

*Menoufia University*

*Faculty of Computers and Information*

*Computer Science Department*



Graduation Project Documentation

**PSCE: Problem Solving Contest  
Environment**



BY

Ahmed Samir Daoush

Amr Hassan Saud

Ola Galal ElKhouly

Ola Nazmy Farag

**Supervisor: Dr. Amira Ibrahim**

**July 2019**



## **ACKNOWLEDGMENT**

First to God, the most Gracious, the Most Merciful, who gave us the patience and endurance to complete this work

We dedicate this word to our supervisor,

Dr. Amira Ibrahim

Who had the knowledge to teach us what we need to know, and the wisdom to let us to learn for ourselves.

so special thanks to our supervisor who helped us from the first second to the last one, who gives the courage and boldness to take right steps without any fear from our failure and frustration at the beginning of time, and still on this case until we go ahead on right way.

We also dedicate it to everyone who helped us throughout the course of development of this project. Without your help, we would not have succeeded.

Last but not least, we dedicate it to our friends and the families for always standing by our sides and never losing faith in our abilities.



## **ABSTRACT**

PSCE is a dynamic distributed real-time system designed to manage and control programming contests in a variety of computing environments. It is designed to manage and control programming contests in a variety of computing environments. It includes support to a variety of programming languages. The system is designed to allow teams to use any language development tool which can be invoked from a command line and generates an executable file. It is automatically timestamping and archive submitted code. PSCE supports two types of judging. The first type is automated (or computer) judging and second type are manual (or human) judging. These two types used independently or in conjunction with each other. It allows students (contestants) to submit programs over the same network to contest judges. The judges can recompile the submitted program, execute it, view the source code and/or execution results, and send a response back to the team. The judge also can retrieve and re-execute archived runs. It also provides a mechanism for students to submit clarification requests and queries to the judge, and for the judge to reply to queries and to issue broadcast to students. It can also detect plagiarism and exclude students who are cheating.



# TABLE OF CONTENT

LIST OF FIGURES .....	iv
LIST OF TABLES .....	v
LIST OF ACRONYMS/ABBREVIATIONS .....	vi
1 INTRODUCTION .....	1
1.1 OVERVIEW .....	1
1.2 MOTIVATION .....	3
1.3 PROCEDURES AND METHODOLOGY .....	3
2 BACKGROUND / RELATED WORK .....	4
2.1 MODEL .....	4
2.2 METHODOLOGY .....	5
2.3 SOFTWARE DESIGN PATTERN .....	6
2.4 CHALLENGES .....	7
2.5 PROBLEMS .....	9
2.6 RELATED WORK .....	10
3 SYSTEM LIFE DEVELOPMENT CYCLE .....	12
3.1 PLANNING .....	13
3.2 ANALYSIS AND REQUIREMENTS .....	14
3.2.1 System Requirements .....	14
3.2.1.1 Functional System Requirements .....	14
3.2.1.2 Non-Functional System Requirements .....	14
3.2.2 User Requirements .....	15
3.2.2.1 Functional User Requirements .....	15
3.3 DESIGN .....	15
3.3.1 DATA FLOW DIAGRAM (DFD) .....	17
3.3.1.1 Context Diagram .....	17
3.3.1.2 Level 0 Diagram .....	18
3.3.1.3 Level 1 Diagram: .....	20
3.3.1.4 Level 2 Diagram: .....	20
3.3.2 ENTITY RELATIONSHIP DIAGRAM (ERD) .....	21
3.3.3 UNIFIED MODELING LANGUAGE (UML) .....	22
3.3.3.1 Use Case Diagram .....	22
3.3.3.2 Sequence Diagram .....	24

3.4	DEVELOPMENT.....	26
3.5	INTEGRATION AND TESTING.....	27
3.5.1	Program Test.....	28
3.5.2	System Test.....	28
3.5.3	Testing Methods.....	29
3.5.3.1	Static vs. dynamic testing.....	29
3.5.3.2	Box Approaches .....	29
3.5.3.2.1	White-box testing .....	29
3.5.3.2.2	Black-box testing.....	31
3.5.3.2.3	Grey-box testing.....	31
3.6	IMPLEMENTATION .....	32
3.7	OPERATIONS AND MAINTENANCE .....	33
3.8	WHY SDLC PHASES?.....	34
4	PSCE TOOLS.....	35
4.1	ALGORITHMS .....	35
4.1.1	Minimum Edit Distance .....	35
4.1.2	Balanced Binary Search Tree.....	36
4.1.3	Red-Black Tree .....	36
4.1.4	Hashing .....	39
4.1.5	Quicksort.....	39
4.2	TECHNOLOGIES.....	39
4.2.1	Socket Programming in Java .....	39
4.2.2	System Calls in Java .....	40
4.2.3	TeX .....	40
4.2.4	MathJax.....	40
4.3	DESKTOP APPLICATION .....	47
4.3.1	Java .....	47
4.3.2	FXML .....	48
4.3.3	MySQL .....	48
4.4	WEB SITE.....	48
4.4.1	HTML .....	48
4.4.2	CSS .....	49
4.4.3	JavaScript.....	49
4.4.4	Bootstrap.....	50
4.4.5	Java Servlet .....	50



4.4.6	JSP.....	51
5	SYSTEM CONFIGURATION AND GUIDE .....	53
5.1	DESKTOP APPLICATIONS CONFIGURATION .....	53
5.1.1	Software Requirements:.....	53
5.1.2	Hardware Requirements: .....	53
5.2	WEBSITE CONFIGURATION .....	53
5.2.1	Software Requirements:.....	53
5.2.2	Hardware Requirements: .....	53
5.3	HOW TO USE? .....	54
5.3.1	Desktop application (Student-side and Judge-side).....	54
5.3.2	Web Application (Problems, Clarifications, and Standing).....	60
5.3.3	Desktop Application (Plagiarism).....	63
6	CONCLUSION AND FUTURE WORK .....	65
6.1	CONCLUSION .....	65
6.2	FUTURE WORK .....	65
7	REFERENCES .....	66
	APPENDIX.....	67

## LIST OF FIGURES

<b>Figure 2-1: Phases of Agile Model .....</b>	<b>5</b>
<b>Figure 2-2: MVC Components .....</b>	<b>7</b>
<b>Figure 3-1: What is the System?.....</b>	<b>12</b>
<b>Figure 3-2: The different phases in the system development life cycle .....</b>	<b>13</b>
<b>Figure 3-3: Data Flow Diagram.....</b>	<b>18</b>
<b>Figure 3-4: Level 0 Diagram .....</b>	<b>19</b>
<b>Figure 3-5: Level 1 Diagram .....</b>	<b>20</b>
<b>Figure 3-6: Level 2 Diagram .....</b>	<b>20</b>
<b>Figure 3-7: Level 2.1 Diagram .....</b>	<b>21</b>
<b>Figure 3-8: Entity-Relationship Diagram .....</b>	<b>21</b>
<b>Figure 3-9: Use Case Diagram.....</b>	<b>23</b>
<b>Figure 3-10: Contestant Sequence Diagram .....</b>	<b>25</b>
<b>Figure 3-11: Judge Sequence Diagram .....</b>	<b>26</b>
<b>Figure 3-12: System Test Types .....</b>	<b>29</b>
<b>Figure 3-13: Steps of Operations and Maintenance Phase .....</b>	<b>34</b>
<b>Figure 4-1: Minimum Edit Distance Algorithm .....</b>	<b>35</b>
<b>Figure 4-2: Encoding 3-nodes.....</b>	<b>36</b>
<b>Figure 4-3: 1–1 correspondence between red-black BSTs and 2–3 tree .....</b>	<b>37</b>
<b>Figure 4-4: Color representation in red-black BSTs.....</b>	<b>38</b>
<b>Figure 4-5: Search and insert operations .....</b>	<b>38</b>
<b>Figure 4-6: JSP Model architecture .....</b>	<b>52</b>
<b>Figure 5-1: Student Login Interface.....</b>	<b>54</b>
<b>Figure 5-2: Student Wait Interface .....</b>	<b>54</b>
<b>Figure 5-3: Judge Wait Interface .....</b>	<b>55</b>
<b>Figure 5-4: Add Problem interface .....</b>	<b>55</b>
<b>Figure 5-5: Add Confirmation Interface .....</b>	<b>56</b>
<b>Figure 5-6: Judge Main Interface.....</b>	<b>56</b>
<b>Figure 5-7: Student Main interface - Tab 1 .....</b>	<b>57</b>
<b>Figure 5-8: Student Main interface - Tab 2 .....</b>	<b>57</b>
<b>Figure 5-9: Student Submit Confirmation Message .....</b>	<b>58</b>
<b>Figure 5-10: Automatic Response of the Result.....</b>	<b>58</b>
<b>Figure 5-11: Judge Response Interface for Judge .....</b>	<b>59</b>
<b>Figure 5-12: Judge Response Interface for Student.....</b>	<b>60</b>
<b>Figure 5-13: Home Page .....</b>	<b>61</b>
<b>Figure 5-14: Problems Page.....</b>	<b>62</b>
<b>Figure 5-15: Clarification Page.....</b>	<b>62</b>
<b>Figure 5-16: Standing Page.....</b>	<b>63</b>
<b>Figure 5-17: Plagiarism Main Interface.....</b>	<b>63</b>
<b>Figure 5-18: Plagiarism Result Interface .....</b>	<b>64</b>

## LIST OF TABLES

<b>Table 1:</b> Operation symbols .....	44
<b>Table 2:</b> Miscellaneous symbols .....	44
<b>Table 3:</b> Relation symbols.....	45
<b>Table 4:</b> Logical symbols.....	45
<b>Table 5:</b> Grouping brackets.....	45
<b>Table 6:</b> Arrows .....	45
<b>Table 7:</b> Accents.....	46
<b>Table 8:</b> Greek Letters.....	46

## LIST OF ACRONYMS/ABBREVIATIONS

<b>ACRONYM</b>	<b>Definition of Acronym</b>
<b>MVC</b>	Model – View – Control
<b>JSP</b>	Java Server Pages
<b>JSF</b>	Java Server Faces
<b>UI/UX</b>	User Interface / User Experience
<b>SQL</b>	Structured Query Languages
<b>XP</b>	Extreme Programming
<b>TDD</b>	Test-Driven Development
<b>GUI</b>	Graphical User Interface
<b>TCP</b>	Transmission Control Protocol
<b>IP</b>	Internet Protocol
<b>ACM</b>	Association for Computing Machinery
<b>ICPC</b>	International Collegiate Programming Contest
<b>KTH</b>	Royal Institute of Technology
<b>API</b>	Application Programming Interface
<b>OCR</b>	Optical Character Recognition
<b>BST</b>	Binary Search Tree
<b>AVL tree</b>	(named after inventors Adelson-Velsky and Landis) is a self-balancing binary search tree.
<b>JRE</b>	Java Runtime Environment
<b>JVM</b>	Java Virtual Machine

## *Chapter One*

# 1 INTRODUCTION

## 1.1 OVERVIEW

Problem Solving is the core of computer science. Programmers must first understand how a human solves a problem, then understand how to translate this "algorithm" into something a computer can do, and finally how to "write" the specific syntax (required by a computer) to get the job done. It is sometimes the case that a machine will solve a problem in a completely different way than a human.

Computer Programmers are problem solvers. In order to solve a problem on a computer you must:

1. Know how to **represent** the information (data) describing the problem.
2. Determine the steps to **transform** the information from one representation into another.

A computer, at heart, is dumb. It can only know about a few things... numbers, characters, Booleans, and lists (called arrays) of these items. Everything else must be "approximated" by combinations of these data types. A good programmer will **encode** all the **facts** necessary to represent a problem in variables. Further, there are *good ways* and *bad ways* to encode information. Good ways allow the computer to easily *compute* new information.

An algorithm is a set of specific steps to solve a problem. Another way to describe an algorithm is a sequence of unambiguous instructions. The use of the term 'unambiguous' indicates that there is no room for subjective interpretation. Every time you ask your computer to carry out the same algorithm, it will do it in exactly the same manner with the exact same result. The core of what good programmers do is being able to define the steps necessary to accomplish a goal.

A computer programming contest is a competition where teams submit (computer program) solutions to judges. The teams are given a set of problems to solve in a limited amount of time (for example 8-13 problems in 5 hours). The judges then give a pass/fail judgments to the submitted solutions. Team rankings are computed based on the solutions, when the solutions were submitted and how many attempts were made to solve the problem. The judges test in a Black box testing where the teams do not have access to the judges' test data.

The aim of competitive programming is to write the source code of computer programs which are able to solve given problems. A vast majority of problems appearing in programming contests are mathematical or logical in nature. Typical such tasks belong to one of the following categories: combinatorics, number theory, graph

theory, geometry, string analysis and data structures. Problems related to artificial intelligence are also popular in certain competitions.

Irrespective of the problem category, the process of solving a problem can be divided into two broad steps: constructing an efficient algorithm and implementing the algorithm in a suitable programming language (the set of programming languages allowed varies from contest to contest). These are the two most commonly tested skills in programming competitions.

In most contests, the judging is done automatically by host machines, commonly known as judges. Every solution submitted by a contestant is run on the judge against a set of (usually secret) test cases. Normally, contest problems have an all-or-none marking system, meaning that a solution is "Accepted" only if it produces satisfactory results on all test cases run by the judge, and rejected otherwise. However, some contest problems may allow for partial scoring, depending on the number of test cases passed, the quality of the results, or some other specified criteria. Some other contests only require that the contestant submit the output corresponding to given input data, in which case the judge only has to analyze the submitted output data.

PSCE is a dynamic distributed real-time system designed to manage and control programming contests in a variety of computing environments. It includes support to a variety of programming languages. The system is designed to allow teams to use any language development tool which can be invoked from a command line and generates an executable file. It automatically timestamps and archive submitted the code.

Our system is a distributed system, composed of multiple systems working together to achieve the established our objectives. They are integrated to create a professional environment to help people, especially in the academic field to hold their own local (onsite) contests or quizzes.

In particular, it provides support for two distinct methods of judging: automated (or computer) judging and manual (or human) judging. These two methods used independently or in conjunction with each other. It allows students (contestants) to submit programs over a network to contest judges. The judges can recompile the submitted program, execute it, view the source code and/or execution results, and send a response back to the team. The judge also can retrieve and re-execute archived runs. It also provides a mechanism for students to submit clarification requests and queries to the judge, and for the judge to reply to queries and to issue broadcast to students.

The project operates using a client-server architecture. In the Site, there is a single server and multiple clients which communicate with the site server. One such client type is a judge. Our project is concerned with helping teachers in universities running an onsite programming exam or a contest between students (contestants).

It consists of a desktop application with two different interfaces: Contestant and Judge, a webpage to view a real-time scoreboard, another webpage to view clarifications and webpage to view problems.

It also contains a tool that helps in detect cheating or code plagiarism. This tool is applied after the contest had finished, then outputs the cheated and probably cheated contestants.

## **1.2 MOTIVATION**

The main idea of our project is to provide an environment for problem solving contests. Teachers at the faculty of computers and information face a problem when they want to make an onsite programming exam. The project gives them the chance of auto-correcting the exam, add their own problems and view real-time scoreboard. It also allows them to answer any clarifications sent by the students. It can also detect plagiarism and exclude students who are cheating. Our Project aims to ease the process of making a quiz or contest inside a university. Our objectives are - based on technical prescriptive -:

- Design using an easy interface to use by students (contestant) and doctors (judges).
- Provide a good user experience (UX) while using the app.
- Reduce the complexity of the overall process to improve performance as possible.
- Allow fast and concurrent retrieval of information.

## **1.3 PROCEDURES AND METHODOLOGY**

We developed a compatible environment that consists of many tools. First, a client desktop application that makes the student (contestant) interact with the contest and the doctor (judge). Second, a doctor (judge) desktop application that acts as a server. The judge side provides a database server (SQL Database) that record the standing, problems, and clarifications. It also provides a web server that hosts a website contains the JSP files of the standing, problems, and clarifications.

After we finished the development, we tested the code and worked on solving the problems faced us during the test.

## **2 BACKGROUND / RELATED WORK**

### **2.1 MODEL**

In our project, we used **Agile software development**. It is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s). It supports adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.

*“Good documentation is useful in helping people to understand how the software is built and how to use it, but the main point of development is to create software, not documentation.”*

– Scott Ambler, software engineer

Agile Software development mainly targets complex systems and products development with dynamic, non-deterministic and non-linear characteristics. Documentation should be ‘just barely good enough’ that too much or comprehensive documentation would usually cause waste, and developer rarely trusts detailed documentation because it’s usually out of sync with the code, while too little documentation may also cause problems for maintenance, communication, learning and knowledge sharing.

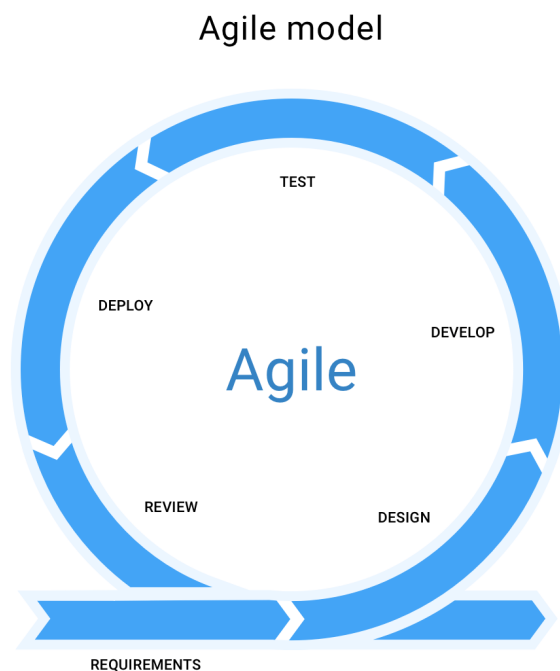
A goal of agile software development is to focus more on producing working software and less on documentation.

The general principles of the Agile Model:

- Satisfy the client and continually develop software.
- Changing requirements are embraced for the client’s competitive advantage.
- Concentrate on delivering working software frequently. Delivery preference will be placed on the shortest possible time span.
- Developers and business people must work together throughout the entire project.
- Projects must be based on people who are motivated. Give them the proper environment and the support that they need. They should be trusted to get their jobs done.



- Face-to-face communication is the best way to transfer information to and from a team.
- Working software is the primary measure of progress.
- Agile processes will promote development that is sustainable. Sponsors, developers, and users should be able to maintain an indefinite, constant pace.
- Constant attention to technical excellence and good design will enhance agility.
- Simplicity is considered to be the art of maximizing the work that is not done, and it is essential.
- Self-organized teams usually create the best designs.
- At regular intervals, the team will reflect on how to become more effective, and they will tune and adjust their behavior accordingly.



**Figure 2-1:** Phases of Agile Model

## 2.2 METHODOLOGY

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development. It supports frequent releases in a short development

cycle, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

Extreme Programming improves a software project in five essential ways; communication, simplicity, feedback, respect, and courage. Extreme Programmers constantly communicate with their customers and fellow programmers. They keep their design simple and clean. They get feedback by testing their software starting on day one. They deliver the system to the customers as early as possible and implement changes as suggested. Every small success deepens their respect for the unique contributions of each and every team member. With this foundation, Extreme Programmers are able to courageously respond to changing requirements and technology.

Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

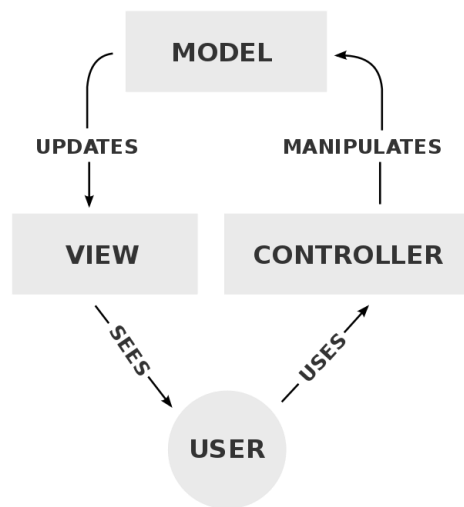
- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.
- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach test cases are written even before any code is written.
- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team come up with new increments every few days after each iteration.
- **Simplicity:** Simplicity makes it easier to develop good quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop good quality software. So, everybody should design daily.
- **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

## 2.3 SOFTWARE DESIGN PATTERN

In software engineering, a software design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Design patterns are formalized best

practices that the programmer can use to solve common problems when designing an application or system.

In our project, we used the **Model–view–controller pattern**. Model–view–controller (MVC) is commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the way’s information is presented to and accepted from the user. The MVC design pattern separates these major components allowing for code reuse and parallel development. Traditionally used for desktop graphical user interfaces (GUIs), this pattern has become popular for designing web applications.



**Figure 2-2:** MVC Components

## 2.4 CHALLENGES

In our software, we support more than one programming language for students to submit their answers, at first, we supported only C++, then we added Java Language. In future work, we are going to add many popular languages like Python, PHP, JavaScript, ...

Server management can be defined as the tasks and services that are done on a server in order to manage it. This usually entails:

- **Monitoring** of the server and apps running on the server. Checking their status, uptime, and monitoring for any new issues.
- **Updating** the server and software installed on the server. Although most server management companies offer this as part of their services, some still don't. Nevertheless, it's still considered as part of the 'server management' process.

- **Setup** and configuration. The actual server setup and configuration of software and services running on the server. Again, this may not be a part of the server management plan offered by some companies, but it most often is.

Now we will explain why we need server management.

- **Fewer costs:** Instead of hiring a full-time system administrator (which can be quite costly), you can get a server management plan and save hundreds (if not thousands) of dollars on a monthly basis. Some service providers offer a ‘managed service’ as an add-on to their servers, but it often is much more expensive than getting managed services from a 3-rd party provider.
- **Quick turnaround time:** Depending on the provider itself and the experience of the sysadmins involved, the turnaround time for server issues and tasks is fairly quick. So instead of wasting hours on troubleshooting and research, you can have experts work on your server and fix any issues in a matter of minutes. This still depends on the severity and complexity of the issue itself, but when compared, having expert server managers working on your server is far quicker than doing it yourself.
- **Fewer worries:** No need to stay updated on the latest security patches and releases, your server will be taken care of.

In our project, server management depends on Internet protocol TCP.

We managed successfully the synchronization between threads and servers, by designing the threads and servers to be independent of each other. Except for Judge thread, it is implemented inside hierarchy design inside an independent thread, so it will not cause synchronization errors. As a result of this independence, the OS dispatcher is the one that responsible for syncing.

We took into consideration the user interface and the user experience (UI/UX) to provide the best and easiest experience when they use the software. A good User Interface is important in the sense that it makes it easier for users to clearly see what our software are. Our software is designed in a way to display the services that we offer without ambiguity, in order to draw our users’ attention and keep them satisfied.

We also didn’t forget the performance of the software, we tried to use the best coding techniques to decrease the complexity in order to achieve the highest possible performance. We optimized our code to make the system run smoothly with minimum hardware requirements. (used Algorithms and Technologies are mentioned in Chapter 4)

## 2.5 PROBLEMS

Reliability means that the system stays reliable while the number of users increases. In our system the reliability depends on two main reasons, server's processors and RAM, the numbers of threads per core in the processor that can handle the students' requests is an important factor. If the hardware-enhanced, the performance will get better.

We used the internet protocol TCP as a communications protocol in the private network of the software. This protocol can be modified easily. It is compatible with all operating systems, so it can communicate with any other system. It is also compatible with all types of computer hardware and networks. It is highly scalable and, as a routable protocol, can determine the most efficient path through the network. But the network itself can suffer from several problems such as bottleneck, heavy traffic, or loss of packages.

1. **Bottleneck:** A network bottleneck refers to a discrete condition in which data flow is limited by computer or network resources. The flow of data is controlled according to the bandwidth of various system resources. If the system working on a network is delivering a higher volume of data than what is supported by the existing capacity of the network, then a network bottleneck will occur. As the name implies, a network bottleneck results in slow communication speeds and limits user efficiency and productivity on a network. To avoid all these problems, systems are built to support a particular data flow capacity so that work can continue without any issues. On a network, each system is able to work according to its processor speed, its memory size, its cache speed, and its network interface card speed. These discrete systems do not rely on other network resources to accept their incoming data at the rate they are sending because these objects only receive data according to their own capacity.
2. **Heavy Traffic:** Network traffic refers to the amount of data moving across a network at a given point of time. Network data is mostly encapsulated in network packets, which provide the load in the network. Network traffic is the main component for network traffic measurement, network traffic control, and simulation. The proper organization of network traffic helps in ensuring the quality of service in a given network.

Network traffic can be broadly classified into the following categories:

- Busy/heavy traffic - High bandwidth is consumed in this traffic.
- Non-real-time traffic - Consumption of bandwidth during working hours.

- Interactive traffic - Is subject to competition for bandwidth and could result in poor response times if prioritization of applications and traffic is not set.
- Latency-sensitive traffic - Is subject to competition for bandwidth and could result in poor response times.

Proper analysis of network traffic provides the organization with the following benefits:

- Identifying network bottlenecks - There could be users or applications that consume high amounts of bandwidth, thus constituting a major part of the network traffic. Different solutions can be implemented to solve these.
- Network engineering - Knowing the usage levels of the network allows future requirements to be analyzed.

**3. Loss of Packets:** Occurs when one or more packets of data traveling across a computer network fail to reach their destination. Packet loss is either caused by errors in data transmission, typically across wireless networks, Packet loss is measured as a percentage of packets lost with respect to packets sent.

The Transmission Control Protocol (TCP) *detects* packet loss and performs retransmissions to ensure reliable messaging. Packet loss in a TCP connection is also used to *avoid* congestion and thus produces an intentionally reduced throughput for the connection.

## 2.6 RELATED WORK

### PC<sup>2</sup>:

PC<sup>2</sup> is the Programming Contest Control System developed at California State University, Sacramento in support of Computer Programming Contest activities of the ACM, and in particular the ACM International Collegiate Programming Contest. It was used to conduct the ACM ICPC World Finals in 1990 and from 1994 through 2009. In 2010, the ACM ICPC World Finals switched to using Kattis, the KTH automated teaching tool; however, PC<sup>2</sup> continues to be used for a large number of ICPC Regional Contests around the world.

PC<sup>2</sup> manages single or multi-site programming contests. It provides a team a way to log in, test solutions, submit solutions and view judgments from judges. PC<sup>2</sup> provides judges a way to request team solutions (from a PC<sup>2</sup> server) run/execute the solution and enter a judgment. The PC<sup>2</sup> scoreboard module computes and creates standings and statistics web pages (HTML/XML).

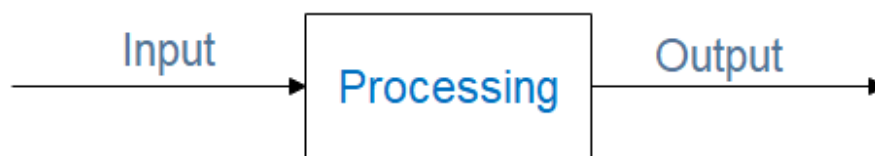
**Kattis:**

Kattis, the KTH (Royal Institute of Technology) Automated Teaching Tool, is a server package to correct ICPC style problems. There are 3 versions of their system (For companies, problem solvers, and for universities). For the first version, Companies version, Kattis automatically screens and evaluates each applicant's technical skills with quick and simple coding challenges. And for the problem solvers version, they offer a wide variety of coding challenges, a great community, a global ranking list and awesome jobs at worldwide companies. The universities version is used by top universities to make better assessments on students work. It corrects the code and assesses code quality.

## *Chapter Three*

### **3 SYSTEM LIFE DEVELOPMENT CYCLE**

A system is a collection of elements or components that are organized for a common purpose. Basically, there are three major components of any system, namely input, processing, and output.



**Figure 3-1:** What is the System?

Systems development is a systematic process which includes phases such as planning, analysis, design, deployment, and maintenance.

In a system different component connected together in order to achieve some goals, and they are independent. For example, the human body represents a completely natural system. Many notional systems such as political system, economic system, educational system and forth bound the process. The objective of the system demands that some output is produced as a result of processing suitable inputs. A well-designed system also includes an additional element referred to as 'control' that provides feedback to achieve the desired objectives of the system.

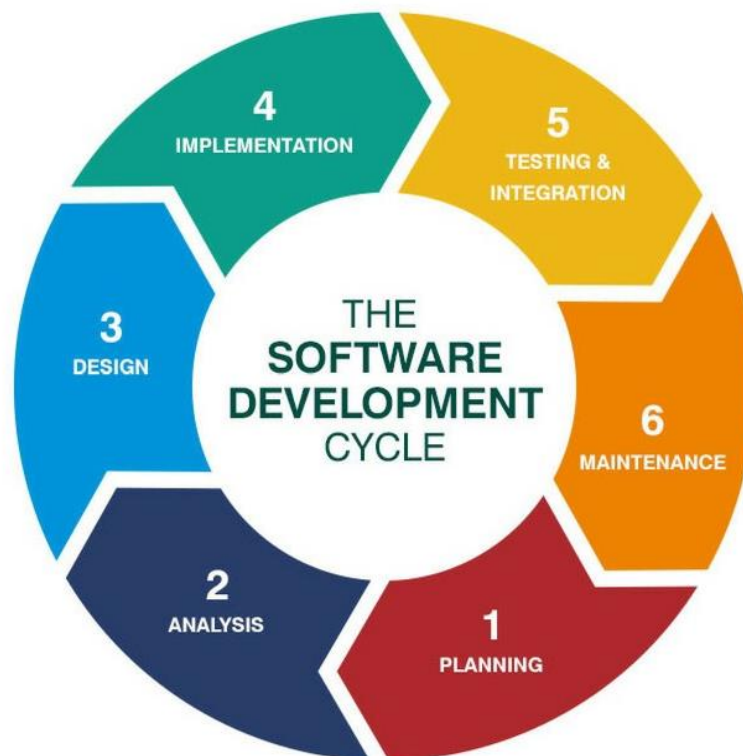
The system life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives an overall list of processes and sub-processes required for developing. System development life cycle means a combination of various activities. In other words, can be regarded that various activities put together as referred to as system development life cycle. In the System Analysis and Design terminology, the system development life cycle also means software development life cycle.

Following are the different phases of system development life cycle:

- Planning
- System Analysis and Requirements



- System Design
- Development
- Integration and Testing
- Implementation
- Operations and Maintenance



**Figure 3-2:** The different phases in the system development life cycle

### 3.1 PLANNING

our project solves the problem of wasting time for students, also save their efforts by organizing solving tasks in an easy way. We have developed our system to solve that problem. Planning involves the preparation of a 'System Proposal' which lists the Problem Definition, Objectives of the System, Resources, Costs, Time, Expected benefits of the new system.... etc. The system proposal is prepared by the system Analyst (who studies the system) and places it before the user management. The management may accept the proposal or request some modifications in the proposal. In summary, the Planning phase passes through the following steps:

- Problem identification and project initiation
- Background analysis
- Inference or findings (system proposal)

## **3.2 ANALYSIS AND REQUIREMENTS**

The second phase is where businesses will work on the source of their problem or the need for a change. In the event of a problem, possible solutions are submitted and analyzed to identify the best fit for the ultimate goal(s) of the project. This is where teams consider the functional requirements of the project or solution. It is also where system analysis takes place or analyzing the needs of the end users to ensure the new system can meet their expectations. Systems analysis is vital in determining what a business's needs are, as well as how they can be met, who will be responsible for individual pieces of the project, and what sort of timeline should be expected. We will discuss system requirements over two types of requirements. We will explain each one.

### **3.2.1 System Requirements**

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behavior of a system or software application. It describes what our System does.

#### **3.2.1.1 Functional System Requirements**

Functional requirements describe the features or attributes of the system. We will explain the Functional System Requirements.

- The system automatically sends results to students or contestants after each submission.
- The system also supports automated judging mode where judging is performed by software rather than by human judges.
- The system automatically timestamps and archives submitted the code.
- The system supports multiple contests being held over the same network.

#### **3.2.1.2 Non-Functional System Requirements**

Non-Functional Requirements System (NFRs) define system attributes such as.

- Availability, System must be available all time.
- Reliability, System must be reliable when the number of teams/users increase.

- Security, System must be secure against intruders.
- Usability, System must be acceptable by its users and its environment.
- Performance
- Maintainability
- Scalability, System must be scalable.

They serve as constraints or restrictions on the design of the system across the different backlogs.

### **3.2.2 User Requirements**

Often referred to as user needs, describe what the user does with the system, such as what activities that users must be able to perform. User requirements are generally documented in a User Requirements Document (URD) using narrative text. User requirements are generally signed off by the user and used as the primary input for creating system requirements.

#### **3.2.2.1 Functional User Requirements**

We will explain the Functional System Requirements.

- Students or contestants can submit programs over a network to contest judges.
- Students or contestants can view results or standings.
- Students or teams can send clarification or any questions to judges.
- Judges can add problems to the contest and add their test cases.
- Judges can view source codes of the contestants/students and their results.
- Judges can send a broadcast message to student or contestants.
- Judges can recompile the submitted program.
- Judges can send answers to the clarifications sent by contestants.
- Students or contestants can view standing.

## **3.3 DESIGN**

The third phase describes, in detail, the necessary specifications, features and operations that will satisfy the functional requirements of the proposed system which will be in place. This is the step for end users to discuss and determine their specific

business information needs for the proposed system. It's during this phase that they will consider the essential components (hardware and/or software) structure (networking capabilities), processing and procedures for the system to accomplish its objectives. Based on the user requirements and the detailed analysis of the existing analysis, the new system must be designed. This is the phase of system designing. It is the most important phase in the developments of a system. The logical system design arrived at as a result of systems analysis is converted into physical system design. Normally the design proceeds into two stages:

- 1- General Design
- 2- Detailed Design

We will explain each one.

**General Design** In general, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated. If the project is still considered to be feasible, the next design stage will be taken into regard.

**Detailed Design** In the detailed design stage, computer-oriented work begins in serious way. At this stage, the design of the system becomes more structured. The structure design is a blueprint of a computer system solution to a given problem having the same component and inter-relationships among the same components as the original problem. Input, output, databases,

forms and processing specifications are drawn up in detail. In the design stage, the programming language and the hardware and software platform in which the new system will run are also decided. There are several tools and techniques used for describing the system design of the system. These tools and techniques are:

- 1- Flow chart
- 2- Data flow diagram (DFD)
- 3- Data dictionary
- 4- Structured English
- 5- Decision table
- 6- Decision tree

The system design involves:

- i. Input requirement.
- ii. Output requirements.
- iii. Storage requirements.
- iv. Processing requirements.

- v. System control and backup or recovery.

Put another way, the physical portion of system design can generally be broken down into three sub-tasks:

- i. User Interface Design.
- ii. Data Design.
- iii. Process Design.

User Interface Design is concerned with how users add information to the system and with how the system presents the information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the system design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

### **3.3.1 DATA FLOW DIAGRAM (DFD)**

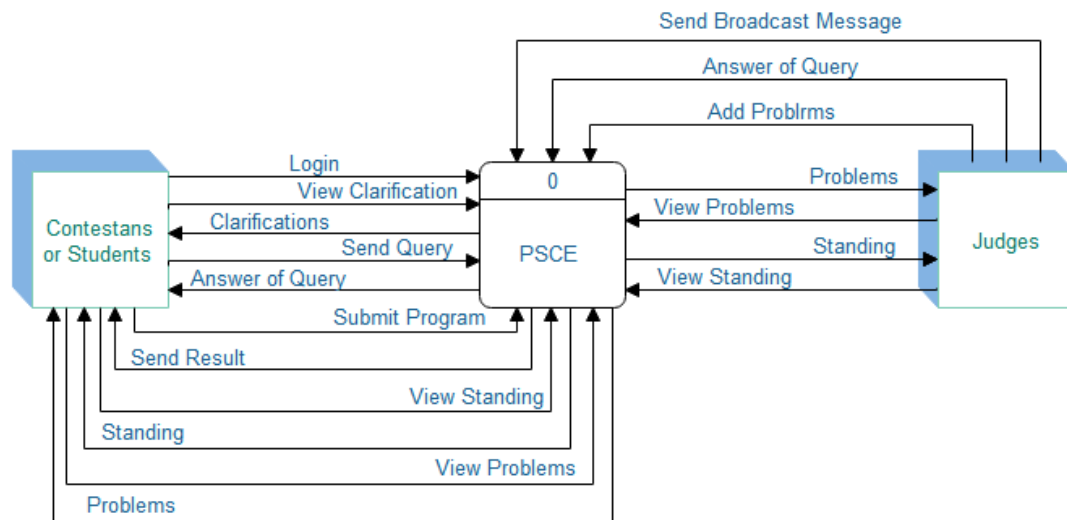
It's easy to understand the flow of data through systems with the right data flow diagram software. DFD represents the activities included in the system. You will understand how the system works through reading its DFD. DFD include some internal diagrams. We will explain it. DFD contains two main diagrams:

- Context diagram.
- Level 0 diagram.

#### **3.3.1.1 Context Diagram**

The context diagram is the highest level in a data flow diagram and contains only one process, representing the entire system. The process is given the number zero. All external entities are shown on the context diagram, as well as a major data flow to and from them. The diagram doesn't contain any data stores and is fairly simple to create, once the external entities and the data flow to and from them are known to analysts.

How the system under consideration as a single is a high-level process and then shows the relationship that the system has with other external entities (systems, organizational groups, external data stores, etc.).

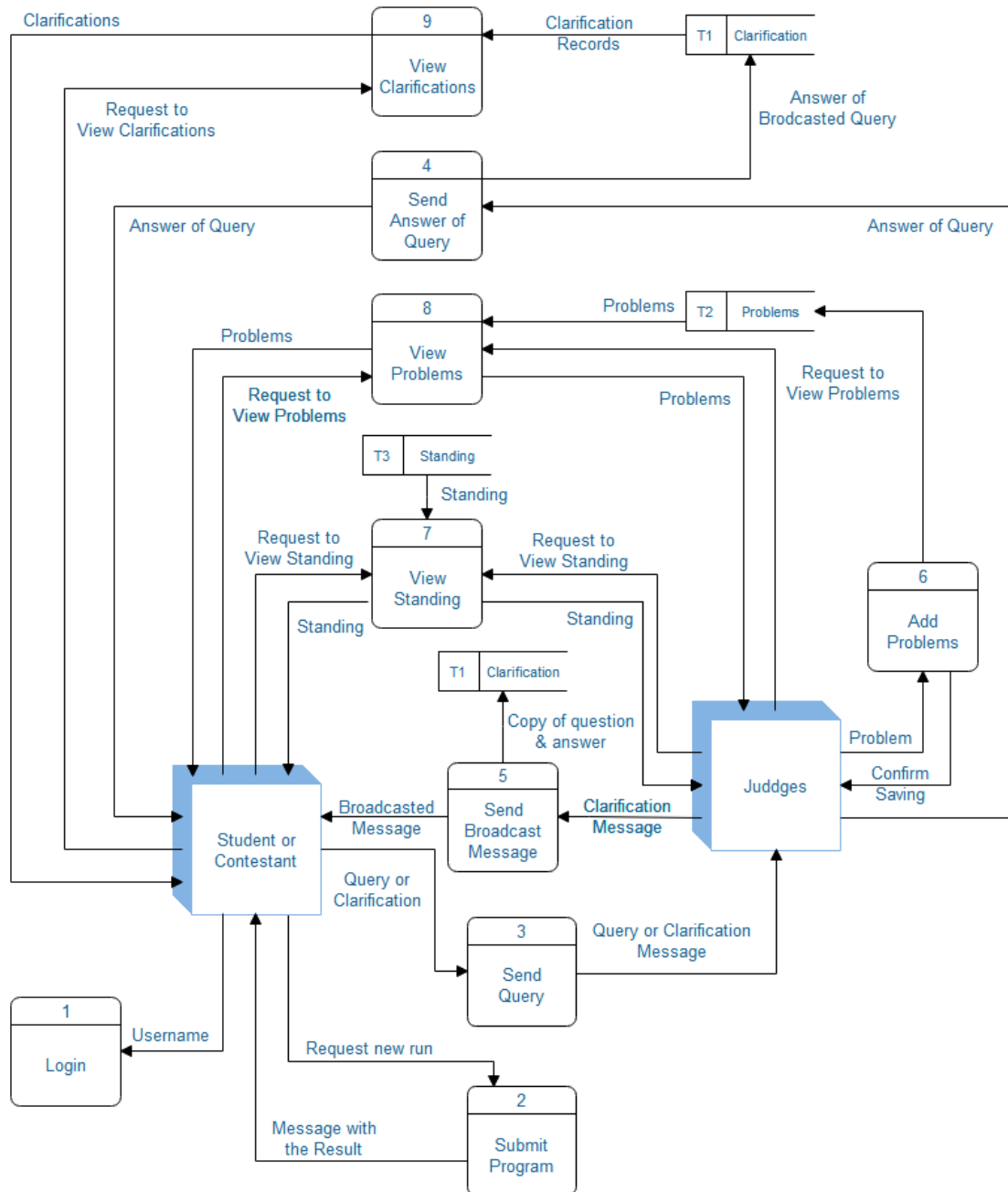


**Figure 3-3:** Data Flow Diagram

As shown in the context diagram, the student or contestants will login into the system then they will be able to do any task they need. Students can any question about any problem they need. Also, can submit any program and send their solutions. They can view all problems exist on the system and view their results. But judges can reply to questions received from students, recompile programs, send broadcast messages to students or contestants, view results to students, add new problems. Through that context diagram, anyone will be able to understand the functions available on the system and also what they can do and what don't.

### 3.3.1.2 Level 0 Diagram

More detail than the context diagram permits the achievable by "exploding the diagrams." Inputs and outputs specified in the first diagram remain constant in all subsequent diagrams. The rest of the original diagram. This abstract representation of the steps that will be executed to achieve a specific task on the system.

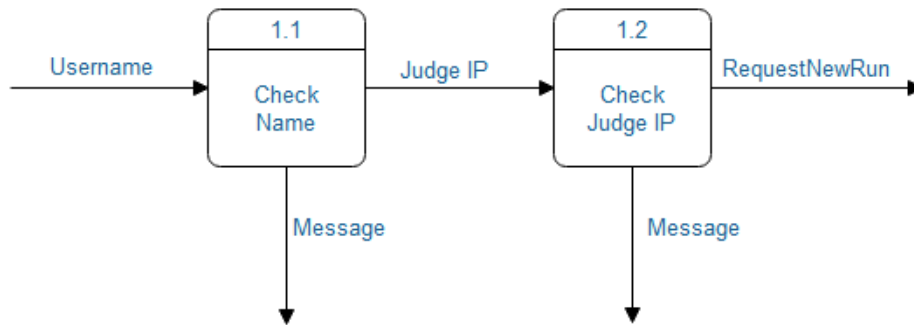


**Figure 3-4:** Level 0 Diagram

The following diagrams include all process from one to three. For example, if a student needs to submit a specific problem, firstly he will log in to the system, if his data is correct and login is done. He will choose a specific problem and click submit, the solution will be sent to judge, when judge receives a solution of the task, will compile it and sent the result back to the student, at the same time the solution and result saved on archive files. We have some internal levels at that level we will explain it through diagrams. Each diagram represents one process but that process has sub-processes:

- Level 1 Diagram
- Level 2 Diagram

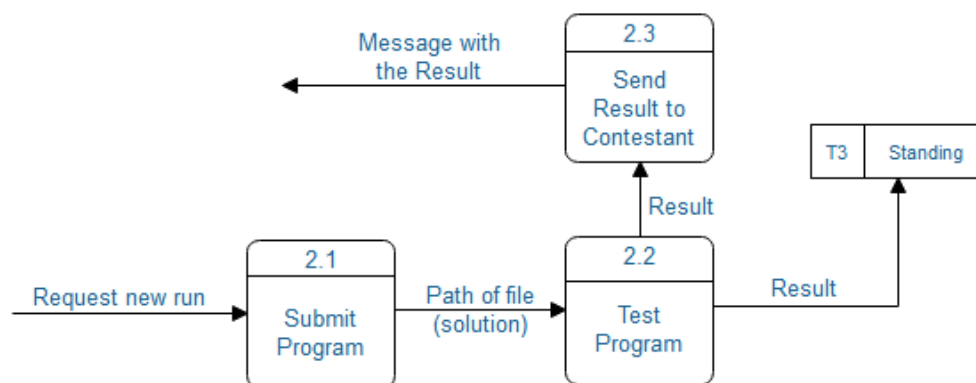
### 3.3.1.3 Level 1 Diagram:



**Figure 3-5:** Level 1 Diagram

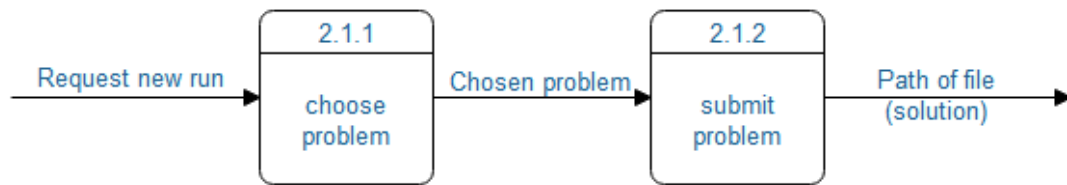
When the student login to the system, his data must be tested and validated. If his data is correct, he will be able to enter the system, if not he will not be able to enter the system.

### 3.3.1.4 Level 2 Diagram:



**Figure 3-6:** Level 2 Diagram



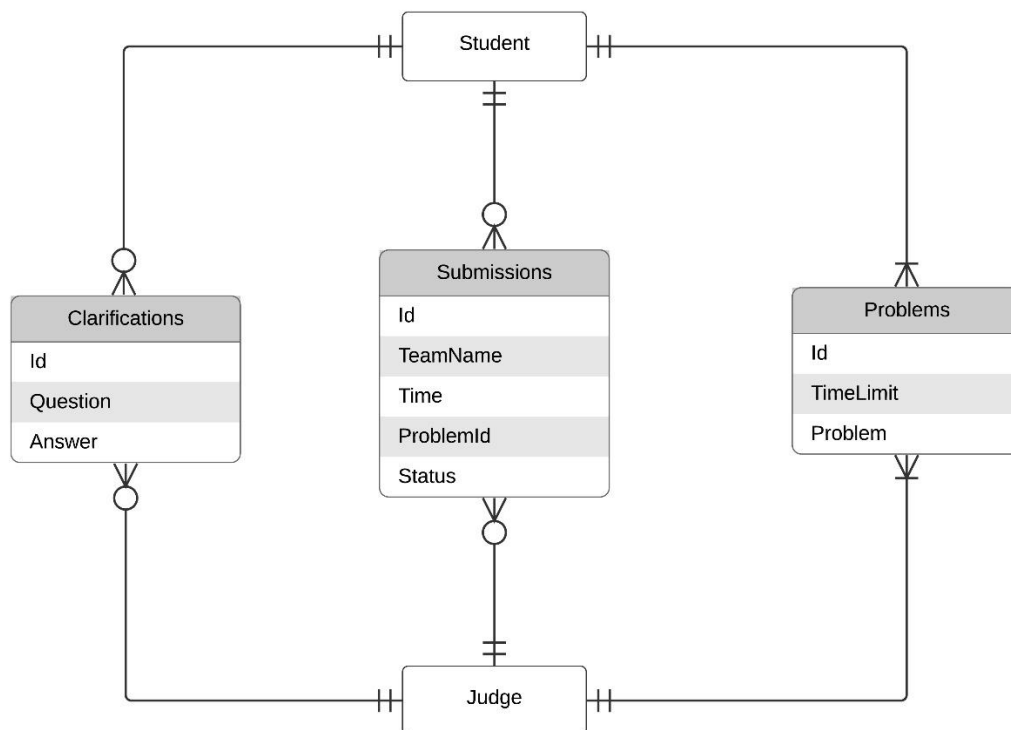


**Figure 3-7:** Level 2.1 Diagram

That diagram will explain the flow of processes will be executed when the student needs to submit a specific problem.

### 3.3.2 ENTITY RELATIONSHIP DIAGRAM (ERD)

An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and can be used as the foundation for the relational database. All entities on the system will be defined on the ERD diagram, also define the interaction between entities on the system.



**Figure 3-8:** Entity-Relationship Diagram

At this diagram, entities on our system are Contestants or Students and Judges, objectives are Problems and Results, this defines the relationships between entities and objectives. For example, one student can submit a problem or more, also one student can have more than one result.

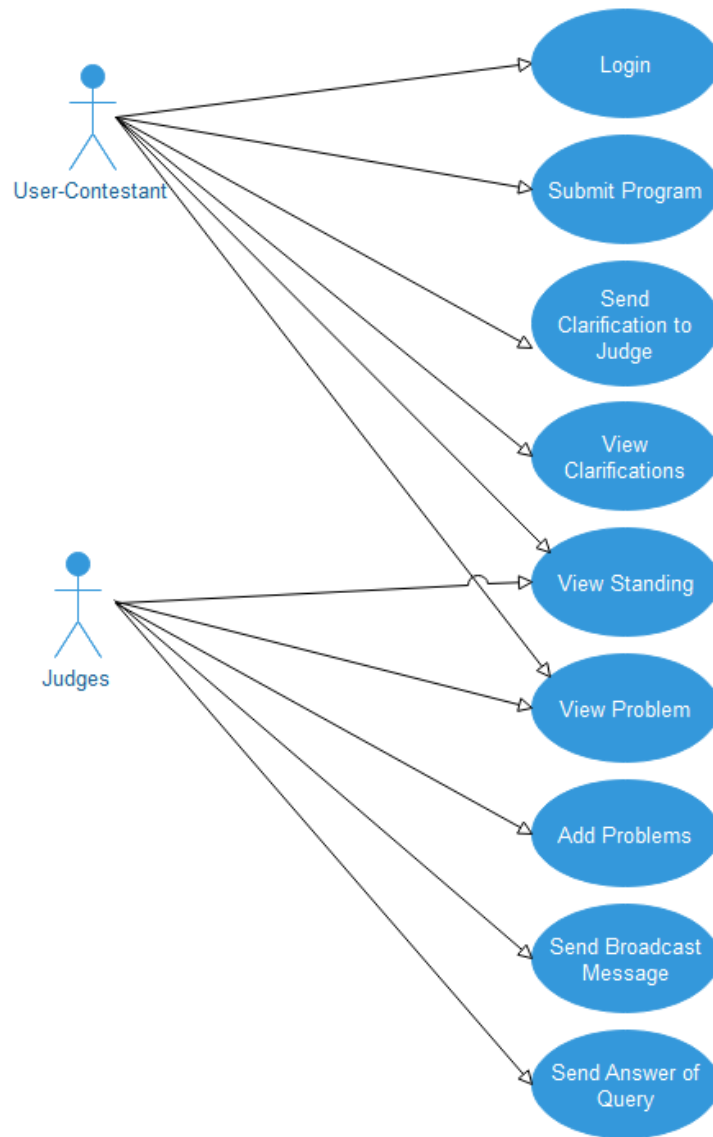
### **3.3.3 UNIFIED MODELING LANGUAGE (UML)**

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. UML is a way of visualizing a software program using a collection of diagrams. UML has two main diagrams:

- Use Case Diagram.
- Sequence Diagram.

#### **3.3.3.1 Use Case Diagram**

Use case diagrams are considered for high-level requirement analysis of a system. So, when the requirements of a system are analyzed the functionalities are captured in use cases. This diagram is a graphic illustration of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. Use case diagrams are drawn to capture the functional requirements of a system.



**Figure 3-9:** Use Case Diagram

As shown in the figure, the system provides a number of functions such as:

- View Problems.
- Add Problems.
- View Standing.
- Submit Program.
- Send Clarification to judges.
- View Clarifications.
- Send a broadcast message.
- Send a reply of clarification.

As we explained before, how the student can interact with the system.

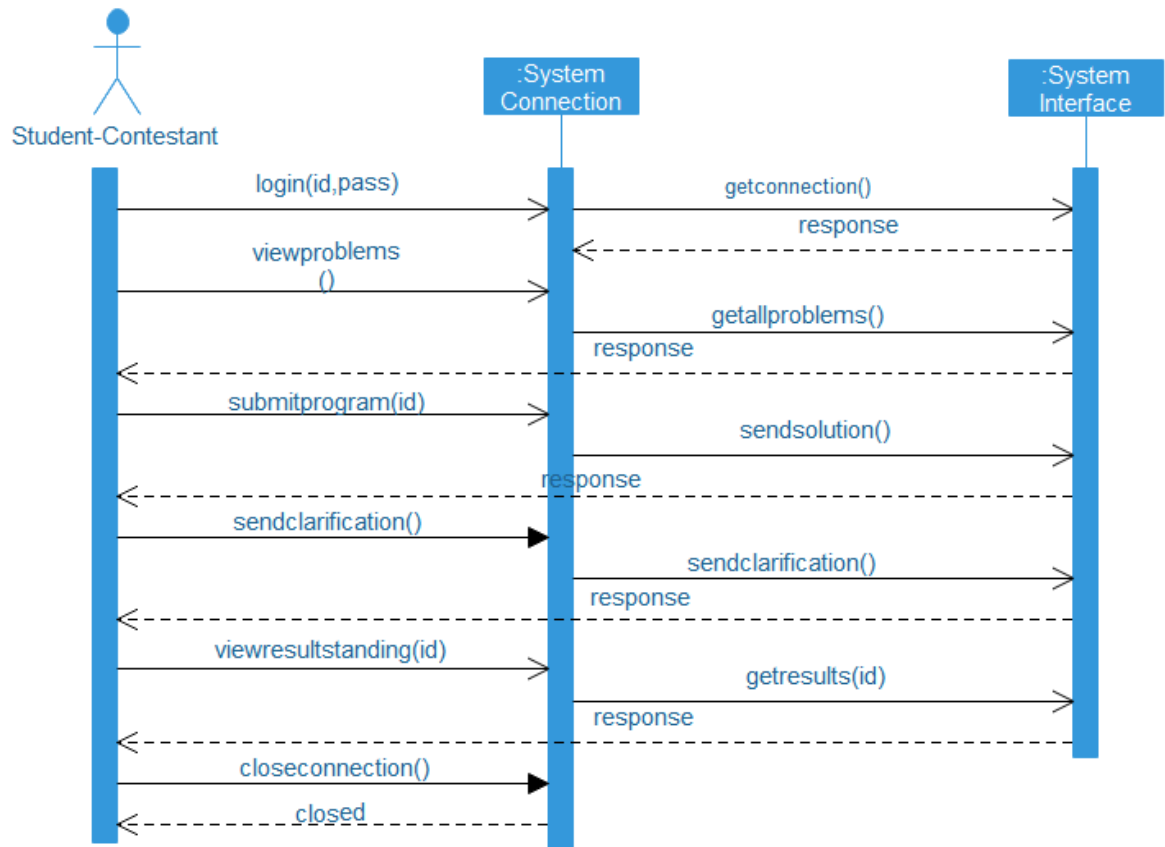
### **3.3.3.2 Sequence Diagram**

UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom (i.e. Lower equals Later).

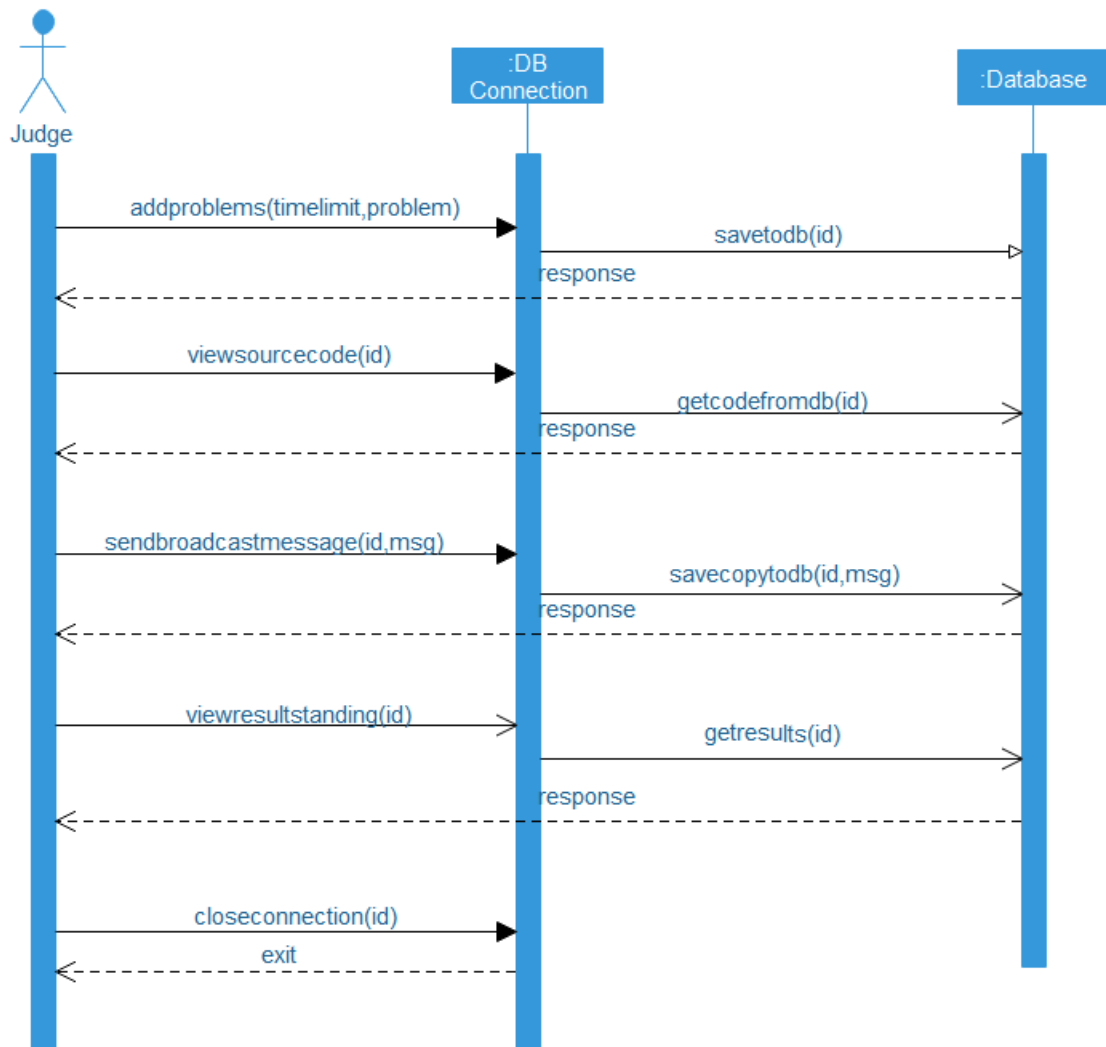
A popular use for them is to document the dynamics in an object-oriented system. For each key collaboration, diagrams are created to show how objects interact in various representative scenarios for that collaboration.

The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them. However, an organization's business staff can find sequence diagrams useful to communicate how the business currently works by showing how various business objects interact. Besides documenting an organization's current affairs, a business-level sequence diagram can be used as a requirements document to communicate requirements for future system implementation.

During the requirements phase of a project, analysts can make use of cases to the next level by providing a more formal level of refinement. When that occurs, use cases are often refined into one or more sequence diagrams. Each entity has a sequence diagram, we will explain each one.



**Figure 3-10:** Contestant Sequence Diagram



**Figure 3-11:** Judge Sequence Diagram

This explains the flow of the processes through the system, in a specific way, in the top of diagram starts the interaction with the system, firstly the judge or the student login to the system, then he will be able to interact into the system. The judge only interacts with the database and can change it.

### 3.4 DEVELOPMENT

The fourth phase is when the real work begins in particular, when a programmer, network engineer and/or database developer are brought on to do the major work on the project. This work includes using a flow chart to ensure that the process of the system is properly organized. The development phase marks the end of the initial section of

the process. Additionally, this phase signifies the start of production. The development stage is also characterized by installation and change. Focusing on training can be a huge benefit during this phase. The system design needs to be implemented to make it a workable system. This demands the development of design into computer understandable language, i.e., programming language. This also called the programming phase in which the programmer converts the program specifications into computer instructions. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature this helps in fast development, maintenance and future changes, if required. Once developers reach an understanding of the end user's requirements, the actual product must be developed. The Development phase can be considered one of conceptual design and may involve:

1. Defining the elements required for the system
2. Considering numerous components such as security level, modules, architecture, interfaces and types of data that the system will support
3. Identifying and evaluating design alternatives
4. Developing and delivering design specifications

We have to take in consideration, Security considerations:

- Conducting risk assessments.
- Testing for system function and security.
- Preparing initial documents for system certification.
- Designing security architecture.
- Developing security plans.

### **3.5 INTEGRATION AND TESTING**

The fifth phase involves systems integration and system testing of programs and procedures normally carried out by a Quality Assurance professional to determine if the proposed design meets the initial set of business goals. Testing may be repeated, specifically to check for errors, bugs and interoperability. This testing will be performed until the end user finds it acceptable. Another part of this phase is verification and validation, both of which will help ensure the program's successful completion. Before actually implementing the new system into operation, a test run of the system is done for removing the bugs, if any. It is an important phase of a successful system. After developing the whole programs of the system, a test plan should match the expected requirements. Sometimes, system testing is considered a part of the implementation process.

Using the test data following test run are carried out:

- Program test
- System test

### **3.5.1 Program Test**

When the programs have been coded, compiled and brought to working conditions, they must be individually tested with the prepared test data. Any errors must be noted and solved at that phase. Next step, all programs will be aggregated to form the whole system. Then will apply System Test phase on the whole system.

### **3.5.2 System Test**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects) and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

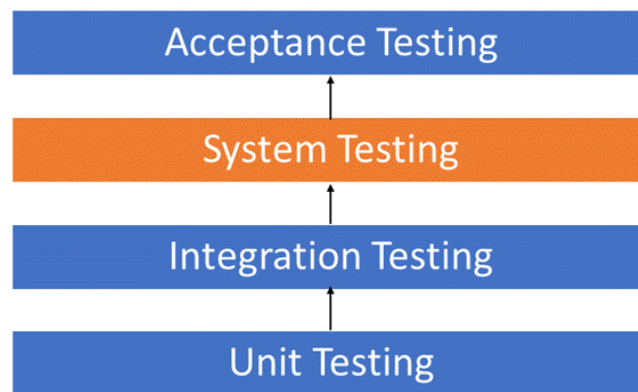
- Meets the requirements that guided its design and development.
- Responds correctly to all kinds of inputs.
- Performs its functions within an acceptable time.
- Sufficiently usable.
- Can be installed and run in its intended environments.
- Achieves the general result of its stakeholders' desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists.





**Figure 3-12:** System Test Types

### **3.5.3 Testing Methods**

#### **3.5.3.1 Static vs. dynamic testing**

There are many approaches available in software testing. Reviews or inspections are referred to as static testing, whereas executing programmed code with a given set of test cases is referred to as dynamic testing.

Static testing is often implicit, it takes place when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis.

Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete to test sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment. Static testing involves verification, whereas dynamic testing involves validation.

Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test cases will detect errors which are introduced by mutating the source code.

#### **3.5.3.2 Box Approaches**

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

##### **3.5.3.2.1 White-box testing**

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-

user. In white box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is similar to testing nodes in a circuit, e.g. in-circuit testing.

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

- API testing – testing of the application using public and private APIs (application programming interfaces).
- Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once).
- Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies.
- Mutation testing methods.
- Static testing methods.
- Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Code coverage as a software metric can be reported as a percentage for:

- Function coverage, which reports on functions executed.
- Statement coverage, which reports on the number of lines executed to complete the test.
- Decision coverage, which reports on whether both the True and the False branch of a given test has been executed.

100% statement coverage ensures that all code paths or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly.

### **3.5.3.2.2 Black-box testing**

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it.

Black-box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements.

This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels but can also dominate unit testing as well.

### **3.5.3.2.3 Grey-box testing**

Grey-box testing involves having knowledge of internal data structures and algorithms for purposes of designing tests, while executing those tests at the user, or black-box level. The tester is not required to have full access to the software's source code. [not in citation given] Manipulating input data and formatting output does not qualify as grey-box, because the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed to the test.

However, tests that require modifying a back-end data repository such as a database or a log file does qualify as grey-box, as the user would not normally be able to change the data repository in normal production operations. [citation needed] Grey-

box testing may also include reverse engineering to determine, for instance, boundary values or error messages.

By knowing the underlying concepts of how the software works, the tester makes better-informed testing choices while testing the software from outside. Typically, a grey-box tester will be permitted to set up an isolated testing environment with activities such as seeding a database. The tester can observe the state of the product being tested after performing certain actions such as executing SQL statements against the database and then executing queries to ensure that the expected changes have been reflected. Grey-box testing implements intelligent test scenarios, based on limited information. This will particularly apply to data type handling, exception handling, and so on.

In our project, we used **White-box testing** (also known as clear box testing, glass box testing, transparent box testing and structural testing, by seeing the source code).

### 3.6 IMPLEMENTATION

The sixth phase is when the majority of the code for the program is written. Additionally, this phase involves the actual installation of the newly developed system. This step puts the project into production by moving the data and components from the old system and placing them in the new system via a direct cutover. While this can be a risky 'and complicated' move, the cutover typically happens during off-peak hours, thus minimizing the risk. Both system analysts and end-users should now see the realization of the project that has implemented changes. After having the user acceptance of the new system developed, the implementation phase begins. Implementation is the stage of a project during which the theory is turned into practice.

The major steps involved in this phase are:

- Acquisition and Installation of Hardware and Software.
- Conversion.
- User Training.
- Documentation.

The hardware and the relevant software required for running the system must be made fully operational before implementation. The conversion is also one of the most critical and expensive activities in the system development life cycle. The data from the old system needs to be converted to operate in the new format of the new system. The database needs to be set up with security and recovery procedures fully defined.

During this phase, all the programs of the system are loaded onto the user's computer. After loading the system training of the user starts. Main topics of such type of training are:

- How to execute the package.
- How to enter the data.
- How to process the data (processing details).
- How to take out the reports.

### **3.7 OPERATIONS AND MAINTENANCE**

The seventh and final phase involves maintenance and regular required updates. This step is when end users can fine-tune the system, if they wish, to boost performance, add new capabilities or meet additional user requirements. Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environment. It has been seen that there are always some errors found in the systems that must be noted and corrected. It also means a review of the system from time to time. The review of the system is done for:

- Knowing the full capabilities of the system.
- Knowing the required changes or additional requirements.
- Studying performance.

If a major change to a system is needed, a new project may have to be set up to carry out the change. The new project will then proceed through all the above life cycle phases. Operations and maintenance are combined into the common term O&M because a facility cannot operate at peak efficiency without being maintained; therefore, the two are discussed as one. Operations and maintenance typically include the day-to-day activities necessary for the building and its systems and equipment to perform their intended function. Next figure will explain the steps involved in the O&M phase.



**Figure 3-13:** Steps of Operations and Maintenance Phase

### 3.8 WHY SDLC PHASES?

If a user determines a change is needed during any phase of the SDLC, the developers might have to proceed through all the above life cycle phases again. The life cycle approach of any project is a time-consuming process. Even though some steps are more difficult than others, none are to be overlooked. An oversight could prevent the entire system from functioning as planned. A System (or Software) Development Lifecycle (SDLC) is useful for managing a planned and controlled development effort. It provides some level of control of the development process to ensure that the ultimate solution is consistent with the original requirements and to ensure that the design process and testing process leading to release of a solution is sound and well-managed. It is a repeatable process. If you develop something with a given SDLC and a similar project comes along, you should be able to use the same process with some level of confidence of success. If you develop something with a given SDLC and a similar project comes along, you should be able to use the same process with some level of confidence of success. Planning and Analysis is the most important stage of SDLC. Many companies spend high cost on these two phases and in the end, they fail because they didn't plan and analyze the product that was needed to invent. Spending more time on these two phases would decrease the failing over time and reduce expenses by producing a needy product or service. This stage must not be ignored to generate more business and desired results.

## 4 PSCE TOOLS

### 4.1 ALGORITHMS

#### 4.1.1 Minimum Edit Distance

The **minimum edit distance** between two strings is the minimum number of editing operations: Insertion, Deletion, Substitution needed to transform one into the other. We used this algorithm in the *detection of cheating or plagiarism*.

Given two strings  $a$  and  $b$  on an alphabet  $\Sigma$  (e.g. the set of ASCII characters, the set of bytes [0 to 255], etc.), the edit distance  $d(a, b)$  is the minimum-weight series of edit operations that transforms  $a$  into  $b$ . One of the simplest sets of edit operations is defined by Levenshtein in 1966.

- Insertion of a single symbol. If  $a = uv$ , then inserting the symbol  $x$  produces  $uxv$ . This can also be denoted  $\varepsilon \rightarrow x$ , using  $\varepsilon$  to denote the empty string.
- Deletion of a single symbol changes  $uxv$  to  $uv$  ( $x \rightarrow \varepsilon$ ).
- Substitution of a single symbol  $x$  for a symbol  $y \neq x$  changes  $uxv$  to  $uyv$  ( $x \rightarrow y$ ).

In Levenshtein's original definition, each of these operations has unit cost (except that substitution of a character by itself has zero cost), so the Levenshtein distance is equal to the minimum number of operations required to transform  $a$  to  $b$ . A more general definition associates non-negative weight functions  $w_{\text{ins}}(x)$ ,  $w_{\text{del}}(x)$  and  $w_{\text{sub}}(x, y)$  with the operations.

Using Levenshtein's original operations, the edit distance between  $a = a_1 \dots a_n$  and  $b = b_1 \dots b_m$  is given by  $d_{mn}$ , defined by the recurrence.

$$\begin{aligned}
 d_{i0} &= \sum_{k=1}^i w_{\text{del}}(b_k), & \text{for } 1 \leq i \leq m \\
 d_{0j} &= \sum_{k=1}^j w_{\text{ins}}(a_k), & \text{for } 1 \leq j \leq n \\
 d_{ij} &= \begin{cases} d_{i-1,j-1} & \text{for } a_i = b_j \\ \min \begin{cases} d_{i-1,j} + w_{\text{del}}(a_i) \\ d_{i,j-1} + w_{\text{ins}}(b_j) \\ d_{i-1,j-1} + w_{\text{sub}}(a_i, b_j) \end{cases} & \text{for } a_i \neq b_j \end{cases} & \text{for } 1 \leq i \leq m, 1 \leq j \leq n.
 \end{aligned}$$

Figure 4-1: Minimum Edit Distance Algorithm

Edit distance finds applications in computational biology and natural language processing, e.g. the correction of spelling mistakes or OCR errors, and approximate string matching, where the objective is to find matches for short strings in many longer texts, in situations where a small number of differences is to be expected.

### 4.1.2 Balanced Binary Search Tree

A balanced binary search tree is a tree that automatically keeps its height small (guaranteed to be logarithmic) for a sequence of insertions and deletions. This structure provides efficient implementations for abstract data structures such as associative arrays.

The key reason why a BST offers such great performance is that it allows us to ignore irrelevant values. Thus decreasing the number of comparisons a program must perform to find a data element.

There are many common algorithms to convert trees that are not balanced. Some of the most popular algorithms are: Red-black trees, AVL trees. Both of these data structures force a tree to remain balanced and therefore can guarantee search performance.

### 4.1.3 Red-Black Tree

A red-black tree is a binary search tree used to implement 2–3 tree with very basic code. The idea is to have a tree like the binary search tree, but balanced, and also like the 2–3 tree, but each node has one key and two links.

So, to represent a red-black BST tree, we start by a standard BST tree that represents 2–3 tree by encoding the 3-nodes. We encode a 3-node as two 2-nodes connected by a single *red* link that leans left.

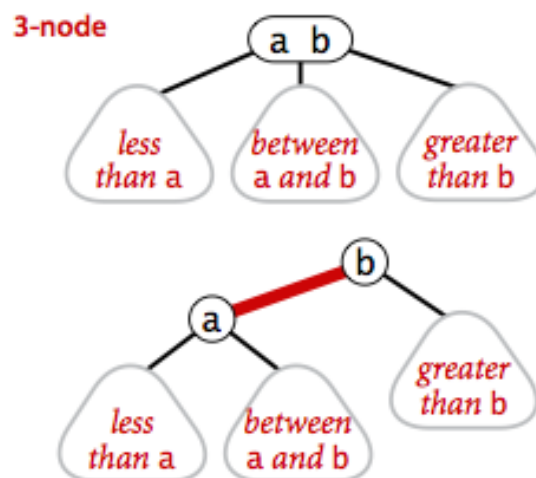
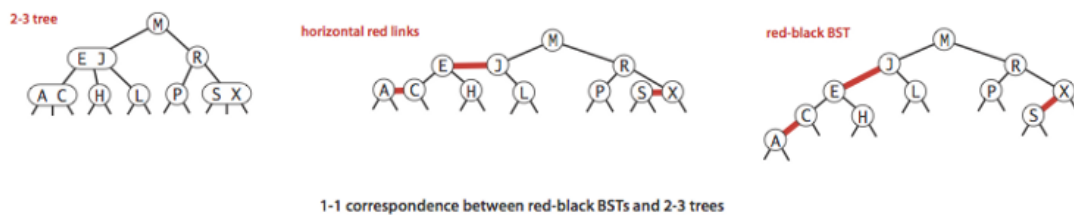


Figure 4-2: Encoding 3-nodes



This way, we can maintain a 1–1 correspondence with 2–3 tree; meaning, given any 2–3 tree, we can immediately derive a corresponding red-black BST, and vice-versa.



**Figure 4-3:** 1–1 correspondence between red-black BSTs and 2–3 tree

Properties of Red-Black BSTs:

- Red links act as a glue for 3-nodes.
- No node has two red links connected to it.
- Every path from the root to null has the same number of black links; **perfect black balance**.

Now, we are going to show how to implement the symbol tables using a red-black binary search tree. So, the main operations of would be the same; *get*, *put*, *delete*, *contains* & *isEmpty*.

## Implementation

We’ll start by the red-black BST class. It has a reference variable called “root” points to the root node (same as BSTs).

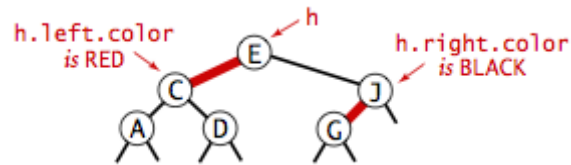
The Key and Value type is any generic type. And, for keys, they are *Comparable*. So, we can use *compareTo()* method, to compare whether one key is less than the other, or greater than, or equal.

In addition, we have two static boolean variables; *RED* and *BLACK*, which represent the values of red and black links.

```
public class RedBlackBST<Key extends Comparable<Key>, Value> {
    private Node root;        // root of the BST

    private static final boolean RED    = true;
    private static final boolean BLACK = false;
}
```

Then, we have an inner class called “Node” to represent each node within the tree. Each node has a key, value, left and right reference to child nodes, and the color of the link connected to its parent.



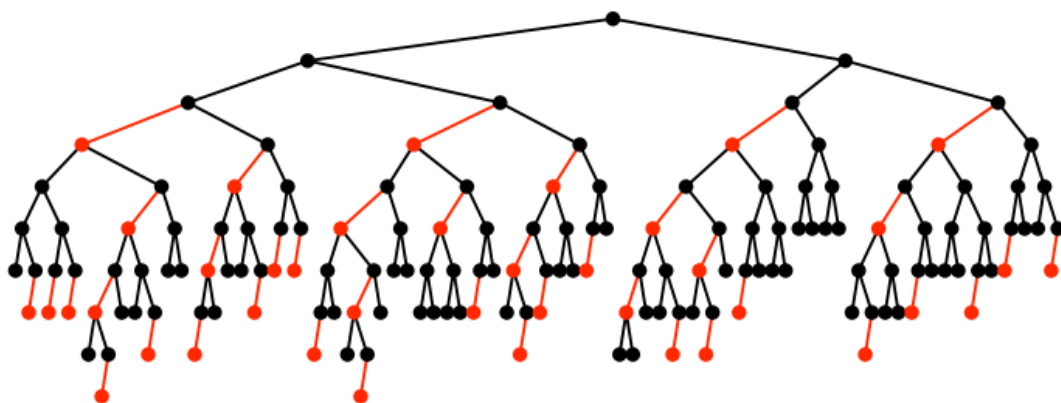
**Figure 4-4:** Color representation in red-black BSTs

```
private class Node {
    private Key key;           // sorted by key
    private Value val;         // associated data
    private Node left, right;  // left and right subtrees
    private boolean color;     // color of parent link
    private int count;         // number of nodes in the subtree

    public Node(Key key, Value val, boolean color, int count) {
        this.key = key;
        this.val = val;
        this.color = color;
        this.count = count;
    }
}

private boolean isRed(Node x) {
    if (x == null) return false; // null links are black
    return x.color == RED;
}
```

Search and insert operations in a red-black BST are guaranteed to take  $(2 \log N)$  at most.



**Figure 4-5:** Search and insert operations

#### **4.1.4 Hashing**

Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for faster access of elements. The efficiency of mapping depends on the efficiency of the hash function used. In our project, we used hashing in for Standing.

#### **4.1.5 Quicksort**

Quicksort is an efficient sorting algorithm, serving as a systematic method for placing the elements of a random-access file or an array in order. Developed by British computer scientist Tony Hoare in 1959 and published in 1961, it is still a commonly used algorithm for sorting. When implemented well, it can be about two or three times faster than its main competitors, merge sort and heapsort.

Quicksort is a comparison sort, meaning that it can sort items of any type for which a "less-than" relation (formally, a total order) is defined. In efficient implementations it is not a stable sort, meaning that the relative order of equal sort items is not preserved. Quicksort can operate in-place on an array, requiring small additional amounts of memory to perform the sorting. It is very similar to selection sort, except that it does not always choose worst-case partition.

Mathematical analysis of quicksort shows that, on average, the algorithm takes  $O(n \log n)$  comparisons to sort  $n$  items. In the worst case, it makes  $O(n^2)$  comparisons, though this behavior is rare.

### **4.2 TECHNOLOGIES**

#### **4.2.1 Socket Programming in Java**

Java Socket programming is used for communication between the applications running on different JRE (Java Runtime Environment). It can be connection-oriented or connectionless.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket. The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

### 4.2.2 System Calls in Java

A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel. System call provides the services of the operating system to the user programs via Application Program Interface (API).

It involves the use of two Java classes, the Runtime class and the Process class. Basically, you use the exec method of the Runtime class to run the command as a separate process. Invoking the exec method returns a Process object for managing the subprocess.

We used the system calls to handle the submissions of the students during the contest.

### 4.2.3 TeX

TeX is a typesetting system which was designed and mostly written by Donald Knuth and released in 1978. TeX is a popular means of typesetting complex mathematical formulae. It has been noted as one of the most sophisticated digital typographical systems. TeX is popular in academia, especially in mathematics, computer science, economics, engineering, linguistics, physics, statistics, and quantitative psychology. TeX was designed with two main goals in mind: to allow anybody to produce high-quality books using minimal effort, and to provide a system that would give exactly the same results on all computers, at any point in time.

<i>Markup</i>	<i>Renders as</i>
The quadratic formula is $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	The quadratic formula is $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

### 4.2.4 MathJax

MathJax is an open-source JavaScript display engine for LaTeX, MathML, and AsciiMath notation that works in all modern browsers. It was designed with the goal of consolidating the recent advances in web technologies into a single, definitive, math-on-the-web platform supporting the major browsers and operating systems. It requires no setup on the part of the user (no plugins to download or software to install), so the

page author can write web documents that include mathematics and be confident that users will be able to view it naturally and easily. Simply include MathJax and some mathematics on a web page, and MathJax does the rest.

Some of the main features of MathJax include:

- High-quality display of LaTeX, MathML, and AsciiMath notation in HTML pages
- Supported in most browsers with no plug-ins, extra fonts, or special setup for the reader
- Easy for authors, flexible for publishers, extensible for developers
- Supports math accessibility, cut-and-paste interoperability, and other advanced functionality
- Powerful API for integration with other web applications

Most MathJax symbols attempt to mimic in text what they look like rendered, like oo for  $\infty$ . Many symbols can also be displayed using a TeX alternative.

**Table 1:** Operation symbols

Type	TeX alt	See
+		+
-		-
*	cdot	·
**	ast	*
***	star	★
//		/
\	backslash setminus	\
Xx	times	×
÷	div	÷
><	ltimes	⋈
><	rtimes	⋈
><	bowtie	⋈
@	circ	◦
o+	oplus	⊕
Ox	otimes	⊗
o.	odot	⊙
Sum		Σ
prod		Π
^^	wedge	∧
^^^	bidwedge	∧
Vv	vee	∨
Vvv	bigvee	∨
Nn	cap	∩
Nnn	bigcap	∩
Uu	cup	∪
Uuu	bigcup	∪

**Table 2:** Miscellaneous symbols

ype	TeX alt	See
2/3	$\frac{2}{3}$	23
2^3		23
sqrt x		$\sqrt{x}$

ype	TeX alt	See
root(3)(x)		$\sqrt[3]{x}$
int		∫
oint		∮
del	Partial	∂
Grad	Nabla	∇
+ -	Pm	±
O/	Emptyset	∅
Oo	Infty	∞
Aleph		ℵ
∴	Therefore	∴
∵	Because	∵
...	ldots	...
cdots		...
Vdots		⋮
Ddots		⋱
quad		
/_	Angle	∠
Frown		⌒
/_ \	Triangle	△
diamond		◇
Square		□
_	Lfloor	⌊
_	Rfloor	⌋
~	Lceiling	⌈
~	Rceiling	⌉
CC		ℂ
NN		ℕ
QQ		ℚ
RR		ℝ
ZZ		ℤ
"hi"	text(hi)	hi

**Table 3:** Relation symbols

Type	TeX alt	See
=		=
!=	ne	≠
<	lt	<
>	gt	>
<=	le	≤
>=	ge	≥
-<	prec	<
-<=	preceq	≤
>-	succ	>
>=	succeq	≥
In		∈
!in	notin	∉
sub	subset	⊂
sup	supset	⊃
sube	subseteq	⊆
supe	supseteq	⊇
-=	equiv	≡
~=	cong	≅
~~	approx	≈
prop	propto	∝

**Table 4:** Logical symbols

Type	TeX alt	See
And		andand
Or		oror
not	Neg	¬
=>	implies	⇒
If		If
<=>	lff	⇔
AA	forall	∀
EE	exists	∃

Type	TeX alt	See
_ _	Bot	⊥
TT	Top	⊤
--	Vdash	⊢
==	Models	⊨

**Table 5:** Grouping brackets

Type	TeX alt	See
(		(
)		)
[		[
]		]
{		{
}		}
(:	langle	⟨
:)	rangle	⟩
<<		⟨
>>		⟩
{: x )		x)
( x :}		(x
abs(x)		x
floor(x)		⌊x⌋
ceil(x)		⌈x⌉
norm(vecx)		x→

**Table 6:** Arrows

Type	TeX alt	See
Uarr	Uparrow	↑
Darr	downarrow	↓
Rarr	rightarrow	→
->	To	→
>->	rightarrowtail	↘
->>	twoheadrightarrow	⇒
>->>	twoheadrightarrowtail	↗

Type	TeX alt	See
->	Mapsto	$\mapsto$
Larr	Leftarrow	$\leftarrow$
Harr	leftrightarrow	$\leftrightarrow$
rArr	Rightarrow	$\Rightarrow$
lArr	Leftarrow	$\Leftarrow$
hArr	Leftrightarrow	$\Leftrightarrow$

**Table 7:** Accents

Type	TeX alt	See
hat x		$\hat{x}$
bar x	overline x	$\bar{x}$
ul x	underline x	$\underline{x}$
vec x		$\vec{x}$ $\rightarrow$
dot x		$\dot{x}$
ddot x		$\ddot{x}$
overset(x)(=)	overset(x)(=)	$\overset{x}{=}$
underset(x)(=)		$\underset{x}{=}$
ubrace(1+2)	underbrace(1+2)	$\underbrace{\quad}$
obrace(1+2)	overbrace(1+2)	$\overbrace{\quad}$
color(red)(x)		$x$
cancel(x)		$\cancel{x}$

**Table 8:** Greek Letters

Type	See	Type	See
Alpha	$\alpha$		

Type	See	Type	See
Beta	$\beta$		
Gamma	$\gamma$	Gamma	$\Gamma$
Delta	$\delta$	Delta	$\Delta$
Epsilon	$\epsilon$		
varepsilon	$\epsilon$		
Zeta	$\zeta$		
Eta	$\eta$		
theta	$\theta$	Theta	$\Theta$
vartheta	$\vartheta$		
iota	$\iota$		
kappa	$\kappa$		
Lambda	$\Lambda$	Lambda	$\Lambda$
Mu	$\mu$		
Nu	$\nu$		
Xi	$\xi$	Xi	$\Xi$
Pi	$\pi$	Pi	$\Pi$
Rho	$\rho$		
Sigma	$\Sigma$	Sigma	$\Sigma$
Tau	$\tau$		
Upsilon	$\Upsilon$		
Phi	$\Phi$	Phi	$\Phi$
Varphi	$\varphi$		
Chi	$\chi$		
Psi	$\Psi$	Psi	$\Psi$
Omega	$\omega$	Omega	$\Omega$

## Standard Functions

sin, cos, tan, sec, csc, cot, arcsin, arccos, arctan, sinh, cosh, tanh, sech, csch, coth, exp, log, ln, det, dim, mod, gcd, lcm, lub, glb, min, max, f, g.



## Special Cases

Matrices: `[[a,b],[c,d]]` yields to  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

Column vectors: `((a),(b))` yields to  $\begin{pmatrix} a \\ b \end{pmatrix}$

Matrices can be used for layout: `{(2x,+,17y,=,23),(x,-,y,=,5):}` yields 
$$\begin{cases} 2x + 17y = 23 \\ x - y = 5 \end{cases}$$

Complex subscripts: `lim_(N->oo) sum_(i=0)^N` yields to  $\lim_{N \rightarrow \infty} \sum_{i=0}^N$

Subscripts must come before superscripts: `int_0^1 f(x)dx` yields to  $\int_0^1 f(x)dx$

Derivatives: `f'(x) = dy/dx` yields  $f'(x) = \frac{dy}{dx}$

For variables other than x,y,z, or t you will need grouping symbols: `(dq)/(dp)` for  $\frac{dq}{dp}$

Overbraces and underbraces: `ubrace(1+2+3+4)("4 terms")` yields  $\underbrace{1 + 2 + 3 + 4}_{4 \text{ terms}}$ .

`obrace(1+2+3+4)^("4 terms")` yields  $\overbrace{1 + 2 + 3 + 4}^{4 \text{ terms}}$ .

## 4.3 DESKTOP APPLICATION

### 4.3.1 Java

Java is an object-oriented programming language developed at Sun Microsystems. It is now owned, along with the rest of Sun Microsystems, by Oracle. It is, by most accounts, one of the most frequently used programming languages around, and so the skills are available in many individuals offering their services to enterprises.

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them.

### **4.3.2 FXML**

FXML is an XML-based user interface markup language created by Oracle Corporation for defining the user interface of a JavaFX application.

It provides a convenient alternative to constructing such graphs in procedural code and is ideally suited to defining the user interface of a JavaFX application, since the hierarchical structure of an XML document closely parallels the structure of the JavaFX scene graph. However, anything that is created or implemented in FXML can be expressed using JavaFX directly.

### **4.3.3 MySQL**

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius' daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available and offer additional functionality.

MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Applications that use the MySQL database include TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, and Drupal. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flickr, and YouTube

## **4.4 WEB SITE**

### **4.4.1 HTML**

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `<img />` and `<input />` introduce content into the page directly. Others such as `<p>...</p>` surround and provide information about document text and may include

other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), the maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

#### **4.4.2 CSS**

CSS or Cascading Style Sheets is rather a markup language. When paired with HTML, CSS allows a developer to decide and define how a web page or a website will eventually look or how it will appear to the visitors of the web platform. Some of the elements which CSS has an impact on include font size, font style, the overall layout, the colors and other design elements. This is a markup language that can be applied to several types of documents including Plain XML documents, SVG documents as well as XUL documents. For most websites across the world, CSS is the platform to opt for if they need help to create visually attractive webpages and finds use not just in the creation of web applications but also mobile apps.

The language's syntax is pretty similar to that of HTML and XHTML, which work well in synchronization and combination of one another. The Style sheets included in CSS consist of a selector and a declarator. The simple syntax of the language uses several English language words to define the styling properties.

#### **4.4.3 JavaScript**

JavaScript is one of the most popular and dynamic programming languages used for creating and developing websites. This language is capable of achieving several things including controlling the browser, editing content on a document that has been displayed, allowing client-side scripts to communicate with users and also asynchronous communication. It was developed by Netscape and borrows a lot of its syntax from C language. JavaScript is used very widely and effectively in creating desktop applications as well as for developing games.

One of the best things about JavaScript for you as a developer or a website owner is that this is one of the few programming languages that are accepted and supported by all the major browsers without the need of any compilers or plug-ins. It can also be worked with on platforms that are not web-based, for example-desktop widgets and PDF docs. This is a multi-paradigm language which means that it has a combination of features. Also, JavaScript supports functional and object-oriented programming styles.

The features of a language define the way it will work, the way it responds, how easy is its code and what it can achieve. The following are some of the main features of JavaScript programming language for your reference:

- **Structured:** JavaScript is a highly structured language with a proper and planned syntax that has been derived from C. This language too has a function scoping by it lacks block scoping, unlike C. It too differentiates between statements and expressions, just like the fundamental C web programming platform.
- **Dynamic:** The types in JavaScript are not related with variables but with values. This is a dynamic programming language that enables you to test the type of an object in many different ways. Also, this programming language is object-oriented where all the objects are associative arrays.
- **Functional:** All functions in JavaScript are objects and are all first-class. They are associated with their own functions as well as characteristics. For example, a function within a function is called a nested function whereas this language also supports anonymous function.

#### 4.4.4 Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation and other interface components.

Bootstrap is a web framework that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font, and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The end result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap also comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields.

#### 4.4.5 Java Servlet

A Java servlet is a Java software component that extends the capabilities of a server. Although servlets can respond to many types of requests, they most commonly implement web containers for hosting web applications on web servers and thus

qualify as a server-side servlet web API. Such web servlets are the Java counterpart to other dynamic web content technologies such as PHP and ASP.NET.

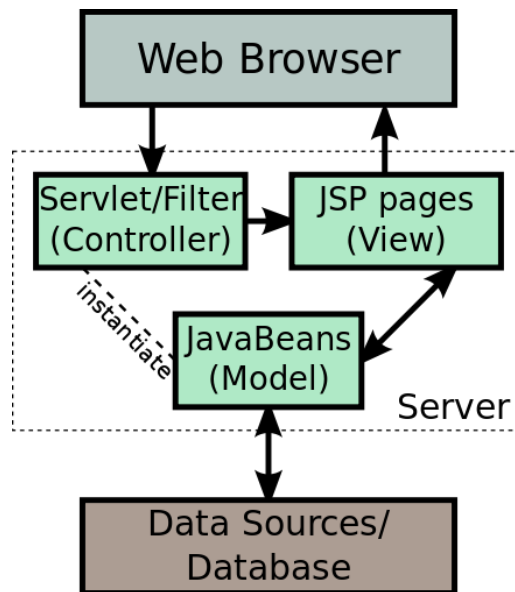
A Java servlet processes or stores a Java class in Java EE that conforms to the Java Servlet API, a standard for implementing Java classes that respond to requests. Servlets could in principle communicate over any client-server protocol, but they are most often used with the HTTP. Thus "servlet" is often used as shorthand for "HTTP servlet". Thus, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML but may be other data such as XML and more commonly, JSON. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL mapping.

A Servlet is an object that receives a request and generates a response based on that request. The basic Servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. Servlets can be generated automatically from Java Server Pages (JSP) by the Java Server Pages compiler. The difference between servlets and JSP is that servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML.

#### **4.4.6 JSP**

JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language.

To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.



**Figure 4-6:** JSP Model architecture

Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime, therefore JSP is a Servlet; each JSP servlet is cached and re-used until the original JSP is modified.

JSP can be used independently or as the view component of a server-side model–view–controller design, normally with JavaBeans as the model and Java servlets (or a framework such as Apache Struts) as the controller.

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, such as HTML, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, contain Java bytecode rather than machine code. Like any other Java program, they must be executed within a Java virtual machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment.

We preferred to use JSP rather than JSF because JSP is suitable for small scale application, that doesn't need any complications.

## **5 SYSTEM CONFIGURATION AND GUIDE**

### **5.1 DESKTOP APPLICATIONS CONFIGURATION**

#### **5.1.1 Software Requirements:**

- Windows OS (Windows 7, 8 or 10).
- MySQL Server.
- Apache Server.

#### **5.1.2 Hardware Requirements:**

- 2.8 GHz Processor or Above.
- RAM 8 GB or Above.
- HDD 15 GB Hard Disk Space or Above.
- Local Area Network.

### **5.2 WEBSITE CONFIGURATION**

#### **5.2.1 Software Requirements:**

- Any OS.
- Browser.
- Connection to the same LAN or WAN.

#### **5.2.2 Hardware Requirements:**

- 2.8 GHz Processor or Above.
- RAM 15 GB or Above.

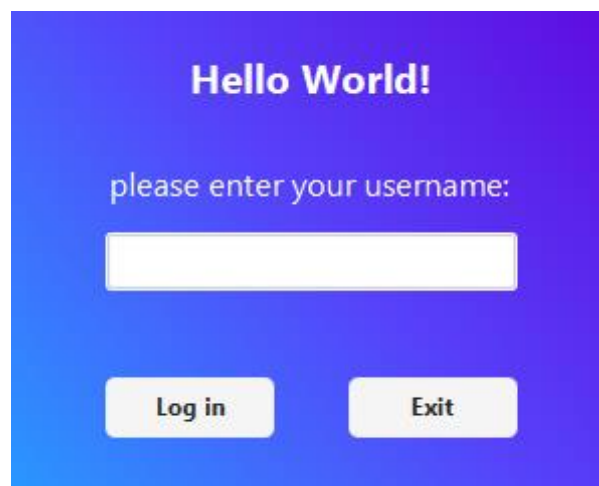
## 5.3 HOW TO USE?

### 5.3.1 Desktop application (Student-side and Judge-side)

Before you start using the application, you should create a network inside the place that held the quiz or the contest. You should make sure that the MySQL server and Apache server is working well on the judge server. We designed this light-weighted environment to help you held quiz or contest with minimal software and hardware requirements. You don't need a separate server device to hold your quiz or contest.

Each student should use the student interface that is shown in Figure 5-1.

The student enters his username or ID (without spacing). Then the student is asked to enter the Judge IP to enter the system. Judge IP is important because it allows the students and judge to communicate over the network using specifically TCP.

The image shows a desktop application window with a blue-to-purple gradient background. At the top, the text "Hello World!" is displayed in white. Below it, the prompt "please enter your username:" is shown in white. Underneath the prompt is a white rectangular text input field. At the bottom of the window, there are two white buttons with black text: "Log in" on the left and "Exit" on the right.


**Figure 5-1:** Student Login Interface

The image shows a desktop application window with a blue-to-purple gradient background. At the top, the text "Waiting Judge to start contest .." is displayed in white. Below it, the label "Judge IP:" is shown in white, followed by a white rectangular text input field. At the bottom center of the window, there is a single white button with black text that says "Start".

**Figure 5-2:** Student Wait Interface

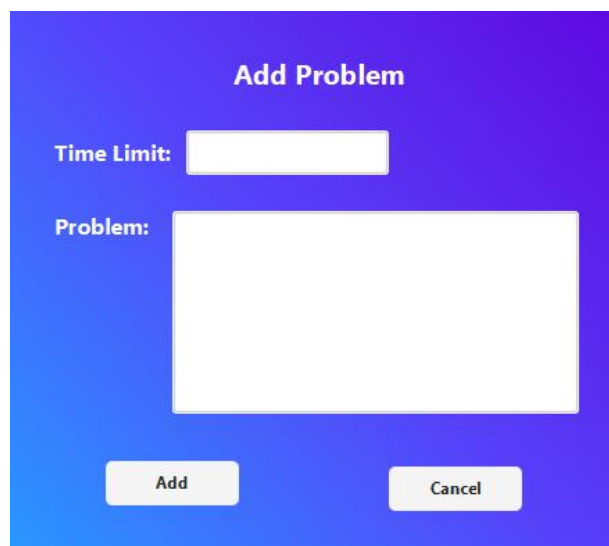


Then the student is asked to wait until the judge starts the contest. While the student is waiting for the judge, he should enter the time period of the contest and the problems, using the judge interface shown in Figure 5-3.

The image shows a web interface with a blue-to-purple gradient background. At the top, the text "Hello World!" is displayed in white. Below it, the label "Contest Time:" is followed by two white input boxes separated by a colon. Underneath these inputs is a white button with the text "Add Problems". At the bottom of the interface are two more white buttons, "Start" on the left and "Exit" on the right.

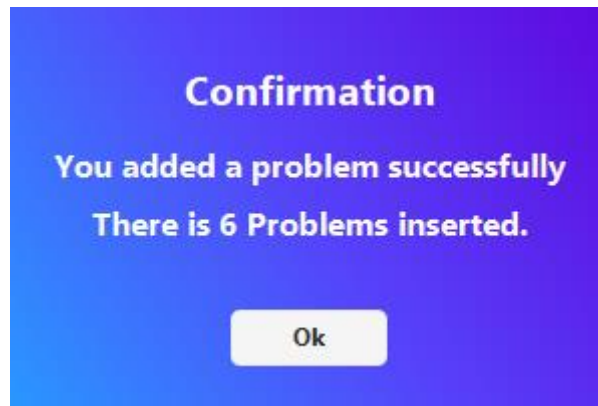
**Figure 5-3:** Judge Wait Interface

At this point, Judge enters the problems by clicking the button (Add Problems), so it could be viewed later at the contest from the browser. By using the interface in Figure 5-4, the judge can add the problems. The added problem is recorded in the database.

The image shows a web interface with a blue-to-purple gradient background. At the top, the text "Add Problem" is displayed in white. Below it, the label "Time Limit:" is followed by a white input box. Underneath this is the label "Problem:" followed by a large white text area. At the bottom of the interface are two white buttons, "Add" on the left and "Cancel" on the right.

**Figure 5-4:** Add Problem interface

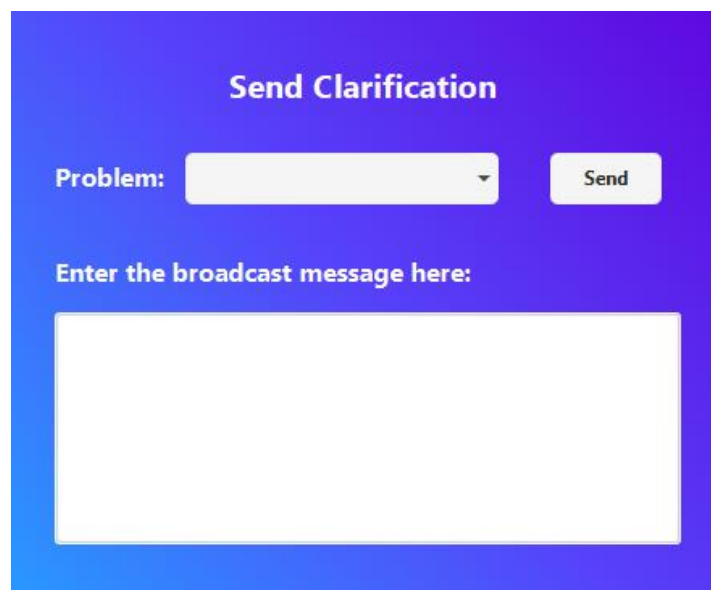
The Judge enters each problem with its time limit, the click (Add) button, and if the problem is added successfully to the database, it shows a confirmation message, as shown in Figure 5-5.



**Figure 5-5:** Add Confirmation Interface

When the judge finish enters all problems, he clicks (Cancel) button, then he will be back to Figure 5-3. Then, he clicks (Start) button to start the contest. When he clicks start, the student now can enter the system and answer the problems.

The Judge interface at this moment is shown in Figure 5-6.



**Figure 5-6:** Judge Main Interface

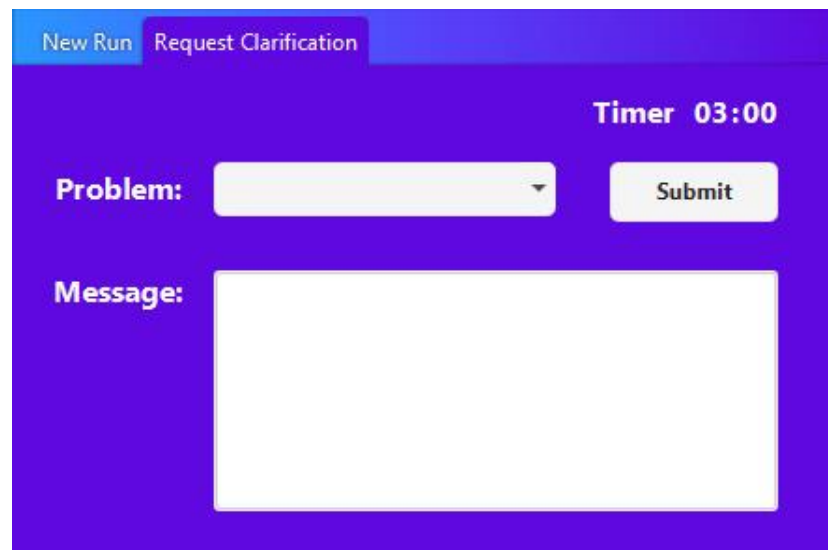
The Main Interface of Judge let him choose a specific problem to send a broadcasted message about it to all of the students.

Basically, the student interface is a Tab Pane consists of two tabs, each of them contains a counter that counts down the time of the contest. The first one is (New Run), where the student can choose the Problem ID and the Path of the file to submit a new run. The programming language the student used to solve a problem is detected by the system.



**Figure 5-7:** Student Main interface - Tab 1

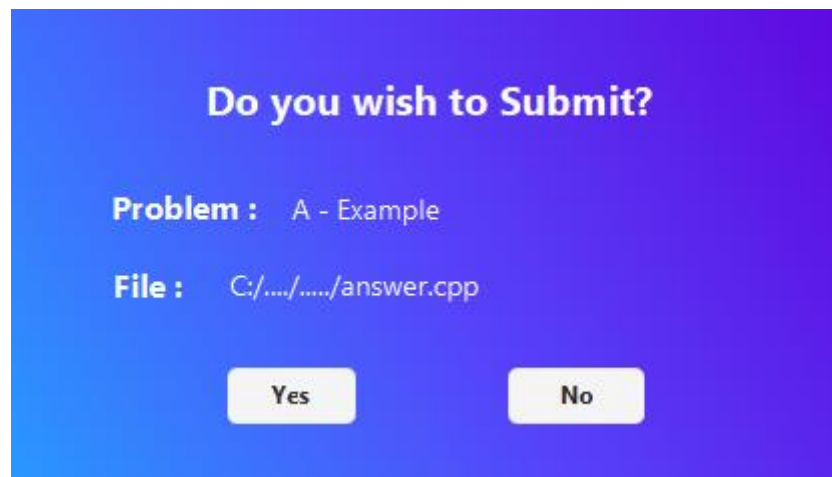
The second tab is (Request Clarification), where the students are able to send a clarification question to the judge. The student chooses a specific problem and write the body of the message then click (Submit).



**Figure 5-8:** Student Main interface - Tab 2

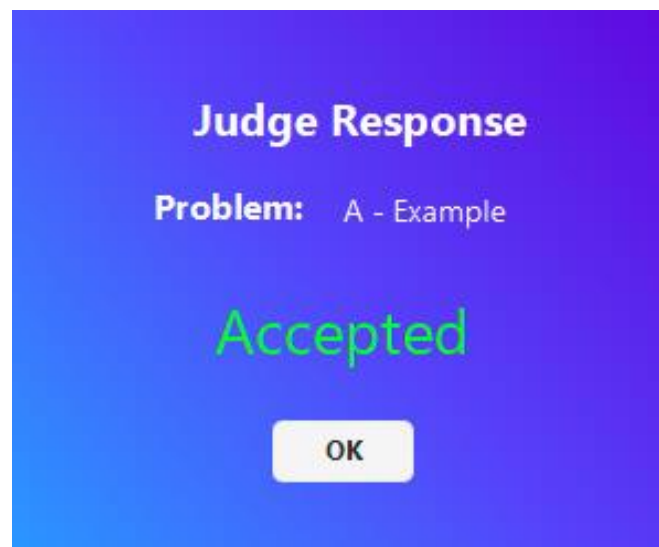
When the student chooses a problem to answer, selects the answer file from the (Browse) button and clicks (Submit) button, it shows a confirmation message.

The importance of this confirmation message for the student is to make sure that he chooses the correct problem and the correct file path.



**Figure 5-9:** Student Submit Confirmation Message

After the automatic judging complete, the student receives the result as shown in Figure 5-10. There is 5 types of responses, 0 Compilation error (Blue), 1 Accepted (Light Green), 2 Runtime error (Red), 3 Time Limit Exceeded (Grey) and 4 Wrong Answer (Black).



**Figure 5-10:** Automatic Response of the Result

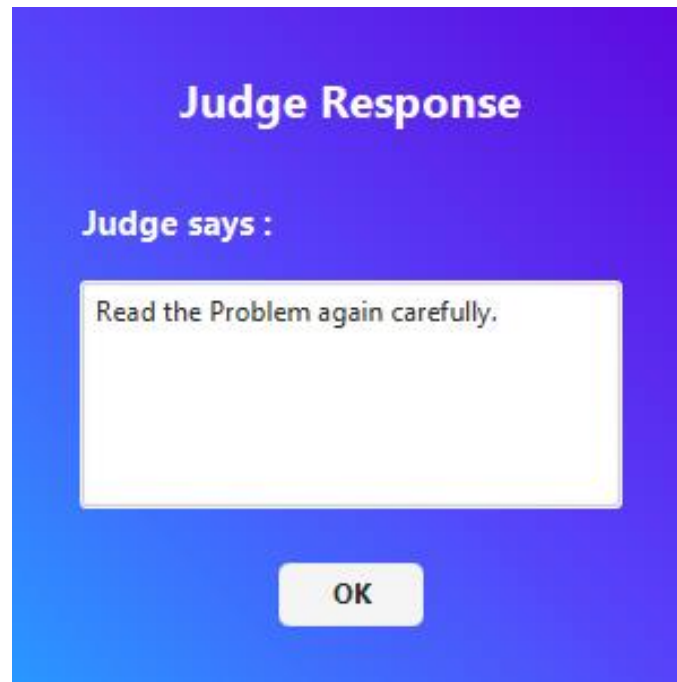
When the student tries to send a clarification message, and after he chooses a specific problem and writes the body of the message, on the judge side, the message appears as follows.



The image shows a digital interface for a judge to respond to a question. The background is a blue gradient. At the top, the text "Judge Response" is displayed in white. Below this, the label "Question:" is shown in white. A white rectangular box contains the question text "What is meant by X in Problem B?". Underneath the question box, the label "Enter the response here:" is displayed in white. Below this label is a larger white rectangular box for the judge's response. At the bottom of the interface, there are two white buttons with rounded corners. The left button is labeled "Broadcast" and the right button is labeled "Respond", both in black text.

**Figure 5-11:** Judge Response Interface for Judge

Figure 5-11 let the judge enters the response of the question, and give him two options, the first one is to broadcast the message to all students if he sees that the question is important and need to tell all of the students about it. The second option is just to reply to the student who asked the question. Figure 5-12 shows the message that appears at the student side when the judge broadcast the message or respond to it.

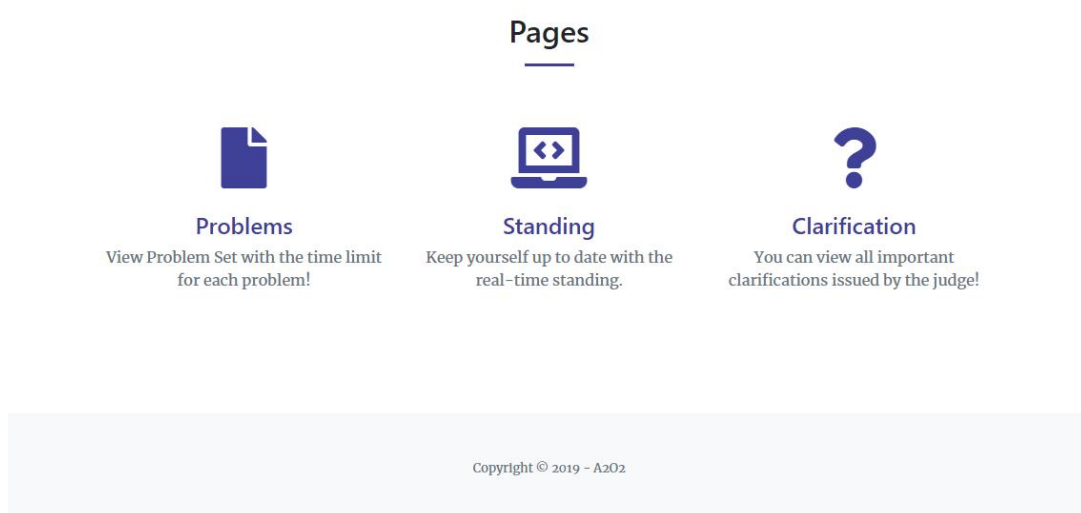
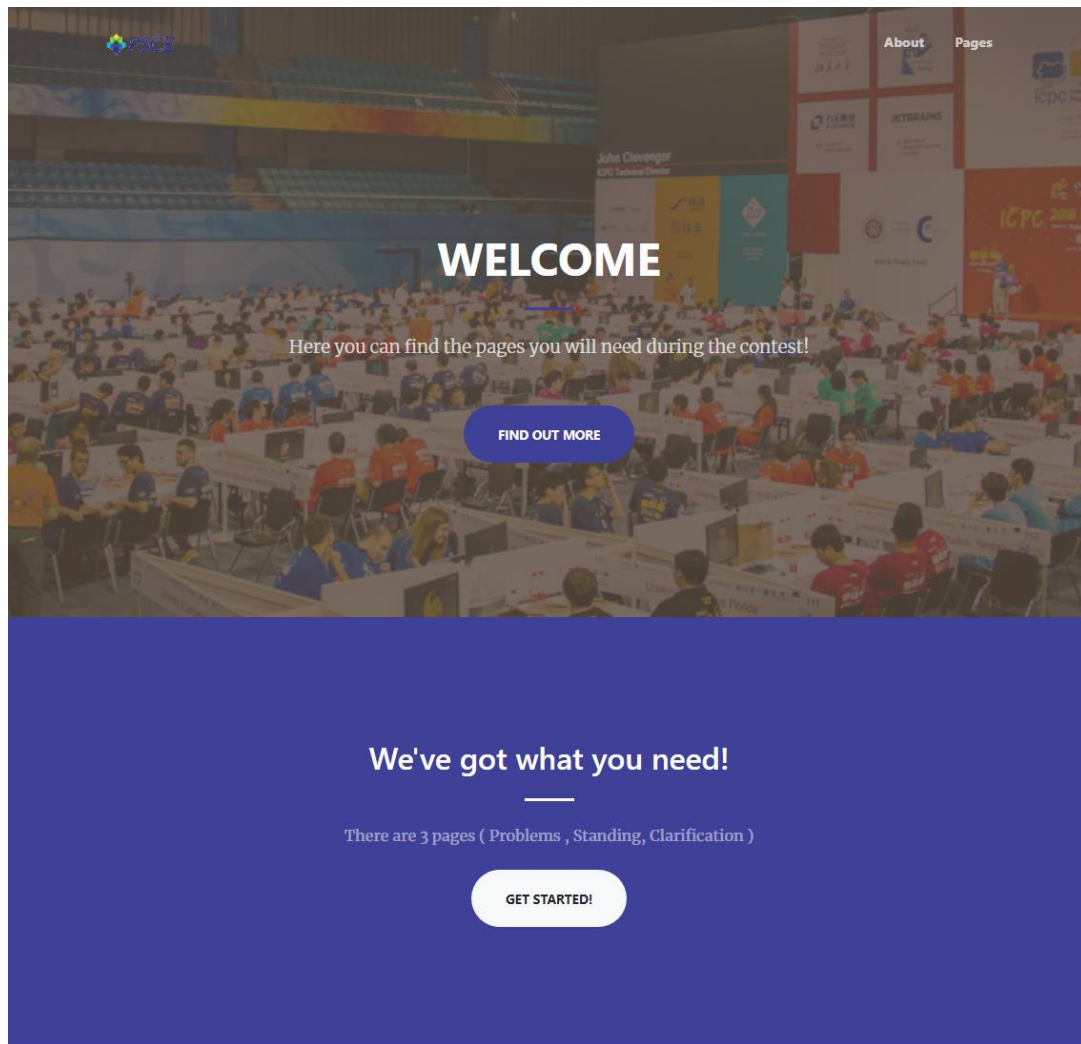


**Figure 5-12:** Judge Response Interface for Student

### **5.3.2 Web Application (Problems, Clarifications, and Standing)**

When the contest starts, both judge and student are able to view problems, clarifications, and standing through a web site that could be accessed locally on the site.

The home page is just a landing page that serves as the entry point for the web pages inside the website. Basically, there is three web pages problems, clarifications, and standing. They help the student find all that he will need during the contest in one place.



**Figure 5-13: Home Page**

The first webpage is the problems, it contains the problem ID, time limit and the problem itself.

Problems		<a href="#">Clarification</a> <a href="#">Standing</a> 2019
ID	TIME LIMIT	PROBLEM
A	2	<p>When <math>a \neq 0</math>, there are two solutions to <math>ax^2 + bx + c = 0</math> and they are</p> $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$
B	2	$\sum_{i=0}^n i^2 = \frac{(n^2 + n)(2n + 1)}{6}$

**Figure 5-14:** Problems Page

The second webpage is the clarification, it contains all the clarification that has been broadcasted during the contest. It consists of the question associated with its answer.

Clarification		<a href="#">Standing</a> <a href="#">Problems</a> 2019
NUMBER	QUESTION	ANSWER
1	what is your name	amr
2	what is your age	3312
3	what is your mission	null
4	what is your fav thing	MLW

**Figure 5-15:** Clarification Page

The third webpage is the Standing, is contains all submission of the students during the contest. It also contains each team rank, number of solved problems, and number of tries for each problem.



Standing				<a href="#">Clarification</a> <a href="#">Problems</a> 2019													
RANK	TEAM NAME	SOLVED / TIME		A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	team2	5	201		1	1		2						1	1		
2	asd	1	0											1			
3	tasd	1	0												1		
4	tagtag	1	22	1						1							
5	team1	1	1645	1													
6	sola	0	0				1										
7	xteam	0	0					1									
8	solaTeam	0	0				1										

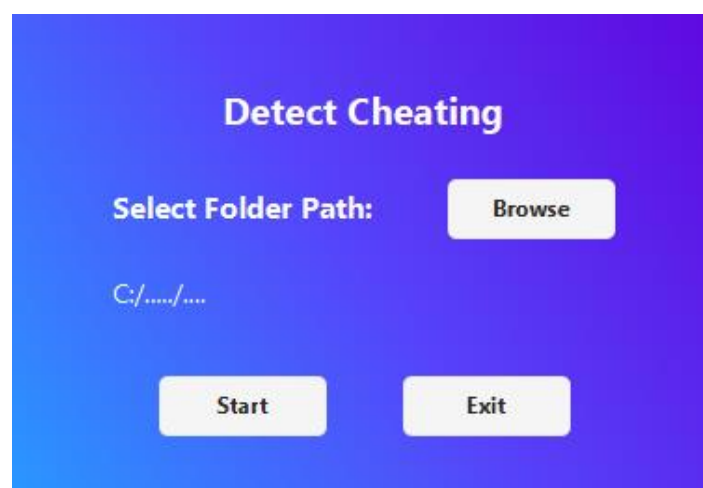
Cell colours	
Solved	
Tried, incorrect	
Untried	

**Figure 5-16:** Standing Page

### 5.3.3 Desktop Application (Plagiarism)

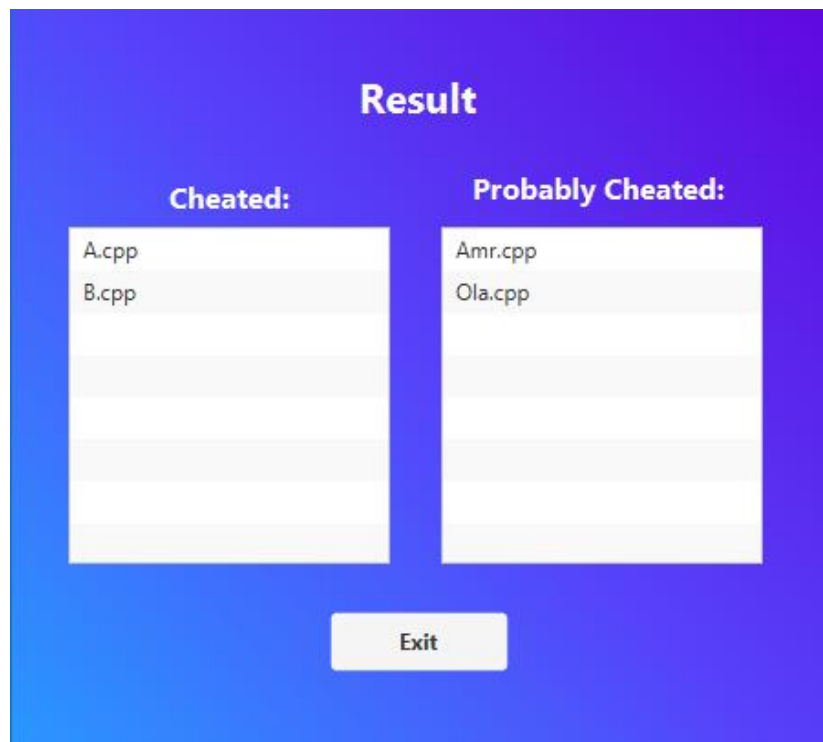
After the contest is finished, the plagiarism program allows the judge to check if any student cheats during the contest. The Result is divided into two types, the first type who is 100% cheated, and the second type is probably cheated.

The Judge is asked to locate the archive folder that contains the submissions of the student. Then click (Start) Button as shown in Figure 5-17.



**Figure 5-17:** Plagiarism Main Interface

The system checks all archived files, and output the result as shown in Figure 5-18.



**Figure 5-18:** Plagiarism Result Interface

## **6 CONCLUSION AND FUTURE WORK**

### **6.1 CONCLUSION**

In this project, a complete judging system for students or contestants. Our system supports automated judging that saves much time for users while solving tasks or any problems. This lets students solve tasks in an easy way, unlike traditional methods. The judge can discover the case that if two or more student cheating with each other. This help judges in the event of a contest or any other tests. Our system will be available all time with good efficiency. Also, it can accept a huge number of students and support more than one contestant without any degradation. The system has two interfaces, one for student or contestants and other for judges. Students can ask about anything they need and receive an answer from judges. Also, the judge can send a broadcast message to students in case of a contest. It is a complete system that can be applied in our faculty and its results are reliable to be taken. We will show some of our real test that made on that system in our faculty.

### **6.2 FUTURE WORK**

Our system support submitting problem solution in two languages, C++ and Java, we are looking forward to supporting other popular programming languages such as Python, JavaScript, and PHP.

Allow further distribution of the system is important to achieve higher performance with minimum hardware requirements.

Our system is using the minimum edit distance algorithm for the cheating detection system. This algorithm has an acceptable accuracy of detection, but we are going to improve this algorithm by adding a step of Lexical Analysis. This step will help us to uniform the variables, so we could achieve a higher accuracy of detection.

Our system can help people, especially in the academic field to hold locally contests or quizzes. But we also want to add online access to our environment. So, anyone from any place and any time could access the system and create his own local contest or quiz.

## *Chapter Seven*

### **7 REFERENCES**

- [pc2.ecs.csus.edu/pc2desc.html](http://pc2.ecs.csus.edu/pc2desc.html)
- <https://open.kattis.com/>
- [cs.utah.edu/~germain/PPS/Topics/problem\\_solving.html](http://cs.utah.edu/~germain/PPS/Topics/problem_solving.html)
- <http://www.extremeprogramming.org/>
- <https://www.scribd.com/document/63413837/Data-Flow-Diagram-DFDs>
- <https://www.scribd.com/document/63785900/System-Analysis-and-Design>
- <https://www.scribd.com/document/375687902/The-Agile-Methodology>
- <https://www.scribd.com/document/26810581/Software-Testing-Methods>
- <https://www.tug.org/begin.html>
- <https://www.mathjax.org/>
- [https://docs.oracle.com/javase/10/docs/api/javafx/fxml/doc-files/introduction\\_to\\_fxml.html](https://docs.oracle.com/javase/10/docs/api/javafx/fxml/doc-files/introduction_to_fxml.html)

## APPENDIX

Pictures from our real contest test.





