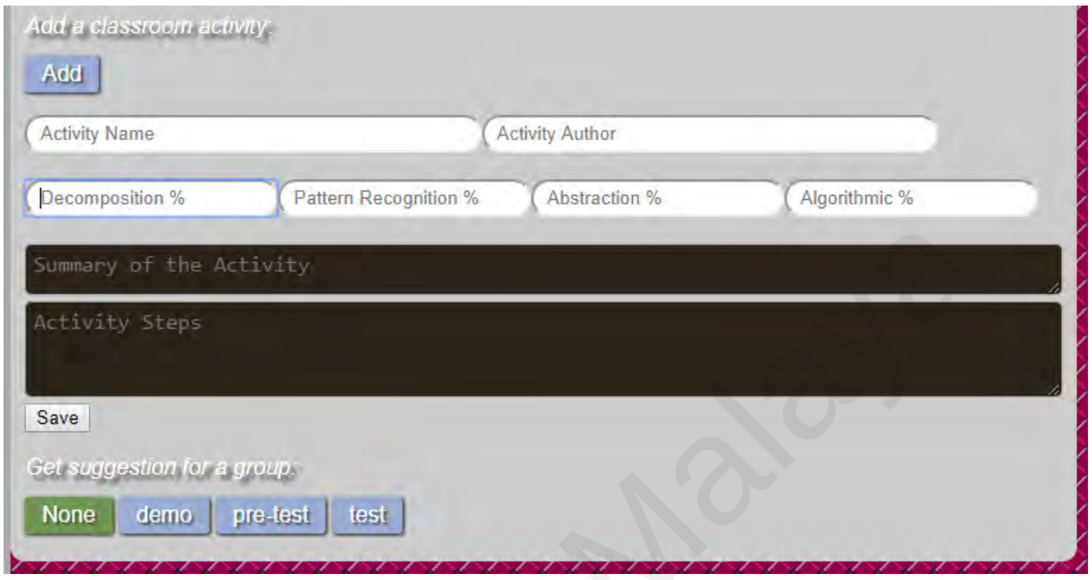


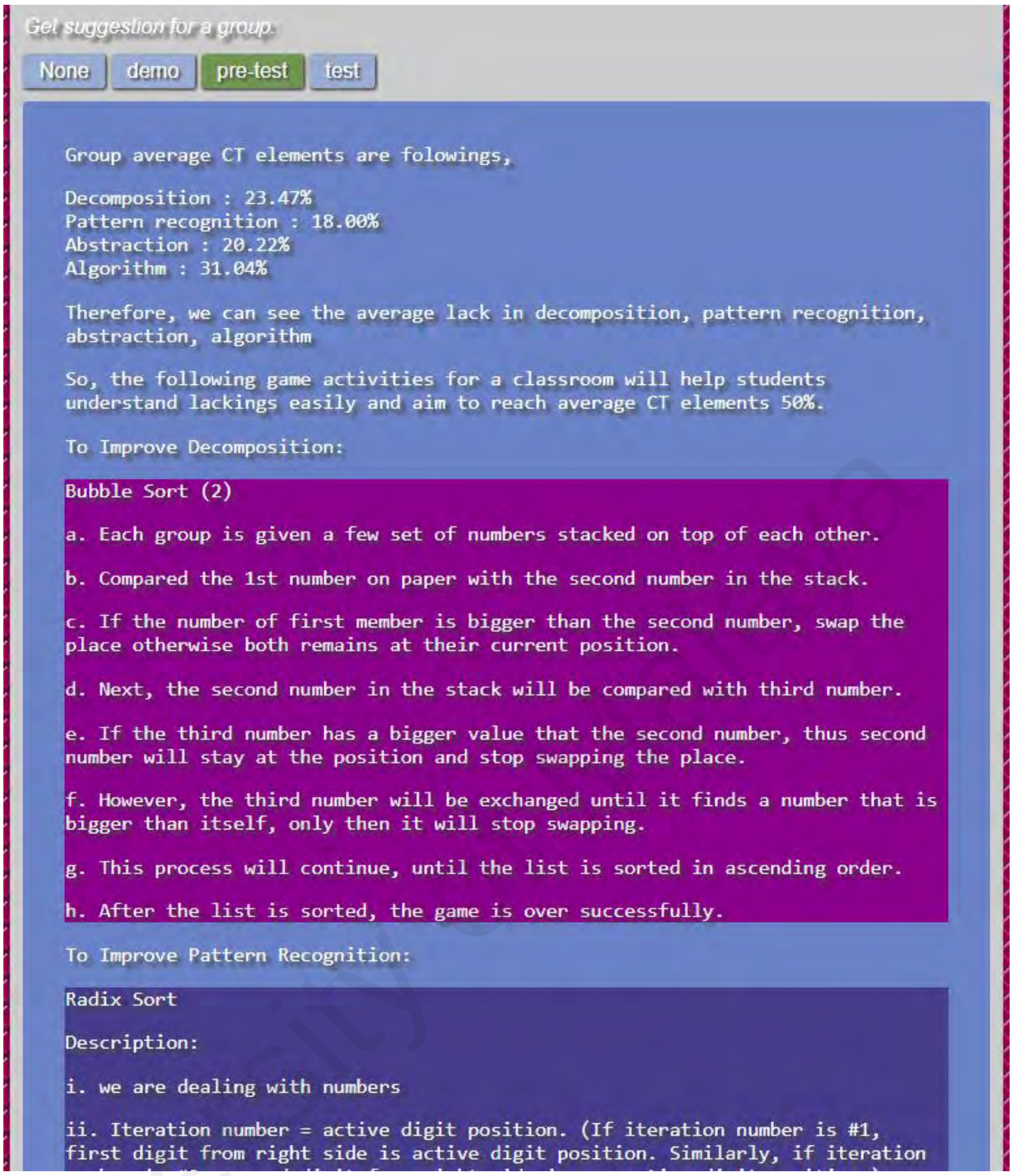
Figure 5.8 Add new activity window and Group Suggestion tabs (instructors only)

5.1.5.2 CT Elements Generator Component

It processes code based on keywords, functions, loops, conditions, and statements used inside code and generates four CT elements in percentage value.

5.1.5.3 Suggestion Window Component

In this section, teachers/instructors can find Classroom Activity Suggestions with Instructions. Shown in Figure 5.9.



Get suggestion for a group:

None demo pre-test test

```
Group average CT elements are folowings,

Decomposition : 23.47%
Pattern recognition : 18.00%
Abstraction : 20.22%
Algorithm : 31.04%

Therefore, we can see the average lack in decomposition, pattern recognition,
abstraction, algorithm

So, the following game activities for a classroom will help students
understand lackings easily and aim to reach average CT elements 50%.

To Improve Decomposition:

Bubble Sort (2)

a. Each group is given a few set of numbers stacked on top of each other.

b. Compared the 1st number on paper with the second number in the stack.

c. If the number of first member is bigger than the second number, swap the
place otherwise both remains at their current position.

d. Next, the second number in the stack will be compared with third number.

e. If the third number has a bigger value that the second number, thus second
number will stay at the position and stop swapping the place.

f. However, the third number will be exchanged until it finds a number that is
bigger than itself, only then it will stop swapping.

g. This process will continue, until the list is sorted in ascending order.

h. After the list is sorted, the game is over successfully.

To Improve Pattern Recognition:

Radix Sort

Description:

i. we are dealing with numbers

ii. Iteration number = active digit position. (If iteration number is #1,
first digit from right side is active digit position. Similarly, if iteration
```

Figure 5.9 Screenshot of the Code Analyzer showing the average CT elements and Suggested activities

5.1.5.4 Controller Component

Controller component has 4 major functionalities. They have been listed as follows:

Map CT Elements: For any given python code, it can map CT elements value in percentage.

Compare: Seven python programs have been saved in the database as reference in order to compare with students' codes.

Save: Students can save their program and fetch it later.

Make Suggestion: Based on the correlation of CT elements between reference sheet and students code it can suggest possible game activities. It includes instructions and steps on how to conduct/play the games.

5.1.6 User Accessibility Scope

It has accessibility scopes for instructors as well as for students. It has been demonstrated in Figure 5.2 using a Use Case diagram.

5.2 Implementation

This section illustrates the implementation of the prototype in following subsections.

5.2.1 Tools

The list of tools was selected based on development of the prototype's need. The list is shown in Table 5.2.

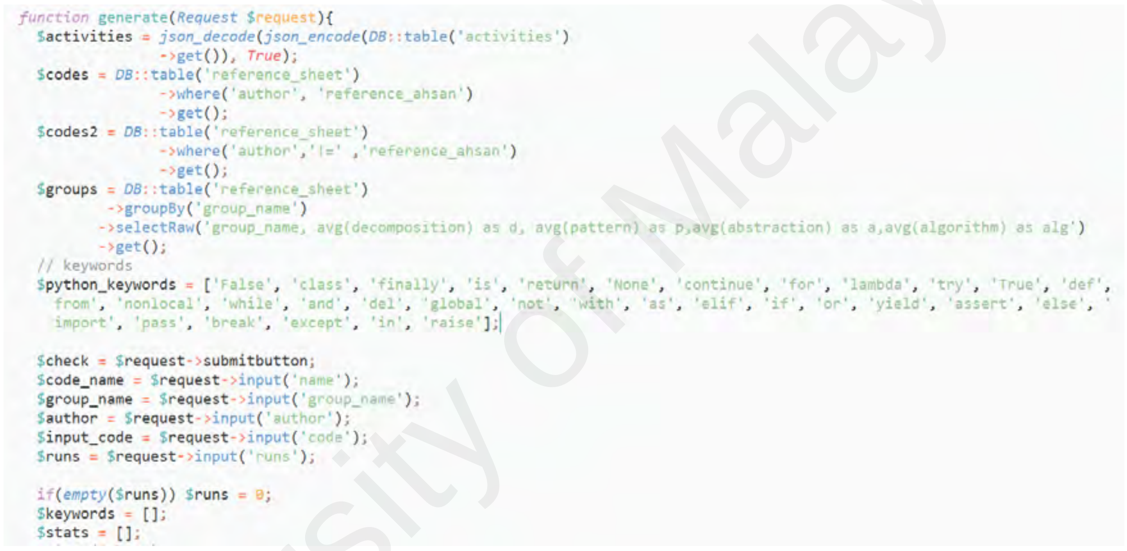
Table 5.2 List of all tools that is used in the development of the prototype

Name	Value	Reason for selection
Database	MySQL	Opensource, free
Programming Language	PHP 7.2	Agile, OS independent
Backend Framework	Laravel v5.6	Mature MVC framework with adequate documentation
Frontend technology	HTML, CSS, JavaScript	Platform independent (browser)
Server	Apache	Light weight

5.2.2 Development

Development has been done by using above mentioned tools. The following figures shows the backend modules of the prototype that has been designed in design section of this chapter.

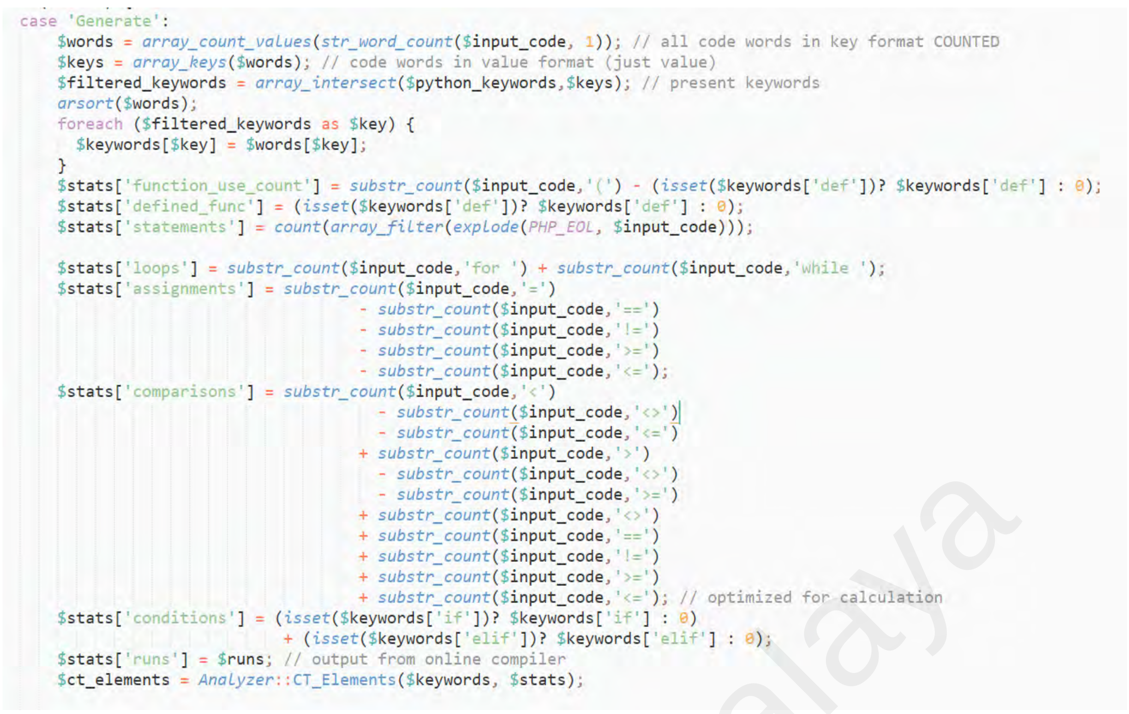
Here, in Figure 5.10, in the generate module, requested code and other information from the code canvas are received and keywords are initialized at this module. The relevant pseudocode can be found in Section 5.1.2.



```
function generate(Request $request){
  $activities = json_decode(json_encode(DB::table('activities')
                ->get()), True);
  $codes = DB::table('reference_sheet')
                ->where('author', 'reference_ahsan')
                ->get();
  $codes2 = DB::table('reference_sheet')
                ->where('author','!=' ,'reference_ahsan')
                ->get();
  $groups = DB::table('reference_sheet')
          ->groupBy('group_name')
         ->selectRaw('group_name, avg(decomposition) as d, avg(pattern) as p,avg(abstraction) as a,avg(algorithm) as alg')
         ->get();
  // keywords
  $python_keywords = ['False', 'class', 'finally', 'is', 'return', 'None', 'continue', 'for', 'lambda', 'try', 'True', 'def', '
    from', 'nonlocal', 'while', 'and', 'del', 'global', 'not', 'with', 'as', 'elif', 'if', 'or', 'yield', 'assert', 'else', '
    import', 'pass', 'break', 'except', 'in', 'raise'];

  $check = $request->submitbutton;
  $code_name = $request->input('name');
  $group_name = $request->input('group_name');
  $author = $request->input('author');
  $input_code = $request->input('code');
  $runs = $request->input('runs');

  if(empty($runs)) $runs = 0;
  $keywords = [];
  $stats = [];
```

Figure 5.10 Screenshot of the 'generate' module of the prototype

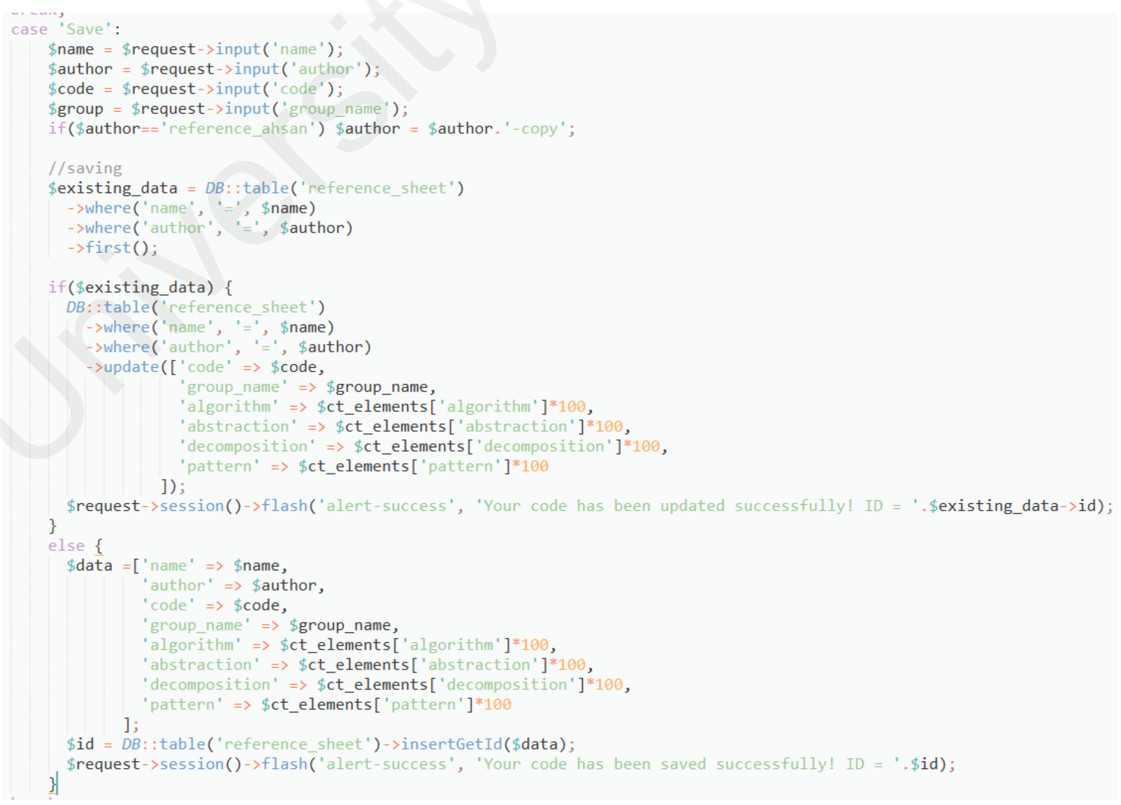
Figure 5.11 shows the association of elements between first layer and second layer.



```
case 'Generate':
    $words = array_count_values(str_word_count($input_code, 1)); // all code words in key format COUNTED
    $keys = array_keys($words); // code words in value format (just value)
    $filtered_keywords = array_intersect($python_keywords,$keys); // present keywords
    arsort($words);
    foreach ($filtered_keywords as $key) {
      $keywords[$key] = $words[$key];
    }
    $stats['function_use_count'] = substr_count($input_code,'(') - (isset($keywords['def'])? $keywords['def'] : 0);
    $stats['defined_func'] = (isset($keywords['def'])? $keywords['def'] : 0);
    $stats['statements'] = count(array_filter(explode(PHP_EOL, $input_code)));

    $stats['loops'] = substr_count($input_code,'for ') + substr_count($input_code,'while ');
    $stats['assignments'] = substr_count($input_code,'=')
                                - substr_count($input_code,'==')
                                - substr_count($input_code,'!=')
                                - substr_count($input_code,'>=')
                                - substr_count($input_code,'<=');
    $stats['comparisons'] = substr_count($input_code,'<')
                                  - substr_count($input_code,'<>')
                                  - substr_count($input_code,'<=')
                                + substr_count($input_code,'>')
                                  - substr_count($input_code,'<>')
                                  - substr_count($input_code,'>=')
                                + substr_count($input_code,'<>')
                                + substr_count($input_code,'==')
                                + substr_count($input_code,'!=')
                                + substr_count($input_code,'>=')
                                + substr_count($input_code,'<='); // optimized for calculation
    $stats['conditions'] = (isset($keywords['if'])? $keywords['if'] : 0)
                            + (isset($keywords['elif'])? $keywords['elif'] : 0);
    $stats['runs'] = $runs; // output from online compiler
    $ct_elements = Analyzer::CT_Elements($keywords, $stats);
```

Figure 5.11 Screenshot of association of second layer with first layer

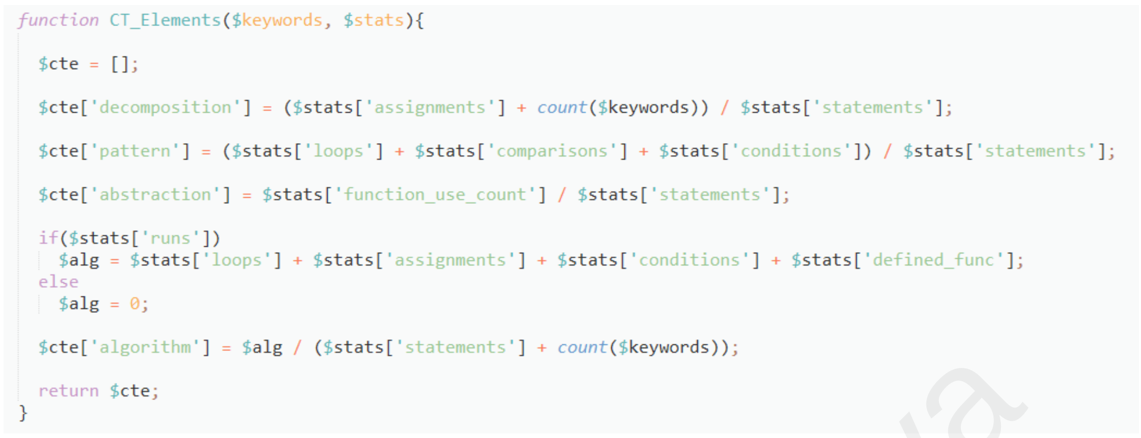
Saving the generated CT score into the database has been performed in 'save' module. [shown in Figure 5.12].



```
case 'Save':
    $name = $request->input('name');
    $author = $request->input('author');
    $code = $request->input('code');
    $group = $request->input('group_name');
    if($author=='reference_ahsan') $author = $author.'-copy';

    //saving
    $existing_data = DB::table('reference_sheet')
      ->where('name', '=', $name)
      ->where('author', '=', $author)
      ->first();

    if($existing_data) {
      DB::table('reference_sheet')
        ->where('name', '=', $name)
        ->where('author', '=', $author)
        ->update(['code' => $code,
                  'group_name' => $group_name,
                  'algorithm' => $ct_elements['algorithm']*100,
                  'abstraction' => $ct_elements['abstraction']*100,
                  'decomposition' => $ct_elements['decomposition']*100,
                  'pattern' => $ct_elements['pattern']*100
                ]);
      $request->session()->flash('alert-success', 'Your code has been updated successfully! ID = '.$existing_data->id);
    }
    else {
      $data =['name' => $name,
              'author' => $author,
              'code' => $code,
              'group_name' => $group_name,
              'algorithm' => $ct_elements['algorithm']*100,
              'abstraction' => $ct_elements['abstraction']*100,
              'decomposition' => $ct_elements['decomposition']*100,
              'pattern' => $ct_elements['pattern']*100
            ];
      $id = DB::table('reference_sheet')->insertGetId($data);
      $request->session()->flash('alert-success', 'Your code has been saved successfully! ID = '.$id);
    }
```

Figure 5.12 Screenshot of the 'saving CT elements' module

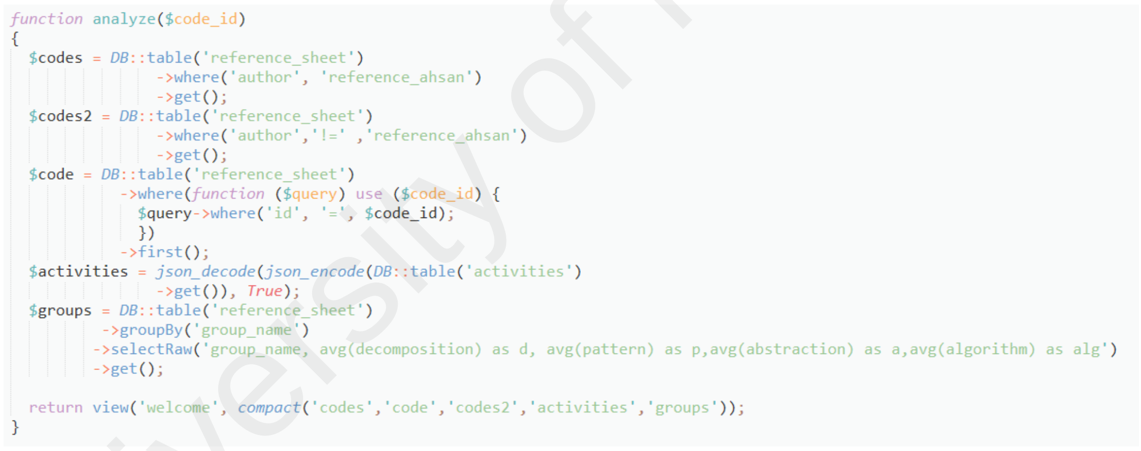
In Figure 5.13, the association of third layer and second layer has been shown.



```
function CT_Elements($keywords, $stats){

  $cte = [];

  $cte['decomposition'] = ($stats['assignments'] + count($keywords)) / $stats['statements'];

  $cte['pattern'] = ($stats['loops'] + $stats['comparisons'] + $stats['conditions']) / $stats['statements'];

  $cte['abstraction'] = $stats['function_use_count'] / $stats['statements'];

  if($stats['runs'])
    $alg = $stats['loops'] + $stats['assignments'] + $stats['conditions'] + $stats['defined_func'];
  else
    $alg = 0;

  $cte['algorithm'] = $alg / ($stats['statements'] + count($keywords));

  return $cte;
}
```

Figure 5.13 Screenshot of association of third layer with second layer

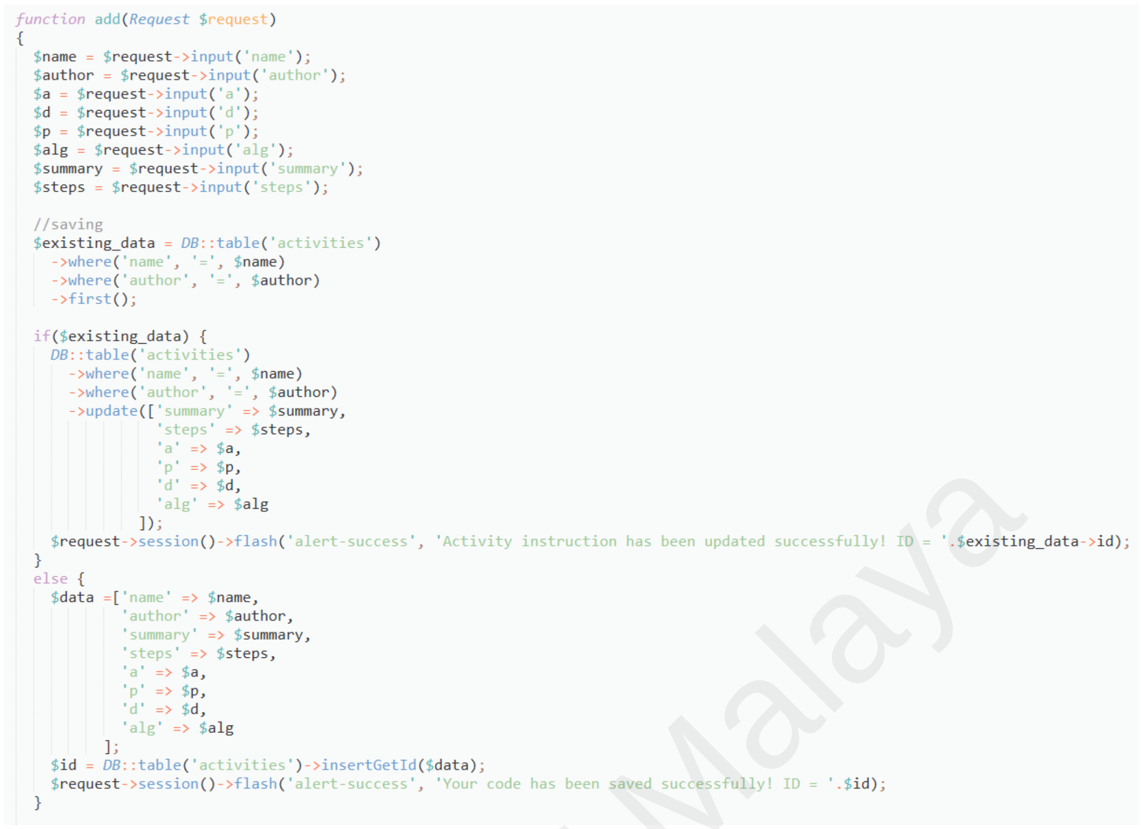
The code canvas being connected with other modules and database are shown in Figure 5.14.



```
function analyze($code_id)
{
  $codes = DB::table('reference_sheet')
              ->where('author', 'reference_ahsan')
              ->get();
  $codes2 = DB::table('reference_sheet')
              ->where('author','!=' ,'reference_ahsan')
              ->get();
  $code = DB::table('reference_sheet')
          ->where(function ($query) use ($code_id) {
            $query->where('id', '=', $code_id);
            })
          ->first();
  $activities = json_decode(json_encode(DB::table('activities')
              ->get()), True);
  $groups = DB::table('reference_sheet')
         ->groupBy('group_name')
        ->selectRaw('group_name, avg(decomposition) as d, avg(pattern) as p,avg(abstraction) as a,avg(algorithm) as alg')
        ->get();

  return view('welcome', compact('codes','code','codes2','activities','groups'));
}
```

Figure 5.14 Screenshot of the ‘analyze’ module of the prototype

Lastly, the activity addition is handled by the ‘add’ module show in the Figure 5.15 next page.



```
function add(Request $request)
{
  $name = $request->input('name');
  $author = $request->input('author');
  $a = $request->input('a');
  $d = $request->input('d');
  $p = $request->input('p');
  $alg = $request->input('alg');
  $summary = $request->input('summary');
  $steps = $request->input('steps');

  //saving
  $existing_data = DB::table('activities')
    ->where('name', '=', $name)
    ->where('author', '=', $author)
    ->first();

  if($existing_data) {
    DB::table('activities')
      ->where('name', '=', $name)
      ->where('author', '=', $author)
      ->update(['summary' => $summary,
              'steps' => $steps,
              'a' => $a,
              'p' => $p,
              'd' => $d,
              'alg' => $alg
          ]);
    $request->session()->flash('alert-success', 'Activity instruction has been updated successfully! ID = '.$existing_data->id);
  }
  else {
    $data =['name' => $name,
          'author' => $author,
          'summary' => $summary,
          'steps' => $steps,
          'a' => $a,
          'p' => $p,
          'd' => $d,
          'alg' => $alg
        ];
    $id = DB::table('activities')->insertGetId($data);
    $request->session()->flash('alert-success', 'Your code has been saved successfully! ID = '.$id);
  }
```

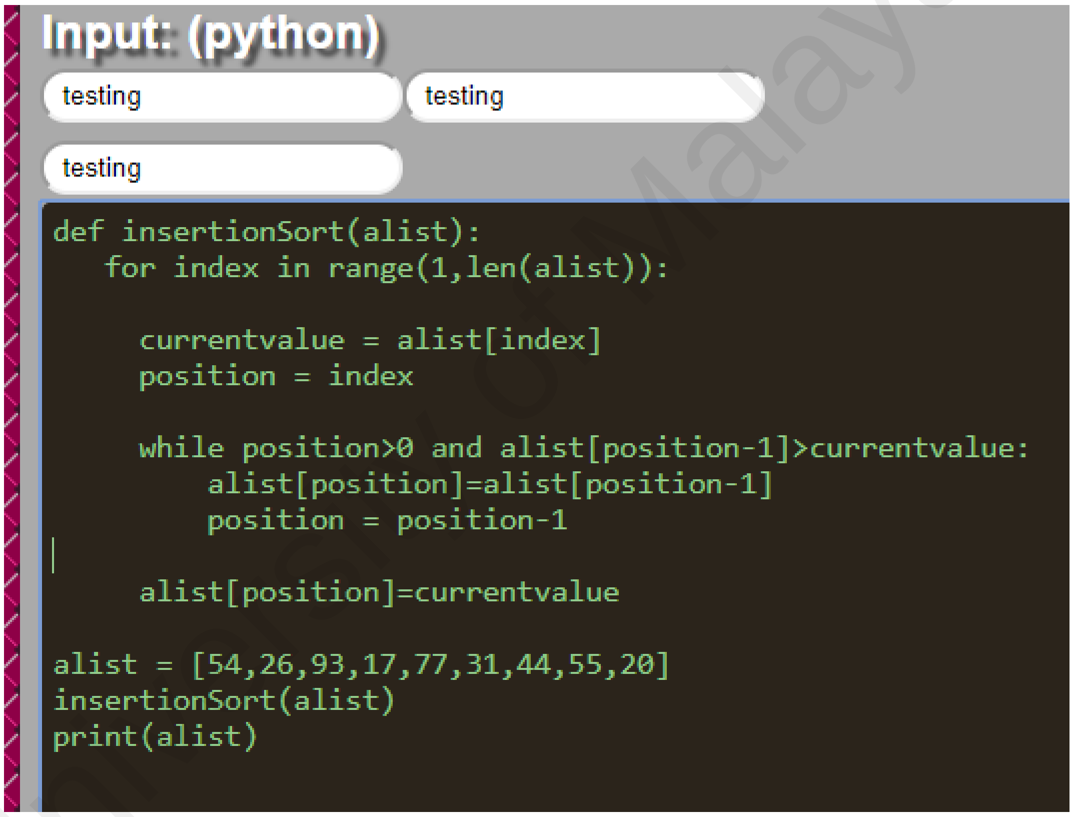
Figure 5.15 Screenshot of the activity add by teacher module

This chapter discussed and illustrated the tool design, tool development and its implementation. In design section, model design, formula design, system design, interface design, and user accessibility scope are explained. Following that, tool implementation begins by selecting proper tools and technologies. Then development begins from the scratch on MVC architecture of Laravel framework using PHP programming language. In this section, all important parts of the development are shown in through screenshots.

CHAPTER 6: TESTING AND EVALUATION

6.1 Testing

For to test the tool three testing methods were used. In the test, multiple known standard python codes were run through the model and their resulting outputs were verified by external calculations. Here is one example, presented for illustration. The following code shown in Figure 6.1 was written in the code canvas for testing. Testing methods will be farther discussed in the following testing sections.



```
def insertionSort(alist):
   for index in range(1,len(alist)):

     currentvalue = alist[index]
     position = index

     while position>0 and alist[position-1]>currentvalue:
         alist[position]=alist[position-1]
         position = position-1

     alist[position]=currentvalue

alist = [54,26,93,17,77,31,44,55,20]
insertionSort(alist)
print(alist)
```

Figure 6.1 Demo code for testing.

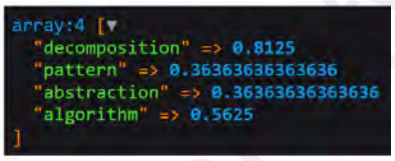
6.1.1 Unit Test

Using unit test, the validity of the proposed model was checked through in-built testing functions provided by Laravel. It passes each of the tests that have been mentioned in this section. Some of the example values are mentioned below,

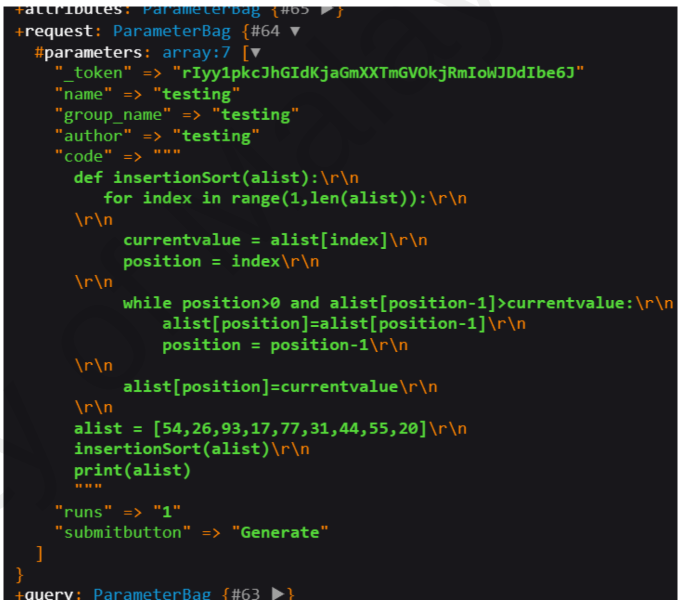
$stats['loops'] = *2*, *$stats['comparisons']* = *2*, *$stats[defined_func]* = *1*, *$stats[conditions]= 0*, *$stats[statements] =11* etc. Here, all can be seen as correct.

6.1.2 Module Test

After the unit test was done the system went through module test. All modules have passed the module test. Again, multiple known standard python codes were run through the model and their resulting CT scores were verified. For the demo code, some of the examples are given below as shown in Figure 6.2.



```
array:4 [▼
  "decomposition" => 0.8125
  "pattern" => 0.36363636363636
  "abstraction" => 0.36363636363636
  "algorithm" => 0.5625
]
```



```
+attributes: ParameterBag {#65 ▶}
+request: ParameterBag {#64 ▼
  #parameters: array:7 [▼
    "_token" => "rIyy1pkcJhGIdKjaGmXXTmGVOkjRmIoWJDdIbe6J"
    "name" => "testing"
    "group_name" => "testing"
    "author" => "testing"
    "code" => """
      def insertionSort(alist):\r\n
         for index in range(1,len(alist)):\r\n
      \r\n
           currentvalue = alist[index]\r\n
           position = index\r\n
      \r\n
           while position>0 and alist[position-1]>currentvalue:\r\n
               alist[position]=alist[position-1]\r\n
               position = position-1\r\n
      \r\n
           alist[position]=currentvalue\r\n
      \r\n
      alist = [54,26,93,17,77,31,44,55,20]\r\n
      insertionSort(alist)\r\n
      print(alist)
      """
    "runs" => "1"
    "submitbutton" => "Generate"
  ]
}
+query: ParameterBag {#63 ▶}
```

(a) (b)

Figure 6.2 (a) Array output of 'generate' module; (b) data received by the 'save module

Using the formula for pattern recognition,

$$Pattern\ recognition \Rightarrow [\{(loops),(conditions),(comparisons)\} : \{(valid\ statements)\}] * 100$$

$$Pattern\ recognition \Rightarrow [\{(2),(0),(2)\} : \{(11)\}] * 100$$

$$Pattern\ recognition \Rightarrow \frac{4}{11} * 100 \Rightarrow 36.36$$

Therefore, it can be seen that the module test result is correct.

6.1.3 Integration Test

After the module test was done the CT model was integrated within the server and to access frontend HTML CSS were used. Laravel views were properly integrated with its controllers and models. Henceforth, all the values generated by the modules are able to communicate each other where necessary and pass necessary information is tested and found to be correct. This integration was alpha tested by other developers and teachers.

6.2 Evaluation

The Code Analyzer interface is specifically built to incorporate the code canvas with visual of generated CT elements and reference CT elements for comparison. Meanwhile, there is an additional tab for instructors to see classroom activity summary and get the suggestions.

Seven university students and seven university lecturers were invited to use the application in order to evaluate the usability of the application tool. The recorded data can be found in Table B(ii) of Appendix B. After going through Code Analyzer Tool, they responded to a questionnaire to evaluate the general aspects, Structure and Navigation, and System evaluation. Here, the ranking is indexed from 1 to 5 with 1-strongly disagree, 2-disagree, 3-neutral, 4-agree, 5-strongly agree. Table 6.1, Table 6.2, and Table 6.3 show the average value of the responses for the general aspects, structure and navigation, and system evaluation. It can be seen that university students and lecturers generally agree to the Code Analyzer tool for generating and comparing CT elements of student's codes and providing classroom activity suggestions.

6.3 General Aspects

Average rating for general aspects is 4.04 (shown in Table 6.1) which indicates on average everyone agrees on all the general aspects mentioned in this section.

Table 6.1 Average rating for general aspects of the tool given by the users

No	Aspects	Average score
I	Goals of the sites are concrete and well defined.	4.07
II	Contents are precise and complete.	4.00
III	General design of the website is recognizable.	4.00
IV	General design of the site is coherent.	4.07
V	The system has been designed as the intended objectives.	4.07

6.4 Structure and Navigation

Average rating for Structure and Navigation is 4.11 (shown in Table 6.2) which indicates on average everyone agrees on all the Structure and Navigation aspects mentioned in this section.

Table 6.2 Average rating for structure and navigation of the tool given by the users

No	Aspects	Average score
I	Structure and navigation are adequate.	4.00
II	Links, buttons are easily recognizable as such.	4.07
III	Broken links/buttons are avoided.	4.00
IV	Redundant links/buttons are avoided.	4.21
V	A link to the initial stage of the page is always present.	4.29

6.5 System Evaluation

Average rating for system evaluation is 4.10 (shown in Table 6.3) which indicates on average everyone agrees on all the system evaluation aspects mentioned in this section.

Table 6.3 Average rating for system evaluation of the tool given by the users

No	Aspects	Average score
I	Tool is intuitively understandable to use.	3.86
II	System takes all the necessary inputs in correct order.	4.21
III	Tool is capable to suggest correct gaming activities.	3.93
IV	Suggested instructions are clear and precise.	4.00
V	The system is fast and responsive.	4.21
VI	Tool can save important data for further management.	4.57
VII	Rate the Code Canvas section	4.14
VIII	Rate the CT Elements view section	4.07
IX	Rate the Activity Suggestion section	3.93

6.6 Summary

Through this chapter, testing and evaluation are shown. In testing, unit test, module test, and integration test have been done for the prototype. Then, evaluation of the tool is done through seven university lecturers and seven postgraduate students. The average evaluation score is a little above four for both type of testers.

CHAPTER 7: CONCLUSION

The conclusion begins with revisiting the aim and objectives of this research. The following are then on the outcome and subsequently the research challenges and scope of improvements.

7.1 Research Findings

The aim of the present research was to explore teaching programming and CT through competitive physical and tactile games by means of a qualitative study. Through our research questions related to participants' interest, understanding of topics, CT, and academic performance, positive results were found for all research questions. This approach of teaching students through competitive physical and tactile games constitutes a potentially powerful tool for instilling interest in students while keeping the classroom maximally effective and enjoyable. The competitive element in game activities may be a crucial ingredient that helped secure students' commitment toward completing the assigned game tasks.

Students could grasp/understand the topics/algorithms covered as the learning outcomes. The game activities also promoted decomposition, as well as algorithmic and abstraction thinking, which are indicators of CT. Finally, the academic performance of a group of students who learned through physical games was higher than that of a group of students who learned in a more traditional classroom.

The second aim of the research was to design and develop a prototype that can link CT with programming directly. From literature review and methodology chapter information regarding CT elements are extracted and a calculable relation between CT and programming was form based on keywords, functions, statements, comparisons and conditions. Then build a MySQL database to manage all the resources. And finally, the entire tool has been successfully put in the web platform with the aid of Laravel framework.

Objective-1

In preliminary stage the students participated in classroom activities and their engagement was confirmed through the result shown in Figure 4.2. They enjoyed classroom activities while it enhances their understanding of programming concept and improves their academic performance as well.

Objective-2

The proposed model shown in Figure 5.1 was used to associate the CT elements to the program codes. The hint for this association was gathered from literature review. The improvement of the students' CT scores after conducting the activities infers the validity of the model.

Objective-3

The prototype developed meets the intended requirement of the Objective-3. The standard code given by the lecurer was used to calculate the referencing CT elements for students. Using the proposed model, the tool is able to generate CT scores of the students' code and students are able to compare CT elements of their codes with the corresponding reference CT elements correctly. The prototype does a cumulative average of students' CT scores and enables lecturers to choose suitable classroom activities for further improvement of the class.

Objective-4

The improvement shown in Table 4.1 confirms the effectiveness of the tool. The students who played the games suggested by the prototype developed has obtained higher score compared to controlled group. Student's improved performance confirms that the generated and compared CT percentage was correct and useful.

7.2 Significant of Study

The preliminary study prolongs our existing knowledge about the impact and effectiveness of nonconventional classroom activities especially using physical games. And this study identifies CT elements from program codes through an assistive tool. The novelty in this study is that for the first time a system is able to analyze python programming code and produce four CT elements which can have much bigger implication in teaching by adding other programming language like Java, C/C++ etc.

7.3 Concluding Remark

Therefore, it can be concluded, based on the evidences gathered herein, that the competitive physical and tactile games enhanced students' understanding of programming and CT concepts. Our execution of the physical game to explain programming concepts, especially in a competitive scenario, clearly showed a tremendous potential to instil interest, capture focus, and increase grade performance among various groups of university students. Also, the design architecture of the tool has been successfully implemented enabling it to generate CT elements from code in percentage value and to give instructors compatible gaming activities for the class.

Comparing the obtained CT value with reference value, now teachers and instructors can easily find which activities to play for a particular student (or a class) to improve his/her lacking in understanding/skill in programming language.

7.4 Research Challenges

A limitation of the present study is that only a few academic topics were covered. More games could be created to cover as many topics as possible. It was intended to use games in all lecture periods and observe the students' performance through summative and formative evaluations.

The major challenge was to conduct workshops with a group of students and keep track of them after. Some participants were out of reach after activity to note the feedback and for validation where some were slow to response from distance makes research progress slower.

7.5 Scope of Improvements

Its functionality is limited to python programming for the moment. Later, C/C++, Java, PHP, and other programming languages can be added as a functioning platform.

In the interim, it does not verify whether an intended objective has been achieved through programming, rather it processes based on keywords, functions, loops, conditions etc. used. Hence, there is a scope to improve it by validating whether intended objective of particular program has been achieved or not achieved.

REFERENCES

Ab Hamid, S. H. (2004). Solutions To Teaching Object-Oriented Programming. *WSEAS Transactions on Communications*, *3*(1), 99–104. Retrieved from http://www.wseas.us/e-library/conferences/cancun2004/papers/485-210.doc

Ab Hamid, S. H., & Leong, Y. F. (2007). Learn programming by using mobile edutainment game approach. In *Proceedings - DIGITEL 2007: First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning* (pp. 170–172). https://doi.org/10.1109/DIGITEL.2007.31

Al-Bow, M., Austin, D., Edgington, J., Fajardo, R., Fishburn, J., Lara, C., … Meyer, S. (2009). Using Game Creation for Teaching Computer Programming to High School Students and Teachers. *Iticse 2009: Proceeding of the 2009 Acm Sigse Annual Conference on Innovation and Technology in Computer Science Education*, 104–108406.

Anshari, M., Almunawar, M. N., Shahrill, M., Wicaksono, D. K., & Huda, M. (2017). Smartphones usage in the classrooms: Learning aid or interference? *Education and Information Technologies*. https://doi.org/10.1007/s10639-017-9572-7

Ateş, A., & Altun, E. (2008). Learning styles and preferences for students of computer education and instructional technologies. *Egitim Arastirmalari - Eurasian Journal of Educational Research*, (30), 1–16.

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads*. https://doi.org/10.1145/1929887.1929905

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer Science Unplugged: School Students Doing Real Computing Without Computers. *Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Bell, T., & Newton, H. (2013). Using Computer Science Unplugged as a teaching tool. *Improving Computer Science Education*, 1–18. Retrieved from routledge-ny.com/books/details/9780415645379/

Bell, T., Rosamond, F., & Casey, N. (2012). Computer science unplugged and related projects in math and computer science popularization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-30891-8_18

Bell, T., Witten, I. H., Fellows, M., Adams, R., McKenzie, J., Powell, M., & Jarman, S. (2006). CS Unplugged. Retrieved May 27, 2018, from http://csunplugged.org/

Bishop, J. (2008). Language features meet design patterns: raising the abstraction bar. In *Proceedings of the 2nd international workshop on The role of abstraction in software engineering*. https://doi.org/10.1145/1370164.1370166

Brady, C., Orton, K., Weintrop, D., Anton, G., Rodriguez, S., & Wilensky, U. (2017). All Roads Lead to Computing: Making, Participatory Simulations, and Social

Computing as Pathways to Computer Science. *IEEE Transactions on Education*. https://doi.org/10.1109/TE.2016.2622680

Bransford, J. D., Brown, A., & Cocking, R. (1999). How people learn: Mind, brain, experience, and school. *Washington, DC: National Research Council*, 374. https://doi.org/10.1016/0885-2014(91)90049-J

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting, Vancouver, BC, Canada*. https://doi.org/10.1.1.296.6602

Brown, C. (2007). Learning Through Multimedia Construction-A Complex Strategy. *Journal of Educational Multimedia and Hypermedia*, *16*(2), 93–124. Retrieved from http://search.proquest.com.ezp.lib.unimelb.edu.au/docview/205853446?accountid=12372%5Cnhttp://sfx.unimelb.hosted.exlibrisgroup.com/sfxlcl41?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&genre=article&sid=ProQ:ProQ%3Aeducation&atitle=Learnin

Campbell, V., & Johnstone, M. (2010). The significance of learning style with respect to achievement in first year programming students. In *Proceedings of the Australian Software Engineering Conference, ASWEC*. https://doi.org/10.1109/ASWEC.2010.33

Creswell, J. W. (2007). Qualitative inquiry and research design: Choosing cmong five traditions. *Qualitative Health Research*. https://doi.org/10.1111/1467-9299.00177

Curzon, P. (2013). cs4fn and computational thinking unplugged. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education on - WiPSE '13* (pp. 47–50). https://doi.org/10.1145/2532748.2611263

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014). Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education on - WiPSCE '14* (pp. 89–92). https://doi.org/10.1145/2670757.2670767

David J. Barnes and Michael Kölling. (2013). *Objects First with Java^{TM}: A Practical Introduction Using BlueJ. University of Kent*. https://doi.org/10.1017/CBO9781107415324.004

Farrell, J. (2014). *Java Programming*. (M. Lee, Ed.) (7th ed.). Boston, USA: Cengage Learning.

Glen, S. (2016). T Test (Student's T-Test): Definition and Examples. https://doi.org/10.1007/978-94-007-0753-5

Hamid, S. H. A., & Ismail, N. (2007). The design of MobiGP by using Tamagotchi. In *Proceedings of the 2007 1st International Symposium on Information Technologies and Applications in Education, ISITAE 2007* (pp. 382–387). https://doi.org/10.1109/ISITAE.2007.4409309

Hanus, M. D., & Fox, J. (2015). Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers and Education*. https://doi.org/10.1016/j.compedu.2014.08.019

Hazzan, O., & Kramer, J. (2008). The role of abstraction in software engineering. In *Companion of the 13th international conference on Software engineering - ICSE Companion '08*. https://doi.org/10.1145/1370175.1370239

Hegazi, M. O., & Alhawarat, M. (2015). The Challenges and the Opportunities of Teaching the Introductory Computer Programming Course: Case Study. In *2015 Fifth International Conference on e-Learning (econf)*. https://doi.org/10.1109/ECONF.2015.61

Jimoyiannis, A., & Tsiotakis, P. (2017). Beyond students' perceptions: investigating learning presence in an educational blogging community. *Journal of Applied Research in Higher Education*. https://doi.org/10.1108/JARHE-06-2015-0046

Johnstone, M. J., & Kanitsaki, O. (2006). Culture, language, and patient safety: Making the link. *International Journal for Quality in Health Care*, *18*(5), 383–388. https://doi.org/10.1093/intqhc/mzl039

Kafai, Y. B. (2001). The Educational Potential of Electronic Games : From Games – To – Teach to Games – To – Learn. *Playing by the Rules*, 1–6. Retrieved from http://culturalpolicy.uchicago.edu/papers/2001-video-games/kafai.html

Klement, M. (2014). How do my Students Study? An Analysis of Students' of Educational Disciplines Favorite Learning Styles According to VARK Classification. *Procedia - Social and Behavioral Sciences*, *132*, 384–390. https://doi.org/10.1016/j.sbspro.2014.04.326

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ System and its Pedagogy. *Computer Science Education*, *13*(4), 249–268. https://doi.org/10.1076/csed.13.4.249.17496

Kunkle, W. M., & Allen, R. B. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computinig Education*, *16*(1). https://doi.org/10.1145/2785807

Lee, C., Potkonjak, M., & Mangione-smith, W. H. W. H. (1997). MediaBeinch : A Tooi for Evaluating and Synthesizing Multimedia and Communications Systems The University of California at Los Angeles 1 Introduction 3 MediaBench Components 2 Previous Work. *Micro'97*. https://doi.org/10.1109/MICRO.1997.645830

Lethbridge, T. C. (2000). Priorities for the education and training of software engineers. *Journal of Systems and Software*, *53*(1), 53–71. https://doi.org/10.1016/S0164-1212(00)00009-1

Liu, A., Newsom, J., Schunn, C., & Shoop, R. (2013). Students learn programming faster through robotic simulation. *Tech Directions*, *72*(8), 16–19. Retrieved from http://www.education.rec.ri.cmu.edu/content/educators/research/files/p16-19 Shoop et al.pdf

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*. https://doi.org/10.1016/j.chb.2014.09.012

Malone, T. W. (1980). What makes things fun to learn? heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems - SIGSMALL '80* (pp. 162–169). https://doi.org/10.1145/800088.802839

Marcolino, A. S., & Barbosa, E. (2017). A survey on problems related to the teaching of programming in Brazilian educational institutions. In *Proceedings - Frontiers in Education Conference, FIE* (Vol. 2017–Octob, pp. 1–9). https://doi.org/10.1109/FIE.2017.8190495

Morse, J. M. (1994). Designing funded qualiative research. In *Handbook of qualitative research*.

Noor, N. M., Aini, M., & Hamizan, N. I. (2014). Video Based Learning Embedded with Cognitive Load Theory: Visual, Auditory, and Kinaesthetic Learners' Perspectives. In *2014 International Conference on Teaching and Learning in Computing and Engineering*. https://doi.org/10.1109/LaTiCE.2014.19

Norwawi, N. M., Abdusalam, S. F., Hibadullah, C. F., & Shuaibu, B. M. (2009). Classification of students' performance in computer programming course according to learning style. In *2009 2nd Conference on Data Mining and Optimization, DMO 2009*. https://doi.org/10.1109/DMO.2009.5341912

Oblinger, D. G. (2004). The Next Generation of Educational Engagement. *Journal of Interactive Media in Education*, *2004*(8), 1–18. https://doi.org/10.5334/2004-8-oblinger

Ortiz, O. O., Franco, J. A. P., Garau, P. M. A., & Martin, R. H. (2017). Innovative mobile robot method: Improving the learning of programming languages in engineering degrees. *IEEE Transactions on Education*. https://doi.org/10.1109/TE.2016.2608779

Papastergiou, M. (2009). Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation. *Computers & Education*, *52*(1), 1–12. https://doi.org/10.1016/j.compedu.2008.06.004

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, *52*, 60–67. https://doi.org/10.1145/1592761.1592779

Roehl, A., Reddy, S. L., & Shannon, G. J. (2013). The Flipped Classroom: An Opportunity to Engage Millenial STudents through Active Learning Strategies. *Journal of Family and COnsumer Sciences*.

Rosen, F. (1831). The Algebra of Mohammed Ben Musa. *London: The Oriental Translation Fund*. https://doi.org///catalog.hathitrust.org/Record/006552165

Rubinstein, A., & Chor, B. (2014). Computational Thinking in Life Science Education. *PLoS Comput Biol*, *10*(11), e1003897+.

https://doi.org/10.1371/journal.pcbi.1003897

Sa Lorca, L. (2018). The Basics of Computational Thinking. Retrieved July 8, 2018, from https://webdesign.tutsplus.com/articles/the-basics-of-computational-thinking--cms-30172

Silapachote, P., & Srisuphab, A. (2017). Teaching and learning computational thinking through solving problems in Artificial Intelligence: On designing introductory engineering and computing courses. In *Proceedings of 2016 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2016*. https://doi.org/10.1109/TALE.2016.7851769

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*. https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D., & Wilensky, U. (2013). RoboBuilder: A Computational Thinking Game. *Sigcse*. https://doi.org/10.1145/2445196.2445430

Willingham, D. T. (2012). *Why Don't Students Like School?: A Cognitive Scientist Answers Questions About How the Mind Works and What It Means for the Classroom*. *Why Don't Students Like School?: A Cognitive Scientist Answers Questions About How the Mind Works and What It Means for the Classroom*. https://doi.org/10.1002/9781118269527

Wing, J. (2006). Computational Thinking - It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use. *Communications of the ACM*. https://doi.org/0001-0782/06/0300

Wolfram, S. (2016). How to Teach Computational Thinking—Stephen Wolfram Blog. Retrieved July 11, 2018, from http://blog.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/

Ying Li. (2016). Teaching programming based on Computational Thinking. In *2016 IEEE Frontiers in Education Conference (FIE)* (pp. 1–7). IEEE. https://doi.org/10.1109/FIE.2016.7757408

Yusof, A. M., & Abdullah, R. (2005). The Evolution of Programing Courses: Course curriculum, students, and their performance. *ACM SIGCSE Bulletin*, *37*(4), 74–78. Retrieved from https://dl.acm.org/citation.cfm?id=1113881

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Title: Enhancing Understanding of Programming Concepts through Physical Games

Conference: Learning Innovation and Teaching Enhancement Research Conference 2017

Date: March 15th - 16th, 2017

Location: Pullman Hotel, Bangsar, Kuala Lumpur, Malaysia